UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

AgroNET: uma plataforma para gerenciamento de grandes volumes de dados na Agricultura de Precisão

Mauricio Alex Zientarski Karrei

Passo Fundo

2017

UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

AGRONET: UMA PLATAFORMA PARA GERENCIAMENTO DE GRANDES VOLUMES DE DADOS NA AGRICULTURA DE PRECISÃO

Mauricio Alex Zientarski Karrei

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada na Universidade de Passo Fundo.

Orientador: Prof. Dr. Willingthon Pavan

Passo Fundo

2017

CIP – Catalogação na Publicação

K18a Karrei, Mauricio Alez Zientarski
Agronet : uma plataforma para gerenciamento de dados na agricultura de precisão / Mauricio Alex Zientarski Karrei. – 2017.
74 f. : il. color. ; 30 cm.

Orientador: Prof. Dr. Willingthon Pavan.
Dissertação (Mestrado em Computação Aplicada) – Universidade de Passo Fundo, 2017.

1. Software. 2. Agricultura. 3. Informática na agricultura. I. Pavan, Willingthon, orientador. II. Título.

CDU: 631:004

Catalogação: Bibliotecária Marciéli de Oliveira - CRB 10/2113

UPF Universidade de Passo Fundo | PPGCA Programa de Pós-Graduação em Computação Aplicada Instituto de Ciências Exatas e Geociências - ICEG

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

MAURICIO ALEX ZIENTARSKI KARREI

Aos vinte e um dias do mês de março do ano de dois mil e dezessete, às 16 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"AgroNET: uma Plataforma para Gerenciamento de Grandes volumes de dados na Agricultura de Precisão"**, de autoria de Mauricio Alex Zientarski Karrei, acadêmico do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, o aluno preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores Willingthon Pavan, José Mauricio Cunha Fernndes e Alexandre Tagliari Lazzaretti. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou o candidato APROVADO. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para o acadêmico apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. Willingthon Pava
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. José Maurício Cunha Fernandes
(Avaliador Interno)

Prof. Dr. Alexandre Tagliari Lazzaretti
(Avaliador Externo)

Prof. Dr. Rafael Rieder
Coordenador do PPGCA

AGRONET: UMA PLATAFORMA PARA GERENCIAMENTO DE GRANDES VOLUMES DE DADOS NA AGRICULTURA DE PRECISÃO

RESUMO

A tecnologia aplicada a agricultura gera atualmente uma grande quantidade de dados a cada instante, oriundas de diferentes fontes, locais e formatos. A interpretação destes dados pode auxiliar no manejo das culturas, apoiando na busca pela redução no uso de insumos e no aumento da quantidade e da qualidade da produção. Entretanto, analisar e propor estratégias para manipular a grande massa de dados gerados por sensores e dispositivos eletrônicos agrícolas, frequentemente são apontados como alguns dos principais desafios para a Agricultura de Precisão. Para tanto, a pesquisa realizada neste trabalho objetivou o desenvolvimento de uma plataforma computacional capaz de receber, tratar e disponibilizar dados provenientes da Agricultura de Precisão. Como resultado, obteve-se uma plataforma e um conjunto de ferramentas que possibilitam o armazenamento e recuperação de grandes volumes de dados, na forma espaço/geo/temporal. Esta plataforma permite, por meio de uma API REST, a manipulação de dados de maquinários, implementos, sensores e dispositivos eletrônicos agrícolas, assim como a manipulação de dados geoespaciai. A fim de validar a plataforma e suas ferramentas, desenvolveu-se uma interface *web* para acesso e manipulação dos dados da plataforma de forma visual.

Palavras-Chave: Agricultura de Precisão, Armazenamento de dados, Arquitetura Orientada a Serviços.

AGRONET: A PLATFORM FOR MANAGING LARGE AMOUNTS OF DATA IN PRECISION AGRICULTURE

ABSTRACT

Technology applied to agriculture generates a large amount of data from different sources, locations and formats. The interpretation of these data can help in the management of the crops, supporting in the reduction in the use of inputs and in the increase in the quantity and the quality of the production. However, analyzing and proposing strategies to manipulate the large amount of data generated by sensors and agricultural electronic devices are often pointed out as some of the key challenges for Precision Agriculture. Therefore, the research carried out in this work aimed the development of a computational platform capable of receiving, processing and making available data from Precision Agriculture. As a result, we obtained a platform and a set of tools that allow the storage and retrieval of large amounts of data, in the space/geo/temporal format. This platform allows, through a REST API, the data manipulation of machinery, implements, sensors and agricultural electronic devices, as well as the manipulation of geospatial data. In order to validate the platform and its tools, a textit web interface has been developed for access and manipulation of the platform data in a visual way.

Keywords: Precision Agriculture, Data Storage, Services Oriented Architecture.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A constante evolução dos sistemas de produção agrícola tem se mostrado cada vez mais como uma importante engrenagem para a produção de alimentos. O crescimento da população e o surgimento de doenças são desafios impostos ao mercado da produção de alimentos [1]. Neste sentido, a tecnologia aplicada no campo tem se tornado aliada dos produtores, trazendo novos recursos e equipamentos que auxiliam a melhora do processo de manejo do campo.

A Agricultura de Precisão (AP), introduzida em meados dos anos 80, é definida por McBratney [2] como a agricultura que aumenta o número de decisões corretas, trazendo benefícios ao longo do tempo. Para tanto, tecnologia como sensores eletrônicos e GPS, podem ser utilizadas para coletar dados do campo, os quais podem auxiliar no aumento da quantidade e da qualidade da produção e também na diminuição de insumos, centrando no desenvolvimento sustentável e tendo em conta a rentabilidade associadas a benefícios ambientais e sociais.

Dessa forma, a medida em que as máquinas e sensores eletrônicos inteligentes surgem no campo, os benefícios de produtividade e processos de cultivo tornam-se, também, cada vez mais evoluídos. Isto está intimamente relacionado com o conceito de *Smart Farming*, ou fazendas inteligentes, o que significa que dispositivos inteligentes, conectados à internet, possuem um papel fundamental sobre o auxílio do controle da fazenda, fazendo com que a inteligência embarcada em sensores seja capaz de efetuar ações autônomas ou viabilizar a execução de ações de forma remota, tornando o processo de gestão agrícola mais facilitado com o uso deste tipo de tecnologia [3].

Equipamentos como estações meteorológicas, monitores de plantio, monitores de pulverização e taxa variável englobam o setor da AP e auxiliam os profissionais da área no manejo do campo. Percebe-se, portanto, que a AP apresenta-se como um setor responsável por realizar a coleta dos mais complexos e variados tipos de informações produzidas no espaço e no tempo, visto sua diversidade de equipamentos, operando nas mais diversas situações [4, 5]. Todavia, as informações que são geradas pelos equipamentos de AP, necessitam ser transmitidas para algum local a fim de serem analisadas e processadas, com o objetivo de gerar informações relevantes aos profissionais do campo.

No entanto, a tecnologia representada pela AP é responsável por gerar um grande volume de dados e com altas variabilidades de tipos. Neste sentido, a grande maioria das dificuldades referem-se ao processo de armazenamento e interpretação dos dados, fazendo com que a necessidade de extrair resultados concretos destes dados seja um dos principais objetivos das ferramentas de AP. Diante disso, conceitos de *Big Data* são geralmente aplicados a este cenário, por representar a capacidade de extrair informações de grandes volumes de dados de forma rápida [3]. Além disso, a alta demanda por conexões com os dispositivos agrícolas também apresenta-se como um acelerador do crescimento do tráfego de dados entre plataformas computacionais agrícolas, necessitando altos custos para prover conectividade de modo escalável aos dispositivos da AP [4, 6].

Os dados geoespaciais, geralmente caracterizados por Sistemas de Informação Geográfica (SIG), do inglês *Geographic Information Systems* (GIS), têm sido cada vez mais utilizados para a AP. Sistemas de Posicionamento Global (GPS) e sistemas geoespaciais são utilizados, tornando possível a aquisição, processamento e utilização de dados espaciais para a geração de produtos, como mapas de produtividade e representação geográfica de áreas de cultivo, assim como delimitações físicas das propriedades rurais [7].

Dessa forma o uso de ferramentas de apoio, favorecem a extração dos mais variados tipos de informações oriundo dos equipamentos de AP. Para tanto, se fazem necessárias tecnologias de alto desempenho para a recepção, processamento e disponibilização dos dados coletados, assim como meios de armazenamento de dados capazes de concentrar um grande volume de informações, incluindo o suporte à tipos de dados complexos e heterogêneos. Assim, soluções computacionais podem ser utilizadas para realizar o processo de gestão e manipulação destes dados, trazendo consigo inúmeras vantagens, influenciando direta na produtividade no campo. Portanto, ferramentas de AP, podem auxiliar os profissionais da área agrícola, de mudar a distribuição e o tempo de aplicação de produtos químicos, utilizando informações como a variabilidade espacial e temporal do campo [4].

Visando apresentar uma solução para o armazenamento e a disponibilização de dados provenientes da AP, este trabalho almejou o desenvolvimento de uma plataforma computacional para recebimento, tratamento e disponibilização destes dados. Para isso, faz-se necessário o desenvolvimento de uma estrutura de banco de dados capaz de armazenar grandes volumes de dados, atendendo a sua alta variabilidade de tipos. Também, o desenvolvimento de uma estrutura de banco de dados capaz de suportar o armazenamento de dados geoespaciais foi determinada, visando englobar o cenário de SIG na plataforma. Por fim, este trabalho também almejou a implementação de uma estrutura de serviços *web* capaz de oferecer a fácil troca de mensagens entre dispositivos da API, e integração com diferentes sistemas agrícolas por meio de um padrão conhecido com suporte a entrada e a manipulação de diferentes parâmetros e tipos de dados da AP.

Este trabalho foi organizado em quatro etapas independentes descritas no formato de artigo científico e inseridas neste trabalho por meio de capítulos distintos. Sendo assim, o Capítulo 2 apresenta uma revisão de literatura acerca dos conceitos e tecnologias utilizadas no desenvolvimento deste trabalho. O Capítulo . 3 apresenta a plataforma AgroNET Services, cujo contexto é o armazenamento e disponibilização de dados da AP. No Capítulo 4 é apresentado o desenvolvimento da plataforma AgroNET GIS, cujo objetivo é uma solução para armazenamento de dados geoespaciais para a AP. No Capítulo 5 é apresentado a plataforma AgroNET View, cujo objetivo é o desenvolvimento de uma plataforma para acesso, visualização e geração de dados da AP. Por fim, os Capítulos 6 e 7 apresentam as conclusões e trabalhos futuros deste trabalho, respectivamente.

2. REVISÃO DE LITERATURA

Este capítulo apresenta uma revisão de literatura acerca dos métodos e técnicas de armazenamento de dados e arquiteturas de sistemas, apresentando ferramentas e recursos que podem ser utilizados para a construção da aplicação proposta neste trabalho.

2.0.1 Banco de Dados

O uso de tecnologias de banco de dados está cada vez mais presente nas áreas em que os computadores são utilizados. Isto significa que estas tecnologias mostram-se como um componente necessário para as mais diversas atividades do cotidiano, como por exemplo a compra de mercadorias em um supermercado, onde na maioria das vezes existe um sistema que mantém atualizado o número de itens do estoque, existindo portanto uma base de dados que contém estes valores.

As tecnologias de bancos de dados surgiram em meados da década de 60, com o objetivo de suprir a necessidade dos computadores para armazenar e gerenciar informações de forma permanente e que possibilitasse acesso automatizado a estes dados. Para gerenciar estes dados, foram criados os Sistemas de Gerenciamento de Banco de Dados (SGBD's), os quais foram destinados ao controle de todos os aspectos de um Banco de Dados, como por exemplo, a gravação, acesso, controles de consistência, leitura e gravação de dados [8]. A Figura 1 apresenta a estrutura básica de um banco de dados.

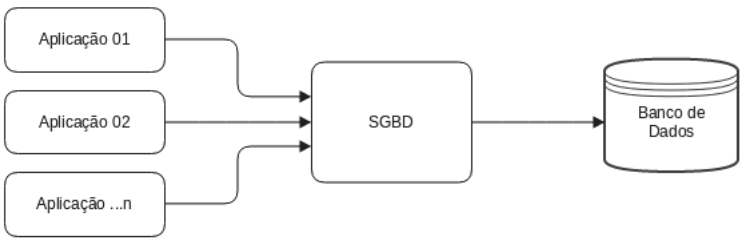

Figura 1. Estrutura Básica de Banco de Dados

Segundo Elmasri [9] um banco de dados é uma coleção lógica de dados relacionados com algum significado inerente. Dessa forma, um banco de dados visa representar aspectos do mundo real, que algumas vezes é chamado de mini-mundo ou Universo de Discurso (UoD, *Universe of Discourse*).

Ainda, Medeiros [10] conceitua banco de dados como sendo um conjunto de dados organizados e com o objetivo de armazenamento, e dotado de mecanismos de manipulação para obtenção de informações e recuperação posterior, dentro de um sistema de informação.

Um banco de dados pode ser gerado e mantido manualmente ou automaticamente. Por sua vez, um banco de dados automatizado oferece recursos de armazenamento e organização dos dados

por meio de um software. Os softwares que possibilitam o usuário criar e manter informações em um banco de dados, são chamados de Sistemas Gerenciadores de Banco de dados (SGBD) [11].

2.0.1.1 Modelo Relacional

Para solucionar problemas das limitações dos Modelos de Bancos de Dados da década de 60, Dr. Codd definiu um novo modelo para resolver problemas de redundância de dados, dependência física e falha na integridade [11]. Um banco de dados Relacional, é composto por tabelas ou relações, que formam um conjunto não ordenado de linhas compostas por uma série de campos que guardam valores [12].

O modelo usa conceitos de uma relação matemática e tem sua base teórica na teoria dos conjuntos e na lógica e predicados de primeira ordem. Sendo assim, este modelo possui como características a possibilidade de relacionar várias tabelas, evitando assim a redundância no armazenamento de dados. O Relacionamento pode ocorrer de três formas: um para um, um para muitos, muitos para muitos. [13] [11].

Ainda, a estrutura de um banco de dados de Modelo Relacional é formada por tabelas, que representam dados, bem como suas relações. Este modelo é composto por uma coleção de operadores, que podem ser acessados pela linguagem SQL, respeitando uma série de de restrições de integridade que definem um conjunto consistente de estados dos elementos [14].

Contudo, os bancos de dados relacionais podem apresentar boa efetividade para tipos simples de dados e envolvendo poucas relações. A medida que aplicações complexas de bancos de dados são requeridas, alguns recursos podem se tornar ineficientes ou não são encontrados no modelo relacional de banco de dados [15]. Limitações, de como por exemplo a ausência de um tipo de dados definido pelo usuário, dificultade de representar objetos do mundo real e divergências entre os objetos da linguagem de programação e as estruturas de banco de dados são fatores que podem impossibilitar o desenvolvimento de aplicações que necessitem de uma base de dados mais complexa.

2.0.1.2 Modelo Objeto-Relacional

Os sistemas com bancos de dados de modelo relacional, já eram muito empregados quando ocorreu o surgimento do modelo orientado a objetos, e descartar as aplicações já desenvolvidas seria inviável, porém havia a necessidade de utilizar recursos que eram propostos pelo modelo orientado a objetos, como por exemplo a representação de dados complexos [16]. Dessa forma, o modelo objeto-relacional foi criado, onde a sua implementação física se baseia no modelo relacional, permitindo manter o mesmo princípio de criação utilizando a linguagem SQL, porém a semântica da aplicação é representada por objetos.

O modelo Objeto-Relacional é constituido por uma integração dos conceitos de modelos Orientado a Objetos e Relacional. Este modelo de banco de dados visa suprir a falta de recursos dos sistemas relacionais, agregando características presentes no modelo de banco de dados Orientado a

Objetos em tabelas, utilizando da linguagem SQL para manipulação das informações. Além disso, o modelo Objeto-Relacional oferece recursos como encapsulamento, herança, consultas avançadas e tipos de dados definidos pelo usuário [17] [18].

Para este novo conceito de modelo de banco de dados, surgiu a necessidade de uma linguagem padrão para o uso do SGBDOR (Sistema de Gerenciamento de Dados Objeto-Relacional). Com isso, definiu-se em 1999 a linguagem SQL3 (*Structured Query Language* 3). O SQL3 é uma estensão da linguagem SQL2, complementando seus recursos na versão anterior e tornando-a voltada para o modelo relacional. Dessa forma, o SQL3 define recursos para manipulação e controle das estruturas do banco de dados objeto-relacional, tornando-se a linguagem padrão para este modelo de banco de dados [19] [15].

Em contra partida, Silberschatz *et al.* [17] citam que o modelo de banco de dados objeto-relacional pode oferecer um tempo de resposta maior, principalmente para operações com uma grande massa de dados envolvente. Entretanto, são elencadas técnicas que visam a redução do tempo de operação em um banco de dados, como por exemplo a criação de índices.

2.0.1.3 Modelo Orientado a Objetos/Documentos

A necessidade de manipulação e armazenamento de dados complexos vem sendo uma demanda crescente no âmbito computacional. Essa necessidade fez com que paradigmas já existentes na programação, fossem adicionados também aos bancos de dados, como por exemplo o modelo orientado a objetos. Esta evolução, possibilitou que informações complexas, como por exemplo, gráficos, imagens, mapas, entre outros, passaram a ser armazenados diretamente em um banco de dados [18].

Sudarshan *et al.* [17] afirmam que bancos de dados tradicionais (relacionais) consistem em tarefas de processamento de dados com tipos de dados relativamente simples, se fazendo adequado para uma ampla gama de aplicações. Como os sistemas, e consequentemente os bancos de dados evoluíram de forma a exigir um esforço maior com tipos de dados mais complexos, tais como *computer-aided* e sistemas de informação geográfica, o modelo relacional emergiu como um obstáculo. A solução foi a introdução de bancos de dados com base em objetos e documentos, os quais permitem operar com tipos de dados complexos, além de oferecerem recursos de armazenamento de informações sem uma estrutura padrão pré-definida (em forma de documentos JSON por exemplo).

Para tanto, Elmasri [13] aponta ainda outra característica presente neste modelo de bancos de dados, onde os objetos/documentos podem ter uma estrutura de objeto de complexidade arbitrária, de forma a conter todas as informações necessárias que descrevem o elemento. Já nos sistemas de bancos de dados tradicionais, geralmente esta informação é dada de forma distribuída entre várias tabelas, levando a perda da correspondência direta entre um objeto do mundo real e sua representação no banco de dados.

Para Boscarioli [20] a modelagem de um banco de dados que utiliza deste paradigma está intimamente ligada aos diagramas de classes gerados para o sistema. Portanto, é possível manter

uma visão igualitária para o tratamento dos dados tanto na codificação, quanto na persistência dos objetos em um banco de dados. Proporcionando assim, uma visão próxima do mundo real tanto no ambiente de programação, quanto no banco de dados.

2.0.1.4 MongoDB

O MongoDB é um estilo de banco de dados orientado a documento escrito na linguagem de programação C++. Sua primeira versão foi lançada no ano de 2007, focado na alta performance, velocidade e escalabilidade. Dessa forma, o *design* do MongoDB oferece recursos para manter cópias do banco de dados em diferentes servidores, e no momento em que algum destas estruturas falhar, outro banco de dados pode entrar em operação instantaneamente de forma a restaurar os valores originais [21] [22].

O principal objetivo do MongoDB não é criar um banco de dados otimizado para o armazenamento de qualquer tipo de dados. Visto que este banco de dados não faz a utilização de linhas como no modelo relacional, e sim de objetos (documentos), o que o torna extremamente rápido e altamente escalável e flexível para operar com tipos de dados complexos. Por isso, algumas características não foram objetivadas pela estrutura do MongoDB, o que não o torna um candidato ideal para certas situações. Por outro lado, situações onde a consistência e a semântica transacional, podem ficar comprometidas, visto que o MongoDB não objetiva este aspecto na sua estrutura, MongoDB é capaz de atender as necessidades [22].

No que se refere ao armazenamento, o MongoDB define uma estrutura denominada BSON, a qual se baseia no formato JSON (*JavaScript Object Notation*). Ao contrário do modelo relacional, onde a estrutura é constituída por tabelas que armazenam dados individualmente, MongoDB armazena todas as informações juntas em um único documento. Dessa forma, MongoDB não possuí um esquema, o que possibilita que os dados possam ser atualizados individualmente ou independentemente de qualquer outro relacionamento de documentos. Como vantagem, MongoDB não necessita especificar a estrutura do documento com antecedência, além de oferecer maior desempenho, visto que mantém toda a estrutura dos dados relacionados em um só um lugar [22]. A Figura 2 apresenta a estrutura de inserção e consulta dos dados armazenados pelo MongoDB.

2.0.1.5 CouchDB

CouchDB, mantido pela fundação Apache, é caracterizado por ser um banco de dados robusto e estável na comunidade NoSQL (*Not Only SQL*). A primeira versão deste banco de dados foi lançada no ano de 2005 visando robustez e estabilidade entre os bancos de dados NoSQL. É caracterizada pela premissa de que as infraestruturas de hardwares e redes são eminentemente falhas. Por este motivo CouchDB oferece uma arquitetura descentralizada e é capaz de operar com a alta variabilidade no tráfego, atendendo desde situações com baixa quantidade de requisições até situações da

```
> db.media.insert( { "Type" : "CD",
…"Artist" : "Nirvana",
…"Title" : "Nevermind",
… "Tracklist" : [
… {
… "Track" : "1",
… "Title" : "Smells like teen spirit",
… "Length" : "5:02"
… },
… {
… "Track" : "2",
… "Title" : "In Bloom",
… "Length" : "4:15"
… }
… ]
…}
… )
```

```
> db.media.find()
{ "_id" : "ObjectId("4c1a86bb2955000000004076"), "Type" : "CD", "Artist" :
"Nirvana", "Title" : "Nevermind", "Tracklist" : [
    {
        "Track" : "1",
            "Title" : "Smells like teen spirit",
            "Length" : "5:02"
    },
    {
        "Track" : "2",
        "Title" : "In Bloom",
        "Length" : "4:15"
    }
] }
```

Figura 2. Estrutura de documento MongoDB [22]

ocorrência de um alto volume de solicitações, de forma a ajustar a performance conforme a demanda com mecanismos de tolerância a falhas [23] [24].

Este banco de dados foi construído para a Web. Sendo assim, os dados são armazenados em formato JSON. Além disso, o CouchDB oferece uma API construída sob a arquitetura RESTful (*Representational State Transfer*) para acesso aos dados por meio do protocolo HTTP. Esta API é o principal método de interface com o banco de dados, oferecendo um canal para processar solicitações HTTP, armazenando executando e manipulando informações no banco de dados.[**?**]

Com isso, o CouchDB se encaixa em ambientes de vários níveis de tamanho e complexidade com facilidade. Projetado para ambiente Web, com a possibilidade de ocorrência de inúmeras falhas este banco de dados abstrai as tarefas mais complexas de gerenciamento. Dessa forma, a estrutura do CouchDB torna-se uma boa opção quando a robustez, a consistência e a semântica transacional é requerida [24] [**?**].

2.0.1.6 Cassandra

Cassandra é um banco de dados *OpenSource* criado para ser utilizado pela empresa Facebook em 2008 com o objetivo de obter características ser distribuído, descentralizado, altamente escalável, altamente disponível, tolerante a falhas e consistente. Seu *design* engloba características presentes no banco de dados DynamoDB criado pela empresa Amazon e o conceito de BigTable [25].

A estrutura descentralizada do Cassandra é constituída por uma arquitetura *peer-to-peer* onde os nós não oferecem distinção de funções ou operações. Dessa forma, não há nenhum ponto único de falha visto que todos os nós possuem as mesmas capacidades. Além disso, esta estrutura é uma das chaves para atender a alta disponibilidade deste banco de dados, visto que isso evita interrupções e oferece maior facilidade na operação. Outra característica ofertada pela estrutura replicada é o melhor desempenho local e o menor tempo de inatividade [25] [26].

O banco de dados Cassandra possuí arquitetura baseada em linhas (*row's*), as quais são referenciadas por um índice. Cada linha é composta por colunas (*column's*), as quais mantém o valor

dos dados armazenados, e podem ser acessados por meio da linguagem CQL (*Cassandra Query Language*), a qual foi baseada na linguagem SQL para facilidade de uso, visto que no ponto de vista desta linguagem, os dados são estruturados em tabelas [26]. A Figura 3 apresenta a arquitetura do modelo de dados utilizado pelo Cassandra.

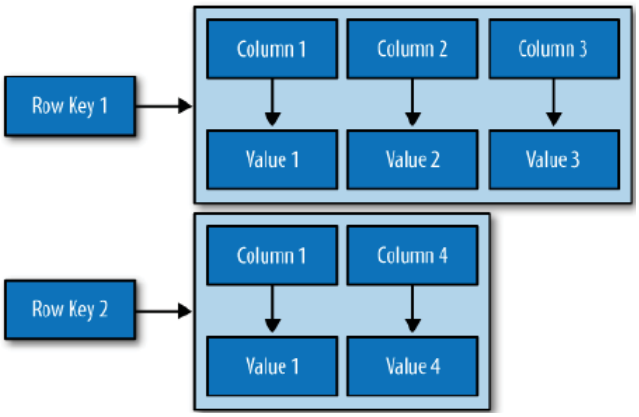

Figura 3. Modelo de dados Cassandra [25]

Todavia, são consideradas algumas limitações impostas pela arquitetura do banco de dados Cassandra. São consideradas: limitação da linguagem CQL para subconsultas e agregação, limitação do tamanho de coluna a 2GB, número máximo de células (linhas x colunas) é de no máximo 2 bilhões em uma tabela, e os dados devem ser compatíveis com o tamanho máximo permitido ao disco em uma mesma máquina, visto que suas restrições de replicação, impedem armazenamentos maiores [26].

2.0.2 Sistemas de Informação Georreferenciados - SIG

Os Sistemas de Informação Georreferenciados (SIG), do Inglês *Geographic Information System (GIS)* pode ser conceituado como um sistema de informação computadorizado que permite a captura, modelagem, armazenamento, manipulação e apresentação de dados georeferenciados [27]. Segundo Miranda [28] este tipo de sistema surgiu com a necessidade do processamento de dados cartográficos, e são comumente utilizados para manipular informações espaciais, dado estes geralmente em formato de mapas.

Com isso, SIG's são utilizados para análise e manipulação de dados geográficos de forma automatizada. Estes sistemas são caracterizados por realizar análises complexas de dados, relacionando valores numéricos com dados geográficos, tendo a capacidade de analisar e manipular diferentes tipos de dados, como dados espaciais, cartográficos e modelos numérico de terrenos e dados tradicionais, como valores alfanuméricos e numéricos [29]. Com isso, estes sistemas podem ser aplicados em diversas situações, como por exemplo no controle de tráfego, levantamento demográfico, sistemas de cartografia, monitoramento de áreas, entre outras [30] [31].

Segundo Queiroz Worboys [32] e Worboys *et al.* [27] as estruturas de bancos de dados geográficos podem ser divididas em dois segmentos: estruturas vetoriais e estruturas matriciais. A

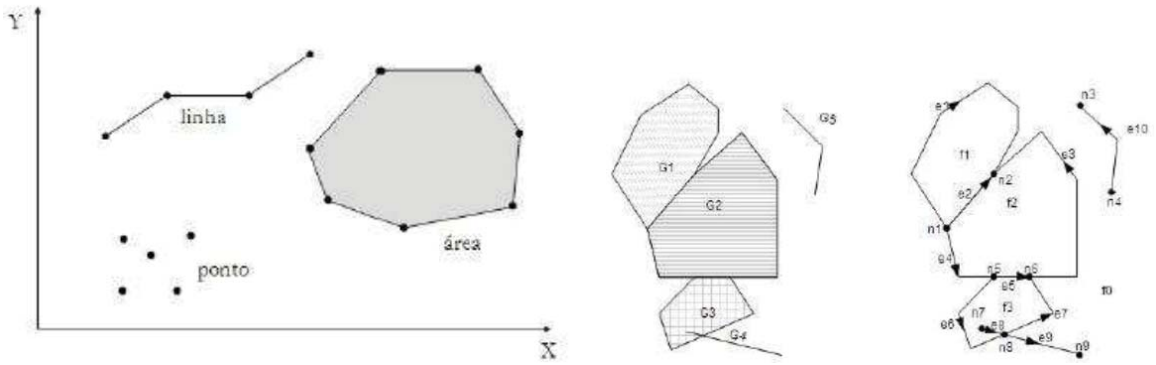

Figura 4. Representação vetorial dos elementos geográficos [32]

primeira geralmente é utilizada para identificar coordenadas de fronteiras de elementos geográficos. Uma coordenada de um vetor é constituída por um par ordenado (x e y) os quais são responsáveis por indicar as localizações dos elementos no espaço. Cada elemento de uma coordenada geográfica pode ser identificado por pontos, linhas ou polígonos, os quais são apresentados nas Figura 1.

Para as estruturas matriciais (também conhecidas como raster), são utilizadas representações em formato de grade. Dessa forma, cada célula da matriz é constituído por um atributo para o qual representa parte do objeto completo que se está representando, a qual pode ser acessada por meio de suas coordenadas. Este tipo de representação parte do princípio que o espaço que se está representando pode ser tratado como uma superfície plana, e cada célula da martriz está associada a uma parte do elemento [32]. Na Figura xx é mostrado a representação matricial de uma área georreferenciada.

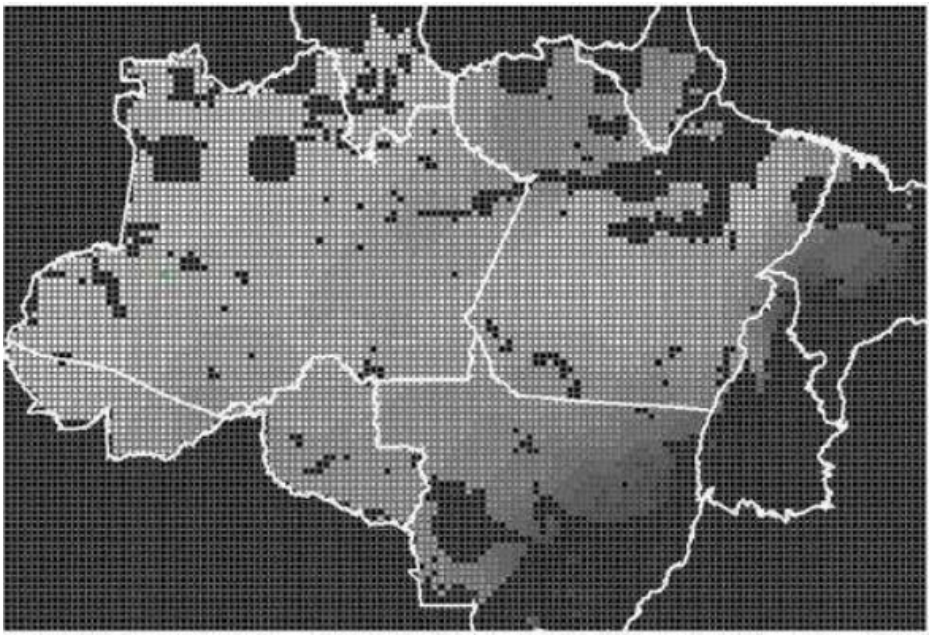

Figura 5. Representação matricial dos elementos geográficos [32]

2.0.2.1 Bancos de Dados Geográficos - BDG

Os sistemas de Banco de Dados Georreferenciados (BDG's) são classificados como banco de dados não-convencionais pois possibilita o armazenamento de dados espaciais ou geográficos, juntamente com seus respectivos relacionamentos e valores, além de armazenar dados comuns como a maioria dos bancos de dados convencionais [29] [31].

A utilização de bancos de dados designados para trabalhar com Sistemas de Informação Georreferenciados vem sendo cada vez mais difundido nas áreas de cartografia e afins. Os dados geográficos ou geoespaciais são caracterizados por possuir recursos próprios no que se refere o modo de armazenamento computacional das informações [33]. Câmara [34] cita ainda que bancos de dados geográficos possuem informações de atributos não descritivos, nos quais os dados espaciais são expressados por meio de coordenadas geográficas (latitude e longitude), e atributos descritivos, os quais expressam atributos referentes a geografia e que podem ser representados por um banco de dados convencional. Dessa forma, os dados espaciais podem ser modelados para representar geometrias como linhas, pontos ou polígonos utilizando o sistema de coordenadas geográficas [35].

Em virtude da exploração dos recursos dos bancos de dados georreferenciados, têm-se requerido cada vez mais necessidades para estes sistemas. Uma vez que as tecnologias buscam oferecer análises e informações de maneira mais acelerada, a exigência por características como eficiência, precisão e velocidade nos resultados aumenta [33]. Com isso, desenvolveu-se bancos de dados e extensões para bancos de dados capazes de lidar com informações georreferenciadas de maneira eficiente. Dentro destes, destacam-se extensões como PostGIS e GeoCouch, cujas são extensões para os SGBD's Postgresql e CouchDB, respectivamente.

2.0.2.2 PostGIS

Uma das características do SGBD PostgreSQL é a extensibilidade, ou seja, a possibilidade de adicionar características capazes de operar com tipos de dados específicos. Com isso, uma extensão de código fonte aberto para trabalhar com valores geográficos foi desenvolvida pela empresa canadense *Refractions Research Inc.* [32] [35]. Esta extensão, chamada PostGIS incluí todas as funcionalidades e objetos padrão OpenGIS (*Open Geospartial Consortium*), que é um consórcio internacional que nasceu da iniciativa de garantir a interoperabilidade de softwares por meio da padronização das funções que tratam dos dados georreferenciados no banco de dados. Com isso, permite-se que diferentes softwares que utilizam informações espaciais, consigam trabalhar de forma uniforme com estas informações [36].

Para efetuar cálculos geoespaciais como área, comprimento, distância, largura e perímetros, o PostGIS utiliza uma biblioteca de software chamada GEOS4, a qual opera sobre a norma SFS (*Simple Feature Access*) para disponibilizar funções capaz de tratar dados geográficos [37]. O PostGIS possuí diferentes tipos de dados, nos quais consegue operar, são estes [37] [38] [39]: *Point LineString Polygon Multipoint MultiLineString MultiPoligon GeometryCollection*

A Figura 9 apresenta a área de salinas do estado de Utah nos Estados Unidos. A Rota 80 ("*Route 80*") representada por uma *linestring*, corta o polígono pelo qual representa o estado de Utah. Além disso, são apresentadas outra representações geométricas, como *point* e a inserção de outros elementos dentro do elemento principal do tipo *polygon* a qual pode também ser representada por uma estrutura *multipolygon*, visto que existem outros dois polígonos inseridos neste (*Bonneville State Park* e *Hill Air Force Base*) [40].

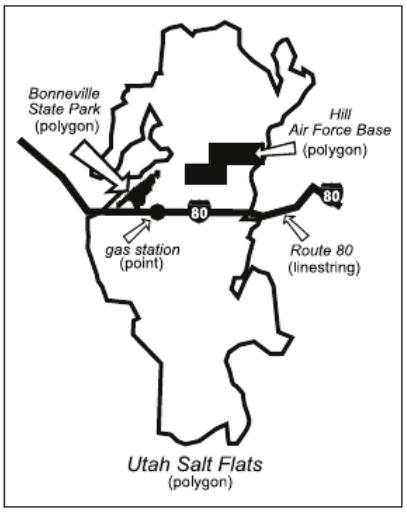

Figura 6. Modelo de *linestring*, *point* e *polygon* do PostGIS [40]

2.0.2.3 GeoCouch

GeoCouch é uma extensão do banco de dados CouchDB para o armazenamento e tratamento de dados georreferenciados. Dessa forma, o GeoCouch oferece uma maneira de construir índices espaciais espaciais adicionados no modelo do CouchDB [41].

GeoCouch apenas permite a manipulação de dados por meio da geometria GeoJSON, o qual é um padrão de formato para dados geoespaciais baseado no padrão JSON. Para dados espaciais definidos em outros formatos, como por exemplo *shapefile*, são definidas ferramentas capazes de realizar a conversão de formatos. Dentre estas ferramentas está a shp2geocouch, que permite a conversão de dados *shapefiles* em GeoJSON [42]. A Figura 7 apresenta um exemplo de dados GeoJSON.

Os dados podem ser representados por *points, linestrings, polygons, multipoints, multilinestrings* e *multipolygons*. Outra característica imposta pelo GeoCouch é a restrição ao acesso à funções geográficas, como *equals, intersect, disjoint, intersection, union* e *difference*. Dessa forma, é necessário o tratamento destas operações de forma externa ao banco de dados [42].

```
{
    "update_seq": 508,
    "rows": [
        {
            "id": "5807724052",
            "bbox": [
                -157.964782,
                21.4836,
                -157.964782,
                21.4836
            ],
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -157.964782,
                    21.4836
                ]
            },
            "value": {
                "image":
                "http://farm3.static.flickr.com/2646/5807724052_9f7b947da9_s.jpg"
            }
        }
        //...MORE ROWS...
    ]
}
```


Figura 7. Exemplo de dados espaciais no GeoCouch [41]

2.0.3 *Model, View, Controller* - MVC

O padrão MVC visa separar o fluxo de lógica da aplicação da interface com o usuário. Este padrão surgiu com o a linguagem Smaltalk-76, e incorpora três principais camadas: Modelo (*Model*), Visão (*View*) e Controle (*Controller*) [43]. Sendo assim, os itens abaixo descrevem as camadas.

- Modelo: Responsável por representar o estado e o comportamento dos objetos do sistema, sendo responsável por gerenciar o comportamento dos objetos.
- Visão: Responsável por apresentar os dados do sistema. Em aplicações *web*, geralmente este modelo podem prever conteúdos HTML, imagens, entre outros.
- Controle: Responsável por atender as requisições do usuário e definir o comportamento do sistema como um todo. Além disso, é nesta camada onde são feitas as interações com a camada modelo, provendo o interfaceamento do usuário com os dados do sistema.

2.0.4 Arquitetura Orientada a Serviços - SOA

A Arquitetura Orientada a Serviços, ou do Inglês *Service-Oriented Architecture* (SOA) é uma estrutura que emprega as necessidades requeridas por uma organização cujos processos podem ser decompostos em forma de serviços. Estes serviços por sua vez são construídos por tecnologias interoperáveis [44].

Murakami [5] afirma que SOA representa um conceito arquitetural abstrato, permitindo a construção de sistemas baseados em componentes com baixo acoplamento, o que permite a interoperabilidade entre sistemas. De forma geral, a arquitetura SOA é uma forma de concepção de um sistema de software para prover serviços para usuários finais ou outras ferramentas por meio de interfaces localizáveis, de forma a permitir que sistemas distintos possam operar de forma transparente [45].

Dentre as tecnologias que aplicam o conceito de SOA e são comumente utilizadas destacam-se os serviços web (*Web Services*). Estes oferecem um método padronizado para integrar aplicações web, de forma com que as partes não necessitam ter conhecimento extensivo das características do sistema com que se está conectado [44]. Murakami [5] acrescenta ainda que *web services* são neutros da tecnologia de transporte, visto que além dos protocolos conhecidos, tais como HTTP (*Hypertext Transfer Protocol*) e HTTPS (*Hyper Text Transfer Protocol Secure*), é possível utilizar protocolos proprietários.

A utilização de *Web Services* na arquitetura SOA consiste na definição de serviços autônomos, que são identificados por URL's com interfaces com processamento de mensagens [46]. Essas mensagens podem ser identificadas por exemplo, pelos padrões XML (*eXtensible Markup Language*) e JSON (*JavaScript Object Notation*). O padrão XML é geralmente utilizado pelo protocolo SOAP (*Simple Object Access Protocol*), o qual oferece características que são utilizadas para troca de informações em plataformas distribuídas ou entre diferentes sistemas. Como alternativa mais simples e leve ao protocolo SOAP para integrar dispositivos pode ser utilizado o estilo arquitetural RESTful (*Representational State Transfer*), o qual utiliza o padrão JSON para a troca de dados e ganhou grande aceitação na Web [47] [48].

2.0.5 NodeJS

Devido aos limites de escalabilidade que podem ser impostos por modelos de programação baseados em *threads*, surgiu a plataforma NodeJS, a qual opta por um modelo baseado em eventos para gerenciar a concorrência entre processos [49]. A arquitetura baseada em eventos pela qual o NodeJS é caracterizado oferece um menor custo computacional, como redução do consumo de memória, altas taxas de transferência (mesmo sob altas demandas) e alto número de conexões simultâneas [50] [51].

NodeJS utiliza a linguagem JavaScript no ambiente servidor. Com isso, o programa é executado em uma única *thread* (*single-thread*) fazendo uso de recursos I/O (*Input/Output*) de forma não bloqueante. Para isso, são utilizadas funções chamadas *callback's* que registram um trecho de código que será executado quando um determinado evento ocorrer. Esta característica oferece maior velocidade na execução das tarefas, visto que o modelo não bloqueante caracteriza-se por dar continuidade ao fluxo do programa, ao mesmo tempo em que são acessados recursos de I/O [49].

Com isso, a plataforma NodeJS oferece menores custos de desenvolvimento e maior eficiência computacional, visto que é capaz de atender um número maior de solicitações sem perdas consideráveis no desempenho. Além disso, a linguagem JavaScript oferece uma representação na-

tiva ao padrão JSON, o qual pode ser utilizado para o fornecimento dos dados por meio de serviços Web. Com isso, a plataforma NodeJS apresenta-se em uma fase de crescimento, servindo como uma alternativa atraente às tecnologias tradicionais como PHP e Java, para construção de aplicações Web [50] [51] [52].

2.0.5.1 Mongoose

O Mongoose é uma biblioteca Javascript para NodeJS que é utilizado para modelagem de objetos para o banco de dados MongoDB. O mongoose oferece uma solução baseada em *Schemas*, a qual é utilizada para modelar os dados da aplicação. Nesta biblioteca, estão incluídas as estruturas de tipagem, validação e lógica de negócios da aplicação [53].

O Mongoose funciona essencialmente como um ORM *Object Relational Mapping*, oferecendo uma interface de comandos MongoDB para manipulação de objetos em NodeJS com o MongoDB [54].

2.0.5.2 PG

PG é um pacote Javascript para NodeJS, o qual oferece uma interface para conexões com o SG BD PostgreSQL [55]. A biblioteca PG oferece recursos para ligações não-bloqueante para NodeJS. Isto significa que o PG mantém a estrutura de operações assíncronas, executadas pelo NodeJS [56].

A cada execução de uma *query*, são definidos dois eventos padrão (*row* e *end*). O evento *row* é disparado sempre que um valor é recebido do PostgreSQL. Com isso, todas as linhas retornadas pelo banco de dados são recebidas pelo evento *row*, até o final da execução da *query*, na qual o evento *end* é chamado [57]. A Figura 8 exibe um exemplo de conexão e consulta ao banco de dados utilizando a biblioteca PG.

```
1
2  var pg = require("pg");
3
4  var conString = "pg://user:pass@localhost:5432/Users"; // conexão com o banco de dados
5  var client = new pg.Client(conString);
6
7  var query = client.query("SELECT * FROM users;"); // submete uma query ao banco de dados
8
9  query.on("row", function (row, result) {
10     result.addRow(row); // recebe um valor (linha) do banco de dados
11 });
12
13 query.on("end", function (result) {
14     client.end(); // finaliza a conexão com o PostgreSQL
15 });
```

Figura 8. Pacote PG NodeJS

2.0.5.3 Express JS

O Express JS é um *framework* Javascript para NodeJS de estrutura relativamente pequena, utilizado para criação de API's (*Application Programming Interface*) e adicionar novos recursos. Este

framework facilita a organização das funcionalidades do aplicativo com um *middleware* e roteamento para abstração do protocolo HTTP [58] [59].

A principal característica do Express, é fornecer uma camada mínima entre a aplicação e o servidor. Essa característica mantém o *framework* flexível e extensível, oferecendo ao desenvolvedor de sistemas a flexibilidade de adicionar funcionalidades e recursos para atender cada sistema em específico. Portanto, o Express JS é um *framework* que pode ser utilizado desde projetos menores, até projetos que requeiram um nível de complexidade maior [60].

2.0.6 REST

REST (*Representational State Transfefer* ou em português Transferência de Estado Representacional) é um estilo arquitetural de software o qual define um padrão na forma de como os dados são representados, acessados e modificados na *Web*. O REST consiste de um conjunto de regras básicas, as quais são aplicadas na camada de software. Isto permite criar uma abstração dos detalhes de implementação dos componentes do protocolo *web*, priorizando a interação e manipulação dos dados [61] [62].

Para acesso a camada de dados, o REST disponibiliza seus recursos (elemento), por meio de URIs (*Uniform Resource Identifiers)*, os quais são normalmente links na *web*, constituindo fundamentalmente uma arquitetura cliente-servidor, a qual é projetada para utilizar como meio de comunicação, o protocolo HTTP [61]. Para tanto, os meios de acesso aos dados são definidos por quatro métodos básicos: [62]

- GET: O método GET é utilizado para recuperar a representação de um determinado recurso. Dessa forma, esse tipo de solicitação é, por definição, utilizado apenas para leitura, e não podem modificar o valor ou estado do recurso solicitado;
- PUT: O método PUT é utilizado para transferir um novo estado para um determinado elemento. Tal elemento, é identificado pela URI e este não existindo, ele poderá ser inicializado com o valor fornecido, do contrário, o elemento será atualizado;
- POST: Uma solicitação do tipo POST pode ser interpretada como uma solicitação para criar um novo recurso, o qual deve ser identificado com uma URI. Este método se induz uma redundância com o método PUT, no entanto, o POST permite que a aplicação decida coo identificar o valor recém-criado, enquanto no método PUT, o cliente é que fica responsável por escolher um identificador único para um novo registro;
- DELETE: Solicitações do tipo DELETE são utilizadas para excluir um elemento existente. Após a exclusão do elemento, a URI é invalidada, e o servidor não será mais capaz de responder com êxito as solicitações para a URI excluída.

Utilizando-se dos métodos citados acima, os serviços *web* REST são projetados para serem mais leves, acessados via URI's e utilizam o protocolo HTTP para o tráfego dos dados. Dessa forma,

este padrão se torna um estilo arquitetural independente de linguagem, onde clientes e servidores interagem com um conjunto limitado de operações entre si, reduzindo a complexidade da implementação do padrão de comunicação [61] [62].

2.1 ANGULARJS

AngularJS é um framework JavaScript open-source, mantido pelo Google, que auxilia na execução de single-page applications. Seu objetivo é aumentar aplicativos que podem ser acessados por um navegador web, foi construído sob o padrão model-view-view-model (MVVM), em um esforço para facilitar tanto o desenvolvimento quanto o teste dos aplicativos.

A biblioteca lê o HTML que contém tags especiais e então executa a diretiva na qual esta tag pertence, e faz a ligação entre a apresentação e seu modelo, representado por variáveis JavaScript comuns. O valor dessas variáveis JavaScript podem ser setadas manualmente, ou via um recurso JSON estático ou dinâmico.

O AngularJS é construído sob a crença de que a programação declarativa é melhor do que a programação imperativa quando se trata da construção de interfaces com o usuário e da conexão de componentes software, enquanto a programação imperativa é excelente para a escrita de regras de negócio. [1] O framework adapta e estende o HTML tradicional para uma melhor experiência com conteúdo dinâmico, com a ligação direta e bidirecional dos dados (two-way data-binding) que permite sincronização automática de models e views. Como resultado, AngularJS abstrai a manipulação do DOM e melhora os testes.

3. AGRONET SERVICES: UMA PLATAFORMA DE ARMAZENAMENTO E DISPONIBILIZAÇÃO DE DADOS DA AGRICULTURA DE PRECISÃO

3.1 RESUMO

Ao longo dos anos a área agrícola se destaca pelo aumento da sua produtividade, que se relaciona diretamente com a expansão da Agricultura de Precisão, cujos recursos auxiliam na coleta de dados, servindo de matéria-prima no processo de tomada de decisões. No entanto, os dados são coletados de diferentes sensores equipamentos eletrônicos agrícolas, os quais produzem grandes volumes de dados de diferentes tipos e formatos. Assim, o objetivo deste trabalho é apresentar uma plataforma para armazenamento e disponibilização dos dados da AP. Como resultados, obteve-se uma ferramenta capaz de armazenar grandes volumes de dados de tipos heterogeneos, os quais são recebidos e disponibilizados por meio de uma API REST.

3.2 INTRODUÇÃO

Ao longo dos anos a área agrícola tem se destacado pelo aumento de sua produtividade. Isto está intimamente relacionado à expansão do conceito de Internet das Coisas (*Internet of Things* - IOT), que se aplica também ao cenário da Agricultura de Precisão (AP), com o desenvolvimento de novos produtos, maquinários e equipamentos que melhor se adaptam às condições ambientais e aos novos processos que são criados para o manejo do campo. Por conta destes recursos, dados sobre meteorologia, maquinários, aplicação de insumos, entre outros, formam uma grande massa de dados, a qual pode ser utilizada para a geração de informações fundamentais para o bom manejo das culturas e seu respectivo sucesso [5].

Para Sorensen [1], o grande desafio do setor da AP se refere à forma de armazenamento e a transformação da grande quantidade de dados que são gerados a cada instante de tempo e local no espaço (dados geotemporais) pelos equipamentos de AP, em conhecimento útil e aplicável no processo de tomada de decisões. Este desafio se justifica pela existência de grandes volumes de dados aliada a heterogeneidade de seus tipos, onde por exemplo, dados meteorológicos, dados de plantio e colheita, podem ser gerados. Portanto, devido a vasta tipologia de dados presente no setor de AP, a coleta, o armazenamento e a disponibilização destes dados pode se tornar dificultosa [5].

Dessa forma o uso de ferramentas de apoio, criam condições para a extração dos mais variados tipos de informações gerados pelos equipamentos de AP. Para tanto, se fazem necessárias tecnologias de alto desempenho para a recepção, processamento e disponibilização dos dados coletados e meios de armazenamento de dados capazes de concentrar um grande volume de informações, incluindo o suporte à tipos de dados complexos e heterogêneos. Em resolução disto, soluções computacionais podem ser utilizadas para realizar o processo de gestão e manipulação destes dados, tra-

zendo consigo inúmeras vantagens que possuem influência direta na produtividade do campo. Sendo assim, este tipo de ferramenta oferece oportunidade aos profissionais da área agrícola, de mudar a distribuição e o tempo de aplicação de produtos químicos, utilizando informações como a variabilidade espacial e temporal do campo [4].

Neste contexto, Zhang, Wang e Wang [4] e LI et al. [6] afirmam que uma das barreiras que precisam ser superadas na área da AP, é o armazenamento de forma que possibilite o tratamento e análise de grandes massas de dados. Uma das principais dificuldades que são acarretadas pelo volume de dados se refere ao processo de armazenamento e interpretação dos dados, onde o desejo de responder a essa variabilidade em uma escala tornou-se a principal meta. Outro desafio computacional comumente encontrado neste âmbito, é a capacidade de atender de forma eficaz o alto número de conexões geradas pelos equipamentos de AP e usuários que acessam os dados. Com a geração de grandes tráfegos de dados aliado a necessidade de alta escalabilidade na aplicação, torna-se necessário para prover conectividade e o armazenamento e disponibilização de dados provenientes da AP.

Dessa forma, visto a grande diversidade de dados provenientes da AP, percebe-se a importância de um estudo e definição de metodologias para tratamento, armazenamento e disponibilização de tais dados. Para tanto, este capítulo tem por objetivo apresentar uma plataforma de armazenamento e disponibilização de dados providos pela AP. A seção 3.3 apresenta os recursos utilizados para a implementação do sistema. Na seção 3.4 apresenta a arquitetura, o desenvolvimento e os resultados da implementação da plataforma. Por fim, as seções 3.5 e 3.6 apresenta as discussões e conclusões do trabalho.

3.3 MATERIAL E MÉTODOS

Para atingir os objetivos propostos, o desenvolvimento desse trabalho visou também a escolha das ferramentas mais adequadas para a construção da infraestrutura de armazenamento, tratamento e disponibilização de dados de Agricultura de Precisão. Dessa forma, o desenvolvimento do sistema foi elaborado obedecendo os critérios do padrão MVC (*Model-View-Controller*), o qual possibilita o desenvolvimento do sistema de forma modular [63] [64].

Dessa forma, visando obter o melhor gerenciamento e a melhor separação do código-fonte, o sistema foi desenvolvido utilizando o padrão MVC, conforme é apresentado na figura 9

3.3.1 Armazenamento de Dados

A metodologia de armazenamento dos dados tem papel fundamental ao cenário do qual este trabalho está inserido. Sendo assim, a construção de uma modelagem robusta e eficiente oferece flexibilidade e agilidade no desenvolvimento de aplicações que dependam de tais dados. Para tanto, são impostos alguns desafios, visto a grande quantidade de informações de tipos heterogêneos que são armazenadas neste ambiente. Dados provenientes de coletores meteorológicos, podem ser gera-

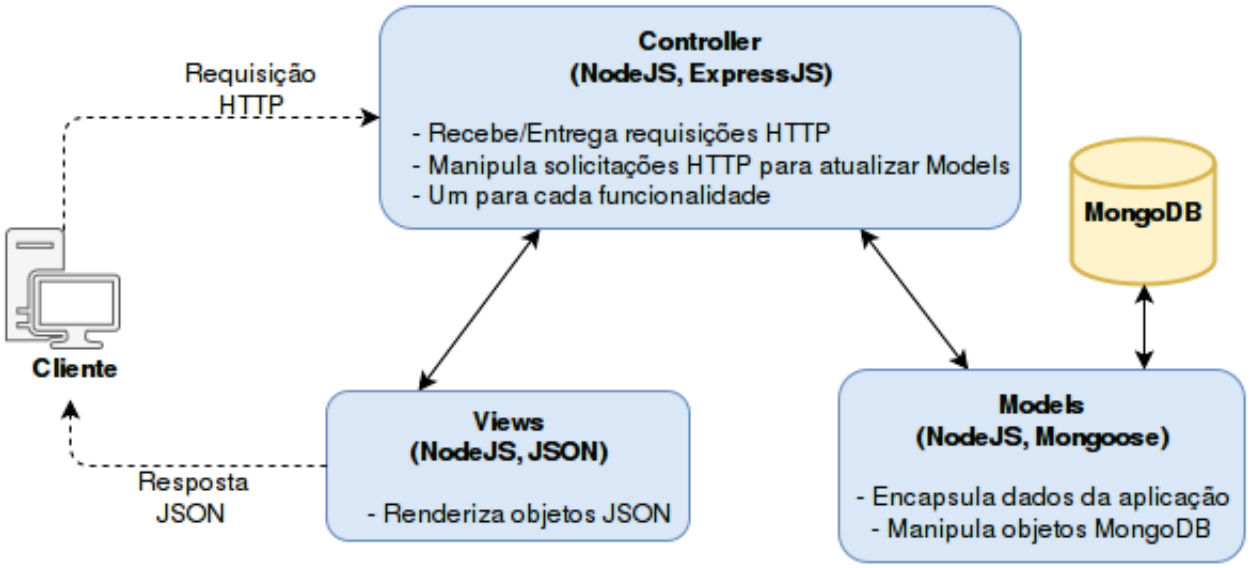

Figura 9. Arquitetura MVC - AgroNET Services

dos de forma horária, por exemplo, totalizando um grande volume de informações ao longo do tempo. Além disso dados gerados por equipamentos de plantio, como: número de sementes e quantidade de fertilizantes, ou por equipamentos de colheita, como: dados de produtividade e umidade da produção, podem ser produzidos em curtos instantes de tempo, visto que geralmente estes dados são associados a uma determinada localização espacial (provida por aparelhos de GPS - *Global Positioning System*) e a um determinado tempo, gerando portanto informações espaciais e temporais.

Dessa forma, o armazenamento destes dados propicia a extração de informações, fornecimento de dados para, modelos de simulação, modelos de prognósticos, entre outras ferramentas de processamento de dados, com vistas a oferecer informações consistentes ao usuário. Este resultado, geralmente, pode ser aplicado ao processo de tomada de decisões, que na agricultura, possuí importância significativa ao manejo e a geração de alertas e riscos no ambiente do campo. Para tanto, com base nos dados coletados, é possível gerar e armazenar resultados como mapas de cultivo, mapas de produtividade, informações de riscos (doenças, pragas, ...) e informações de manejo.

Por conta dessas características, a utilização do Modelo Orientado a Documentos, com o Banco de Dados MongoDB, caracteriza-se como a principal escolha. endo visto a alta variabilidade de dados no cenário ao qual este projeto está inserido, verifica-se que as características de possibilitar a criação de documentos de forma não estruturada, aliada a alta performance de manipulação dos dados, o MongoDB é capaz de suprir as necessidades de armazenamento de informações que que não seguem um padrão estrutura, conforme nos modelos tradicionais de bancos de dados [22]. Por conta disso, soluções NoSQL (*Not Only SQL*) tem sido cada vez mais utilizadas em soluções que exigem alta concorrência e eficiência no armazenamento de grandes volumes de dados [65].

3.3.2 Arquitetura Orientada a Serviços Web

É de suma importância para o desenvolvimento de um sistema, além de um banco de dados robusto e eficiente, o uso de ferramentas adequadas que possibilitem a interação com o banco de dados de forma ágil, permitindo os mais diferentes tipos de consultas e forma de manejo dos dados. [65].

Para tanto, para o desenvolvimento deste projeto optou-se pela utilização de tecnologias baseadas em Serviços Web (*Web Services*), obedecendo os conceitos da arquitetura SOA (*Service-Oriented Architecture*) no desenvolvimento da aplicação [1]. No que se refere à troca de mensagens, o uso do estilo arquitetural REST foi adotado tanto para o recebimento de informações de equipamentos e dispositivos agrícolas e de estações meteorológicas, bem como na disponibilização dos dados armazenados.

Como tecnologia de implementação e integração de tais recursos, o NodeJS foi utilizado, juntamente com o *framework* ExpressJS, permitindo o desenvolvimento modular da aplicação, gerando como resultados uma API (*Application Programming Interface*) capaz de receber, tratar, armazenar e disponibilizar os dados provenientes da AP.

3.4 RESULTADOS

Tomando como base para este trabalho a possibilidade de integração entre diversos tipos de sistemas e dispositivos agrícolas, o modelo Cliente x Servidor aplicado a uma infraestrutura baseada em serviços web permite a fácil expansão e/ou conexão com outros sistemas, como por exemplo portais agrícolas, relatórios agrícolas, equipamentos agrícolas, entre outros. Utilizando-se deste propósito, a Figura 10 apresenta um diagrama geral do sistema.

Para a conexão da plataforma com os sensores e dispositivos agrícolas, o padrão arquitetural REST foi utilizado, e obedecendo os métodos disponibilizados pelo padrão: GET, POST, PUT e DELETE, é realizada a interação e a manutenção dos dados trocados pelos dispositivos agrícolas e pela plataforma. Desta mesma forma, Visando padronizar os protocolos de comunicação da plataforma, o acesso aos dados armazenados nesta plataforma, também é dado via REST. As sessões abaixo, descrevem as operações de troca de dados no sistema.

3.4.1 Armazenamento dos dados

Por se tratar de um banco de dados não-relacional, alguns critérios diferentes da representação relacional, devem ser adotados para a modelagem dos dados no MongoDB, visto que a forma de armazenar os dados, podem influenciar diretamente no desempenho das operações realizadas no banco de dados [22].

[1] Coleção de serviços utilizados para troca de dados entre dois ou mais interlocutores [66]

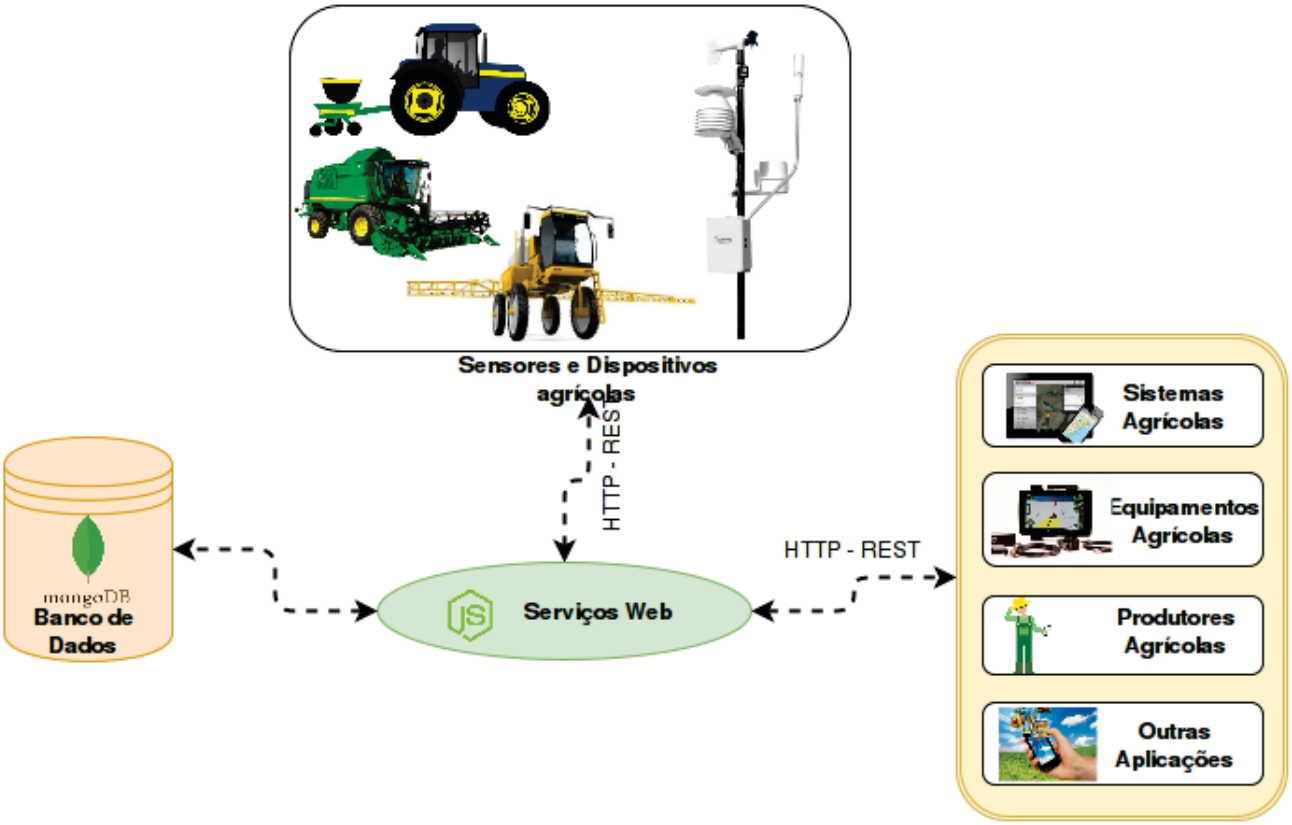

Figura 10. Diagrama geral da plataforma AgroNET Services

Dependendo das necessidades de manipulação dos dados, o MongoDB prima pelo agrupamento das informações em documentos, isto é, sempre que possível os dados devem ser salvos de forma conjunta, evitando esquemas e relacionamentos. Contudo, isto nem sempre pode ser a melhor opção, pois os documentos do MongoDB possuem uma limitação de até 16 MB de tamanho, impossibilitando o armazenamento de dados que são propensos a aumentarem seus volumes. Para tanto, nestes casos, o uso de referências entre documentos, pode ser utilizado para indicar os relacionamentos entre os dados.

Sabendo disso, a modelagem do banco de dados foi construída visando a fácil integração entre diferentes tipos de informações utilizados no cenário da Agricultura de Precisão. Dessa forma, dados de fazendas e seus talhões [2] e suas informações de manejo, tais como aplicações de produtos químicos, ocorrências de patologias, maquinários e implementos agrícolas, bem como os dados que são coletados por estes. Além disso, informações de usuários e seus tipos (produtores agrícolas, operadores de maquinário, entre outros), também podem ser armazenados.

Maquinários e implementos agrícolas, tais como, tratores, semeadoras, pulverizadores e colheitadeiras, serão responsáveis por coletar e transmitir dados para a plataforma. Portanto, dados como número de sementes plantadas por metro, quantidade de fertilizante aplicado por hectare, largura de plantio, quantidade de hectares trabalhados, quantidade de sacas colhidas por hectare, índice de umidade de sementes/fertilizantes, taxa de aplicação de químicos e falhas da máquina são exem-

[2] Representam a divisão real ou imaginária de uma propriedade rural, o que permite um controle mais mais detalhado do manejo da área, de forma individualizada [67].

plos de dados que serão recebidos e armazenados pela plataforma. Além disso, fazem parte deste grupo, dados meteorológicos, como temperatura, precipitação, radiação solar e umidade relativa do ar.

Tomando como base estas informações, e visando a melhor organização, a estrutura do banco de dados, pode ser separada em quatro formas, como são descritas nos enumeradores abaixo:

1. **Dispositivos e dados de dispositivos**: Estes dados, apresentam características únicas em seu armazenamento. Dessa forma, neste caso são criadas duas entidades distintas, que permitem a separação das informações de dispositivos agrícolas, que pode ser qualquer qualquer dispositivo eletrônico que transmite dados para a plataforma, dos dados que são transmitidos por estes. Para tanto, é criado uma referência entre os dos esquemas, cuja relação determina de qual dispositivo determinado dado pertence.

   Visando a melhoria na performance de consultas, os dados transmitidos foram agrupados em formato diário, ou seja, a cada dia é criado um novo documento que armazena todos os registros coletados pelo dispositivo no dia determinado. Isto foi determinado, haja visto o grande volume de dados que pode ser armazenado, esta organização permite obter uma série grande de dados, com um número menor de consultas ao banco de dados, o que reflete em performance na disponibilização do dado [54];

2. **Implementos e maquinários agrícolas:** Estes dados, basicamente são constituídos pelas informações da frota de maquinários agrícolas que são utilizados para o manejo das áreas da fazenda. Geralmente um ou mais dispositivos podem estar associados a estes maquinários. Um exemplo disso, é a instalação de um terminal GPS em um trator, onde o terminal GPS pode ser um dispositivo do sistema, e o trator é o maquinário agrícola cujo dispositivo está vinculado;

3. **Fazendas e Talhões:** As informações referentes a propriedade agrícola, foram agrupadas em documentos, evitando a criação de novos esquemas e relacionamentos. Dessa forma, ao criar um determinado documento, neste podem ser inseridas todas as informações referentes a propriedade agrícola (fazenda) e suas áreas (talhões);

4. **Safras, Aplicações de químicos, ocorrência de patologias:** Este grupo de dados, considera o armazenamento das informações da safra e seu total acompanhamento de surgimento de patologias, bem como informações sobre aplicações de produtos químicos na lavoura. Isso permite registrar informações importantes para o manejo da cultura, bem como armazenar históricos de produtividade separadas por talhão.

3.4.2 Estrutura da aplicação Web

Utilizando do estilo arquitetural MVC, o software foi desenvolvido utilizando a linguagem JavaScript por meio do NodeJS com a utilização do *framework* ExpressJS, criou-se uma arquitetura para

a aplicação, visando a fácil manutenção do sistema e também a sua fácil expansão. Sendo assim, a Figura 11 apresenta a estrutura criada para a aplicação.

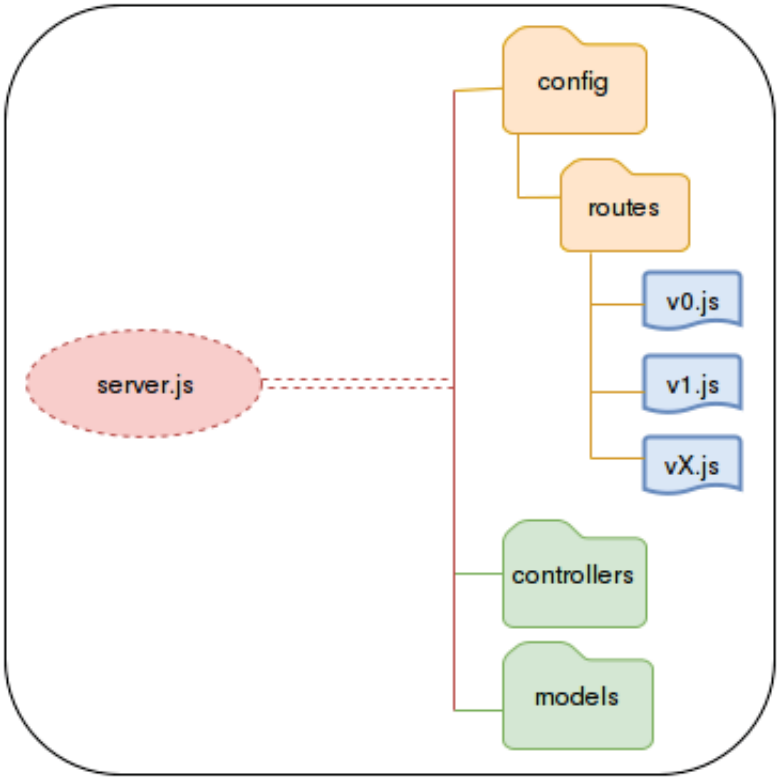

Figura 11. Organização da estrutura de diretórios do projeto

No diretório "controllers", são inseridos todos os arquivos que contém os métodos responsáveis pelo tratamento das requisições da API. Para tanto, são criados arquivos específicos para cada esquema do banco de dados, contendo todos os métodos CRUD (*Create, Read, Update e Delete*), cujos são responsáveis por mapear os dados armazenados no MongoDB em objetos JavaScript.

Por outro lado, a pasta "models"é utilizada para armazenar todos os esquemas do banco de dados. Para cada novo esquema, é criado um novo arquivo com as especificações de atributos e métodos que definem como a entidade será caracterizada. Para realizar o mapeamento entre os esquemas manipulados e o banco de dados MongoDB, utilizou-se do pacote *mongoose*, o qual permite a manipulação dos objetos salvos no banco de dados MongoDB, utilizando um conjunto funções pré-determinadas, que se assemelham a linguagem do MongoDB [**?**]. Para exemplificar este processo, a Figura 12 apresenta o esquema de "Produtos"("Products"), o qual possuí dois atributos: "description", que é do tipo *String* e "type", que é uma referência para um outro esquema cujo é definido pelo nome "ProductType".

Já no diretório "config"são incluídos todos os arquivos de configuração, tais como: credenciais de acesso ao banco de dados, configurações de acesso a API e configurações de permissões do sistema. Esta prática, visa facilitar a manutenção do sistema, visto que os parâmetros necessários para executar o software são mantidos em uma única estrutura, facilitando a manutenção do sistema.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var ProductSchema = new Schema({
        description: {
                type: String,
                required: true,
                default: ''
        },
        type: {
                type: mongoose.Schema.Types.ObjectId,
                ref: 'ProductType',
                required: false
        }
});

module.exports = mongoose.model('Product', ProductSchema);
```

Figura 12. Organização da estrutura de diretórios do projeto

Além disso, o diretório "config", armazena todas as URI's (também chamadas de rotas) da API. Para tanto, estas rotas são configuradas em arquivos JavaScript localizados no subdiretório "routes". Para a criação das rotas, foi definido um esquema de versionamento da API, facilitando a inclusão de novos módulos, cujas novas URI's podem ser adicionadas nos arquivos. A nomeclatura dos arquivos deve obedecer uma determinada ordem, onde devem iniciar obrigatoriamente pela letra "v"e após deve conter um número, sob o qual determinará qual versão da API será utilizada. A Tabela 1 apresenta a mesma rota, porém com versões da API diferentes, onde a primeira URI representa um acesso a versão zero e a segunda URI apresenta um acesso à versão um da API.

Tabela 1. URI's para acesso aos dados de dispositivos da plataforma AgroNET Services, conforme versão da API

Consulta	URI
Listar dispositivos que obedecem o padrão da versão zero da API	dominio.com.br/api/v0/device
Listar dispositivos que obedecem o padrão da versão um da API	dominio.com.br/api/v1/device

3.4.2.1 Serviços Web

Como visto na sessão 3.4.2, a aplicação foi desenvolvida com NodeJS, utilizando o *framework* ExpressJS para implementação de uma API modelada no estilo arquitetural REST, a qual foi utilizada na troca de mensagens.

Sendo assim, o acesso a API é dado por um conjunto de URI's, as quais são responsáveis por encaminhar a solicitação do cliente, para que seu pedido seja atendido. Para a definição das URI's, o REST é caracterizado por visar a criação da nomeclatura de URI's de forma estruturada e descritiva [47]. Dessa forma, foi desenvolvido um padrão de consultas, conforme é apresentado pela Tabela 2.

Conforme visto na Tabela 2, as URI's seguem um padrão bem definido, visando a melhor estrutura objetivando facilitar a legibilidade das rotas criadas para a API. Conforme visto, o acesso aos dados de qualquer entidade do sistema ficou estruturado de forma onde apenas trocando o nome da entidade, o acesso é substituído. Para exemplificar isso, a URI "dominio.com/api/v0/device"acessa a

Tabela 2. Exemplo de URI's para acesso a versão zero da API da plataforma AgroNET Services, listando os dispositivos e seus dados e as fazendas e seus talhões

Consulta	**URI**
Obter dispositivos cadastrados no sistema	dominio.com/api/v0/device
Obter informações de um dispositivo	dominio.com/api/v0/device/:id
Obter dados de um dispositivo	dominio.com/api/v0/device/data
Obter dados de um dispositivo em um intervalo de datas	dominio.com/api/v0/device/data/range/ DD-MM-YYYY/DD-MM-YYYY
Obter o último registro de dados de um dispositivo	dominio.com/api/v0/device/data/last/:id
Obter informações das fazendas cadastradas no sistema	dominio.com/api/v0/farm
Obter informações de uma fazenda	dominio.com/api/v0/farm/:id
Obter talhões de uma fazenda	dominio.com/api/v0/farm/:id/field
Obter um talhão de uma fazenda	dominio.com/api/v0/farm/:id/field/:id

entidade de dispositivos do sistema, porém, para acessar os dados de outra entidade, basta substituir no nome "device"pelo nome de outra entidade, como por exemplo "farm". Neste caso, a nova URI ficará "dominio.com/api/v0/device".

Além disso, outras entidades, como por exemplo "field", a qual representa os talhões da propriedade, são dependentes de outra entidade, neste caso "farm", que representa as fazendas. Sendo assim, como os talhões fazem parte de uma propriedade, crio-se também este vínculo na URI, onde sempre, para acessar os dados de um talhão, é necessário informar o identificador da fazenda a qual pertence ("dominio.com/api/v0/farm/**:ID**/field).

O mesmo, acontece com as URI's utilizadas para as entidades que formam o módulo de acompanhamento e registros de safras, aplicação de insumos, químicos e ocorrências de patologias em um talhão. A construção deste módulo foi baseado no trabalho de Tibola et. al. [68], o qual objetiva a rastreabilidade digital do trigo. Dessa forma, a Tabela 3 apresenta as URI's para acompanhamento de safras, ocorrências de patologias e aplicação de insumos em um determinado talhão.

Tabela 3. Exemplo de URI's para acesso a versão zero da API da plataforma AgroNET Services, listando as entidades adaptadas do trabalho de Tibola et.al. [68]

Consulta	**URI**
Obter safras de um determinado talhão	dominio.com/api/v0/season/farm/:id_farm/field /:id_fiel/season
Obter safras de um talhão entre um determinado período	dominio.com/api/v0/season/farm/:id_farm/field /:id_fiel/season/range/DD-MM-YYYY /DD-MM-YYYY
Obter aplicação de insumos em um talhão em uma determinada safra	dominio.com/api/v0/season/farm/:id_farm/field /:id_fiel/season/:id_season/application
Obter ocorrências de patologias em um talhão em uma determinada safra	dominio.com/api/v0/season/farm/:id_farm/field /:id_fiel/season/:id_season/pathology

Como método de proteção ao acesso a API REST, foi utilizado o padrão OAuth2, por ser altamente utilizado no mercado de desenvolvimento web, e também por implementar esquemas de autenticação modernos, os quais operam em conjunto para restringir o acesso a API. Para tanto, o OAuth2 utiliza uma combinação de técnicas de autenticação, visando a mínima exposição das informações de *login* do usuário [69] [70].

Sendo assim, foi implementado no sistema um esquema de autenticação utilizando o padrão OAuth2. Para tanto, foi implementado no sistema um cadastro de usuários, com diferentes níveis de acesso. Portanto, é possível definir um os dados específicos que podem ser acessados por determinado grupo de usuários.

3.5 DISCUSSÃO

Tendo em vista a grande quantidade de dados que podem ser gerados por atuadores, sensores e outros equipamentos eletrônicos que são aplicados na área agrícola, este trabalho objetivou na construção de uma plataforma para armazenar os dados provenientes da Agricultura de Precisão.

Com o desenvolvimento deste trabalho, percebe-se que o emprego do banco de dados MongoDB, juntamente com NodeJS, os quais são responsáveis por prover o acesso aos dados por meio de uma API web, constituem em uma solução robusta e de fácil desenvolvimento. Isso, pode ser atribuído ao fato de que o desenvolvimento utilizando conceitos modernos pode facilitar a alteração e a manutenção do sistema.

Além disso, o desenvolvimento modular, permite a adição de novas funcionalidades sem necessidades de grandes e custosas alterações no código-fonte do sistema, o que se deve também, ao fato da utilização de um banco de dados orientado a documentos, como é o caso do MongoDB, que oferece recursos para manipulação em objetos, e facilita a criação de novas estruturas, sem a necessidade de custosas alterações em relacionamentos de tabelas, além de se demonstrar muito versátil na modelagem e representação dos dados.

3.6 CONCLUSÃO

Estimulado pelo aumento de soluções que englobam o mercado da Agricultura de Precisão, e pela necessidade de ferramentas robustas e capazes de oferecer recursos capazes de gerir os dados produzidos em ambiente agrícola, desenvolveu-se a solução apresentada neste trabalho, para armazenamento e disponibilização de grandes volumes de dados provenientes da Agricultura de Precisão.

Para a elaboração deste projeto, busco-se a inserir recursos que pudessem facilmente ser modificados e adaptados para diferentes situações. Para tanto, a padronização da representatividade dos dados e a visão geral do cenário da Agricultura de Precisão, possibilitaram o desenvolvimento de uma plataforma capaz de receber, tratar, armazenar e disponibilizar os dados do cenário envolvido, o que define que o objetivo principal do trabalho foi devidamente alcançado.

4. AGRONET GIS: UMA SOLUÇÃO PARA ARMAZENAMENTO DE DADOS GEOESPACIAIS PARA A AGRICULTURA DE PRECISÃO

4.1 RESUMO

A Agricultura de Precisão fornece uma grande variabilidade de dados que são úteis na geração de conhecimento aplicada no processo de tomada de decisões. Dados geoespaciais fazem parte deste conjunto de dados e auxiliam no entendimento do campo. Atualmente, existe uma grande quantidade de dados geográficos disponíveis, que são coletados por satélites, ferramentas de GPS, entre outros meios de geração de dados vetoriais e de imagens. Para tanto, o objetivo deste trabalho é apresentar uma plataforma para armazenamento destes dados geoespaciais disponíveis no cenário da AP. Como resultado, obteve-se uma solução de banco de dados a qual oferece suporte a tipos de dados geoespaciais, os quais podem ser acessados por meio de serviços *web*.

4.2 INTRODUÇÃO

A necessidade de acumulação de uma quantidade cada vez maior de dados é um desafio cada vez mais necessários pelos bancos de dados. Neste sentido, diversos estudos estão sendo abordados para suprir tal necessidade e integrar soluções diferenciadas e capazes de trazer novos recursos para a forma na qual os dados são armazenados. Devido a esta situação, soluções de bancos de dados espaço-temporais tem sido bastante utilizados para suprir diversas necessidades do mercado [38].

Por outro lado, a necessidade de armazenar dados espaço-temporais, ou também chamado de espaciais, pode ser definido com um processo histórico, que acompanha a humanidade por meio de registros de tempo e localização geográfica. Na atualidade isso pode ser desenvolvido por meio da informática, sendo elaborado um banco de dados geográficos. Trata-se de um processo dinâmico, onde a variável tempo torna-se essencial, sendo integrada à representação espacial dos objetos [71] [72].

Na área da agricultura, os dados geoespaciais tem ganhado uma nova perspectiva por conta da evolução das tecnologias em sistemas orbitais e de comunicações. O aprimoramento dos Sistemas de Posicionamento Global (GPS) e a cada vez maior presença da Agricultura de Precisão (AP), são fundamentais para produzir tecnologias como piloto automático e sensores de produtividade. Por conta disso, o geoprocessamento se faz de suma importância para a área agrícola [71]. Na AP, sistemas de informação geográfica são muito importantes, pois a AP trabalha variabilidade espacial que requer dados sobre solo, cultura máquinas agrícolas e aplicações [5].

Para Linseisen [73], os Sistemas de Informação Geográfica (SIG) tem o poder de agregar valores de dados de pontos espaciais com uma enorme exigência de espaço em disco para atributos

e variáveis coletadas por equipamentos de AP, principalmente para abordagem em tempo real com superposição de mapa e para a contabilização de custos e produtos. Dessa forma, inúmeros dados podem ser obtidos, como por exemplo: informações de coordenadas de latitude e longitude de maquinários agrícolas, obtenção de dados de custo (produtos, ...) para a contabilização de custos de produção por espaço geográfico, acompanhamento da largura de trabalho parcial (utilizado por um pulverizador de fertilizantes ou por um implemento de plantio).

A AP trouxe consigo uma variedade de novos desafios para a pesquisa e consequentemente a área do geoprocessamento está incluído neste processo. Neste sentido, Nikilla [74], salienta que são necessários investimentos para tornar um sistema utilizado pela maioria dos agricultores. Para tanto, trabalhos de geoprocessamento e com uma arquitetura de software capaz de enfatizar a comunicação e transferência de dados entre todos os elementos da agricultura moderna são essenciais para garantir o sucesso destes sistemas [74].

Além disso, existe uma gama muito grande de dados geográficos disponíveis no mundo todo. Milhares de imagens coletadas por satélites, aviões, drones e dados geoespaciais disponibilizados por instituições, tal como NASA *National Aeronautics and Space Administration*, fazem parte desta grande massa de dados que são disponibilizados. Por conta disso, quando se fala em armazenamento de informações geoespaciais a grande parte dos casos impera a desorganização, ocasionando em uma dificuldade na hora de recuperá-los. Neste sentido, mostra-se a importância de acoplar boas tecnologias e bons métodos de armazenamento destes dados [75] [76].

Dessa forma, é visto o grande volume de dados geoespaciais que são disponibilizados a todo instante, e percebe-se que a AP é um setor no qual estes dados representam fundamental importância para a agricultura moderna. Para tanto, este capítulo tem por objetivo apresentar uma solução para armazenamento de dados geoespaciais para a AP. Para tanto, a seção 4.3 apresenta os recursos utilizados para a implementação do sistema. Na seção 4.4 apresenta a arquitetura, o desenvolvimento e os resultados da implementação da plataforma. Por fim, a seção 4.5 apresenta conclusões do trabalho.

4.3 MATERIAL E MÉTODOS

Este trabalho tem por objetivo a construção de uma solução para armazenamento e disponibilização de dados geoespaciais. Para tanto, buscou-se elaborar um modelo de banco de dados capaz de armazenar dados geoespaciais dos mais diferentes tipos, tais como: geometrias tais como pontos ou coordenadas geográficas, polígonos, e objetos de imagem raster. Além disso, buscou-se também desenvolver uma plataforma *web* capaz de disponibilizar os dados armazenados no banco de dados utilizando a Arquitetura Orientada a Serviços, do inglês *Services Oriented Architecture* (SOA), utilizando o padrão REST.

4.3.1 Banco de dados

A utilização de bancos de dados espaciais se caracterizam por armazenar não somente dados, mas também permite o relacionamento destes com posições geográficas no espaço. Em sua grande maioria, os banco de dados espaciais podem também ser caracterizados por Sistemas de Informações Geográficas (SIG), ou do inglês *Geographic Information System* (GIS). Para tanto, a classificação dos SIG's é feita por permitirem a realização de operações complexas aos dados georreferenciados, visto que possuem a capacidade de manipular os dados espaciais (cartográficos, de sensoriamento remoto e modelos numéricos de espaço geográfico) e dados não espaciais (descritivos ou alfanuméricos), de forma integrada, provendo recursos para realização de cálculos matemáticos, análise e consultas a estes dados [29].

Para tanto, como banco de dados espacial, adotou-se o SGBD PostgreSQL, o qual oferece extensibilidade, sendo possível adicionar características capazes de operar com tipos de dados específicos [32]. Para operar com o tipo de dado espacial, a extensão PostGIS foi instalada, a qual incluí todas as funcionalidades e objetos padrão OpenGIS (*Open Geospartial Consortium*), que é um consórcio internacional que nasceu da iniciativa de garantir a interoperabilidade de softwares por meio da padronização das funções que tratam dos dados georreferenciados no banco de dados. Com isso, permite-se que diferentes softwares que utilizam informações espaciais, consigam trabalhar de forma uniforme com estas informações [36].

Sabendo disso, a modelagem foi desenvolvida a fins de oferecer suporte de forma genérica aos dados providos pela Agricultura de Precisão. Dessa forma, desenvolveu-se suporte a tipos de dados geométricos, capazes de oferecer suporte dados como mapas de propriedades agrícolas, delimitação de áreas de cultivo ou qualquer outro tipo de dado geométrico. Além disso, é de suma importância também, o armazenamento de dados raster, os quais podem ser produzidos por mapas de produtividades, imagens coletadas por drones, satélites ou outros tipos de dados raster. Sendo assim, os itens abaixo descrevem tais tipos de dados.

- ShapeFiles: ShapeFile é um formato geoespacial regulamentado pelo *Environmental Systems Research Institute* (ESRI). Este formato é utilizado principalmente para armazenar a descrição de uma área geográfica. Neste sentido, as geometrias são o principal objetivo de serem armazenados nestes formatos, tais como pontos, linhas ou areas delimitadas por coordenadas de GPS que podem ser dispostas em gráficos, por exemplo [77]. Dessa forma, um ShapeFile armazena uma topologia geometriaca e atributos contendo informações de um conjunto de dados. A geometria é armazenada no ShapeFile de forma comprimida, com um vetor de coordenadas . Um ShapeFile é formato por um arquivo principal com extensão *.shp, um arquivo indexador (*.shx) e um arquivo de dados (*.dbf) [78].

- Raster: O tipo de dado Raster é definido como sendo um modelo de representação de dados, o qual é representado por uma matriz com células de tamanhos iguais dispostas em linhas e colunas. Cada unidade de célula do Raster contém um valor de atributo e as referentes coordenadas

geográficas de localização [79]. Um raster pode ser constituído de várias *layers* (camadas), onde cada banda é composta de um vetor de pixels (células). São formatos rasters: TIFF e JPEG 2000 [80].

4.3.2 Serviços *Web*

Visando uma plataforma para disponibilização dos dados para a internet, a construção do software baseou-se em tecnologias *web*, visando facilitar o acesso aos dados e permitir a interação entre sistemas por meio de serviços *web* [65]. Para tanto, o sistema foi construído utilizando como base os conceitos da Arquitetura Orientada a Serviços (SOA), a qual é responsável por disponibilizar os dados armazenados no banco de dados PostGIS.

Para a implementação da plataforma, o NodeJS foi utilizado. Dessa forma, acomplaram-se os frameworks *ExpressJS*, o qual foi utilizado para a modelagem da plataforma no padrão MVC (*Model-View-Controller*), o qual é responsável por permitir a modelagem da plataforma sobre tal padrão, de forma modular e robusta [63] [64]. Além disso, o pacote NodeJS *PG* também foi utilizado para permitir a conexão e a manipulação dos dados com o banco de dados. O pacote PG utiliza recursos não-bloqueantes para NodeJS, facilitando a manipulação dos dados de forma assíncrona, mantendo a forma de trabalho original do NodeJS [56] [55].

Como padrão de disponibilização de dados via REST, o formato GeoJSON foi utilizado. O GeoJSON, especifica um formato para representar uma variedade de estruturas de dados geográfias, suportando o formato de dados geométricos de: *Point, LineString, Polygon, MultiPoint, MultiLineString e MultiPolygon*, além de permitir a adição de objetos adicionais na sua estrutura. O formato GeoJSON é especificado pela RFC 7946 [81], a qual foi publicada em Agosto de 2016, em substituição a especificação do GeoJSON de 2008 [82].

4.3.3 Fonte de Dados

Para fins de estudo e caso de testes para o sistema, utilizou-se de dados satelitais de NDVI, os quais foram obtidos do instrumento MODIS. MODIS é um instrumento lançado a bordo dos satélites TERRA (conhecido como EOS AM-1) e AQUA (conhecido como EOS PM-1). O índice de vegetação MODIS é produzido com intervalos de 16 dias em diferentes resoluções espaciais, que variam entre 250 metros, 500 metros e 1000 metros. O acesso aos dados da plataforma MODIS são públicos, e podem ser transferidos do servidor da NASA pelo endereço: https://modis.gsfc.nasa.gov/data/ [83].

4.4 RESULTADOS

Baseando-se em uma arquitetura de modelo Cliente x Servidor, aplicado a troca de dados via serviços *web*, o sistema AgroGIS foi construído principalmente para ser utilizado na disponibilização de dados geoespaciais. Dessa forma, seus dados podem ser aplicados na construção de mapas

geográficos e imagens georeferenciadas. Sendo assim, os dados disponibilizados pela plataforma, podem ser utilizados por outros sistemas dos quais podem ser úteis. Deste modo , a Figura 14 apresenta um diagrama geral do sistema.

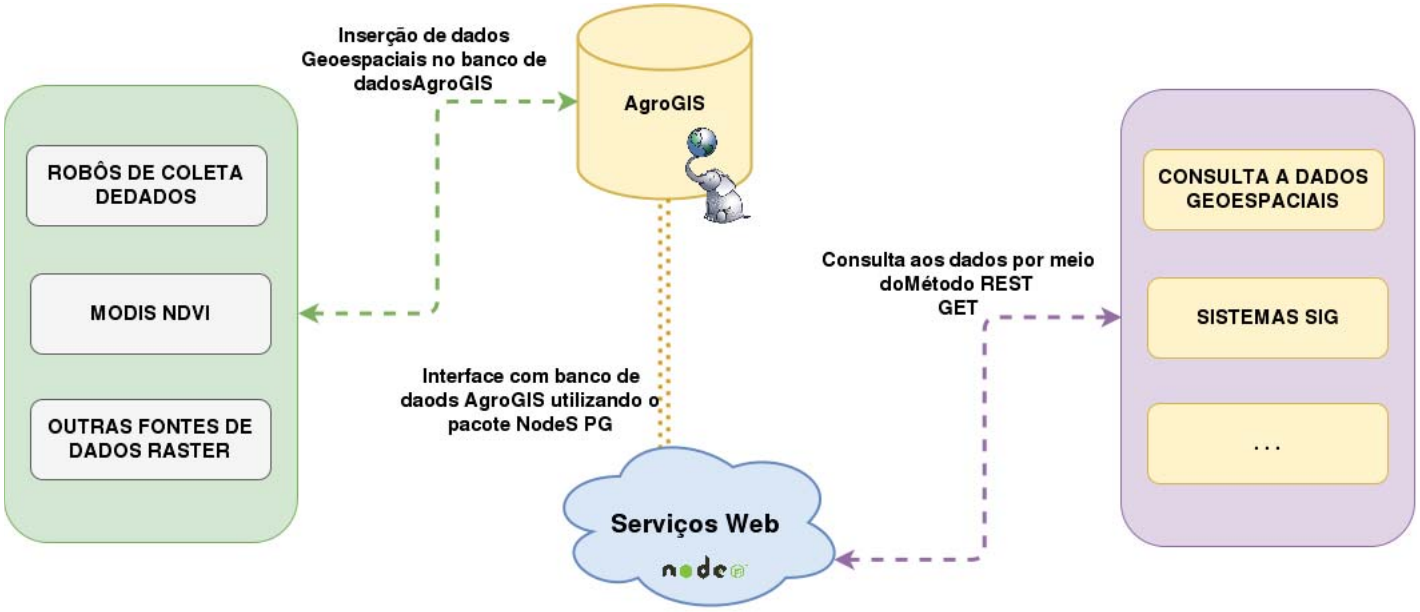

Figura 13. Diagrama geral da plataforma AgroGIS

No que se refere ao acesso aos dados armazenados na plataforma AgroGIS, o padrão arquitetural REST foi utilizando, sendo este responsável por atender as requisições dos usuários nos padrões JSON e GeoJSON. Dessa forma, visa-se a utilização de tecnologias modernas para o tráfego de dados, além de permitir uma padronização na troca de dados com o sistema, facilitando a ampliação deste. Sendo assim, as seções abaixo, detalham a arquitetura e a operação do sistema.

4.4.1 Banco de Dados

O banco de dados foi construído, visando servir como uma solução genérica para armazenamento de dados raster e também de geometrias. A Figura 14 apresenta o diagrama do banco de dados da plataforma AgroGIS.

Para o armazenamento dos dados de tipo raster, foram criadas duas tabelas: "raster_layers"e "raster_files". Na tabela raster_files, são armazenadas informações de descrição do raster, tais como: descrição, data, hora, origem dos dados, tipo de sensores (em casos de dados de satélite por exemplo). Já na tabela raster_layers, estão contidos todas as camadas (*layers)* do raster. Nessa tabela são armazenadas também as demais informações do raster propriamente dito. Portanto informações de tipo de dado (NDVI, Temperatura, chuva, ...), resolução espacial, projeção e unidade de medida da variável armazenada são mantidas. Desta forma, a solução AgroGIS foi pensada para que cada camada do raster seja armazenada em uma linha diferente da tabela, pois apesar de ser possível armazenar várias bandas de um raster em apenas uma linha da tabela, para esta modelagem, percebe-se que criar um novo registro para cada camada oferece maior agilidade para localizar um determinado

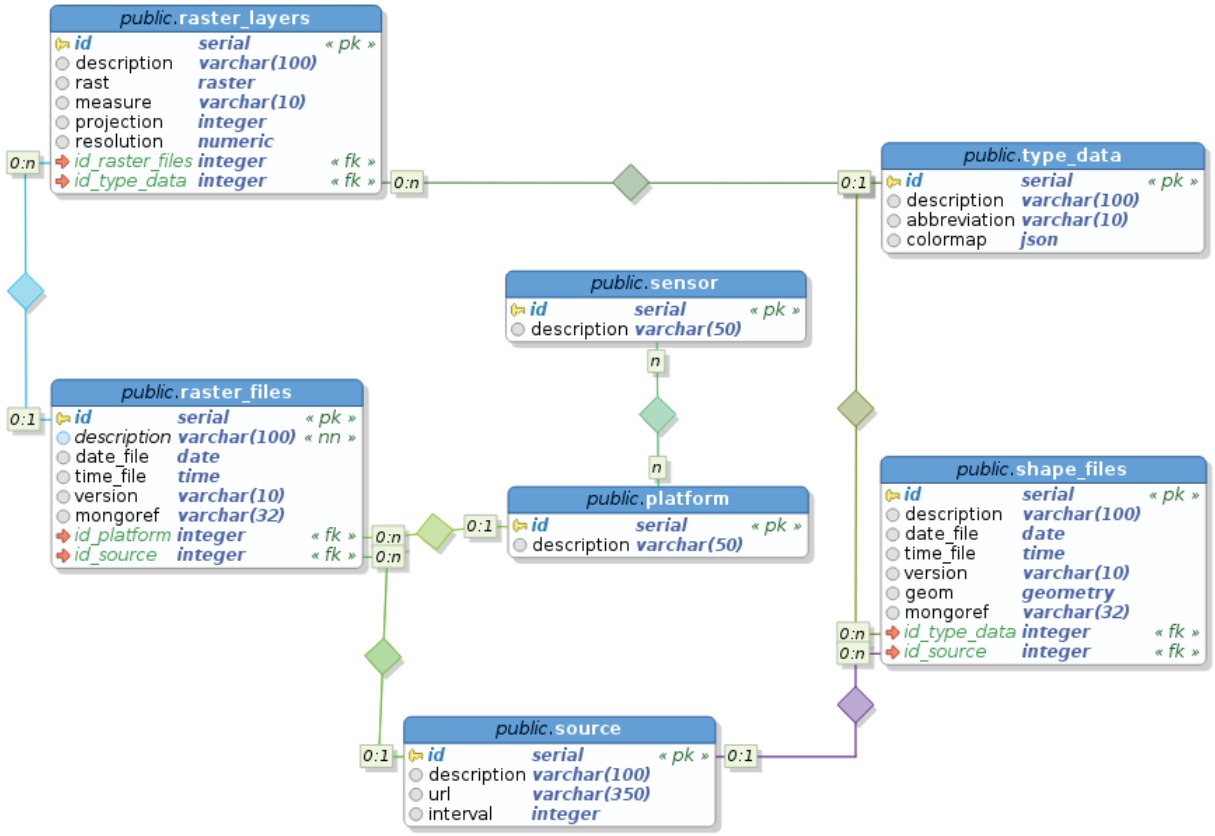

Figura 14. Diagrama do banco de dados da plataforma AgroGIS

arquivo. Neste sentido, caso um raster possuir várias camadas, estas deverão ser armazenadas separadamente, onde farão referência ao raster por meio do campo "id_raster_file", o qual faz referência para a tabela "raster_file".

Já no que se refere ao armazenamento dos arquivos de formato *shape*, foi criada uma tabela denominada de "shape_files". Nessa tabela, podem ser armazenados quaisquer dados de tipos geometricos, tais como dados de tipo *Point*, *Polygon*, *Multipolygon*, *Line*, *Polyline*. Dessa forma, a modelagem facilita o armazenamento de diferentes tipos de dados geométricos, evitando a criação de diferentes colunas na tabela, e consequentemente evitando a inserção de registros vazios. Além disso, permite-se inserir registros de descrição do arquivo, tais como data, hora, descrição. Também, a tabela possuí relacionamento com as tabelas de "type_data", a qual permite identificar diferentes dados, tais como arquivos *shapes* de irrigação, mapas de propriedades rurais, mapas de áreas de cultivo, pontos de divisas, estradas e rios, e *"source"*, que é responsável por armazenar a origem do arquivo.

Visando a possibilidade de integração com o banco de dados da plataforma AgroNET Services descrita no capítulo 3, criou-se uma coluna de referência ao MongoDB, a qual pode armazenar uma variável do tipo String de até 32 caracteres. Com isso, este modelo oferece suporte ao armazenamento de um identificador "_id"do MongoDB, o qual pode ser utilizado para vincular qualquer registro existente no MongoDB, com o o banco de dados proposto. Esta relação, pode ser utilizada por exem-

plo para o armazenamento de dados georeferenciados de propriedades e talhões, cujas informações são armazenadas no modelo AgroNET Services.

4.4.2 Serviços Web

Para prover o acesso aos dados armazenados na plataforma AgroNET GIS, foi desenvolvida uma estrutura de serviços *web*, aplicando os conceitos da Arquitetura Orientada a Serviços (SOA), resultando em uma *Application Programming Interface* (API). Para isso, de forma a construir uma estrutura a parte e independente da aplicação apresentada no Capítulo 3 deste trabalho, determinou-se a construção de um novo software utilizando NodeJS, com o *framework* ExpressJS, possibilitando a implementação de uma API REST, utilizando o padrão arquitetural MVC. Foram construídos também, recursos de versionamento de API, seguindo o mesmo modelo apresentado no Capítulo 3 deste trabalho. P

Diante disso, foram elaboradas URI's para realização de consultas por meio do método GET. Para isso, utilizou-se da padronização da URI de acordo com o nome da tabela do banco de dados, visando o melhor gerenciamento e melhor legibilidade da API. Sendo assim, todas os registros das tabelas do banco de dados podem ser acessados por meio da API desenvolvida, conforme é apresentado na Tabela 4.

Tabela 4. URI's para acesso aos dados das tabelas da plataforma AgroNET GIS por meio do método GET

Consulta	URI
Obter a listagem de todas as fontes de dados cadastradas	http://dominio.com/api/gis/v0/source
Obtera listagem da fonte de dados cujo id foi passado na URI	http://dominio.com/api/gis/v0/source/:id
Obter a listagem de todos os tipos de dados cadastrados	http://dominio.com/api/gis/v0/typedata
Obtera listagem do tipo de dado cujo id foi passado na URI	http://dominio.com/api/gis/v0/typedata/:id
Obter a listagem de todas as plataformas cadastradas	http://dominio.com/api/gis/v0/platform
Obter a listagem da plataforma cujo id foi passado na URI	http://dominio.com/api/gis/v0/platform/:id
Obter a listagem de todos os sensores existentes na plataforma cujo id foi passado na URI	http://dominio.com/api/gis/v0/platform/:id/sensor

Para obter registros geometricos armazenados na tabela "shape_files", foram definidas URI's de forma a fazer uma listagem dos dados armazenados sem exibir a geometria, com fins de facilitar a interpretação dos dados, visto que geralmente os dados de uma geometria compreendem um grande volume. Porém, também foram determinadas URI's capazes de retornar todas as informações do shape, juntamente com sua geometria. Neste caso, os dados retornados pelo serviço *web* são for-

matados em JSON e a geometria, é agregada aos dados retornados, com formato GeoJSON. Para exemplificar isto, a Tabela 5 apresenta as URI's definidas para obtenção de arquivos shape.

Tabela 5. URI's para acesso aos dados da tabela "shape_files"da plataforma AgroNET GIS

Consulta	URI
Obter a listagem dos dados da tabela "shape_files"	http://dominio.com/api/gis/v0/shape
Obter geometria de um shapefile	http://dominio.com/api/gis/v0/shape/geom/:id
Obter a listagem dos dados da tabela "shape_files" de uma determinada data	http://dominio.com/api/gis/v0/shape/date/ :dd-mm-yyyy
Obter a listagem dos dados da tabela "shape_files" de um intervalo de datas	http://dominio.com/api/gis/v0/shape/date /range/:dd-mm-yyyy/:dd-mm-yyyy

O resultado da consulta de um shape com sua geometria, deve ser a união dos atributos armazenados na tabela "shape_files", com a geometria do arquivo. Dessa forma, será incluído no retorno um atributo chamado "geometry", o qual deve conter um atributo "type"o qual indica o formato da geometria, e posteriormente todos os valores de coordenadas de latitude e longitude do arquivo, serão incluídas no atributo "coordinates", no formato GeoJSON. Para converter os registros shape em formato GeoJSON, foi utilizada a função nativa do PostGIS "ST_AsGeoJSON()", cuja é responsável por receber por parâmetro uma geometria, e como retorno apresenta a mesma geometria no formato GeoJSON [84]. Para isso, a Figura 15 apresenta um exemplo de consulta a um arquivo shape, com base na URI:http://dominio.com/api/gis/v0/shape/geom/1.

Para consultas a dados do tipo raster, foram definidas um conjunto de URI's específicas para transformar e retornar os valores em formato geométrico. Para tanto, a matriz raster é inteiramente convertida para um conjunto de formatos geométricos, sobre os quais são atribuídos também um valor, cujo é proveniente da célula do raster a qual a geometria se refere. Para realizar esta conversão, foi escrita uma função no SGBD PostgreSQL, a qual recebe por parâmetro um raster e retorna suas geometrias vinculadas aos valores as quais pertence. Estes dados geométricos, já são formatados em GeoJSON com o uso da função "ST_AsGeoJSON()". A Figura 16 apresenta a função AgroGIS_Raster2Geom(rast raster), construída para a conversão de dados de tipo raster para geometria.

A função AgroGIS_Raster2Geom(rast raster) é utilizada sempre que for requisitada alguma URI para acesso a geometria do raster. Para tanto, foram construídas URI's apenas para informe de listagem dos registros inseridos na tabela "raster_files"e "raster_layers", bem como URI's específicas para obter os valores geométricos do raster. Além disso, também foram desenvolvidas rotas apenas para consultas aos registros, porém sem retornar a geometria do raster. Esta alternativa pode ser utilizada pois evita a troca de mensagens excessivas, visto que geralmente os registros raster contém uma grande quantidade de dados, e isso evita a troca de valores desnecessários. A Tabela 6 apresenta as URI's de consulta aos dados raster.

No que se refere a uma requisição REST GET para obter a geometria de um determinado raster cadastrado, a Figura 17 apresenta um JSON das informações do raster armazenadas nas tabelas "raster_files"e "raster_layers", juntamente com os valores geométricos do raster. Quanto aos valores geométricos do raster, o atributo "raster_geometry"contém as especificações geométricas de

```
{
  "message": "success",
  "value": {
    "id": 1,
    "description": "Granja Carazinho",
    "date_file": "2016-11-15T00:00:00.000Z",
    "time_file": "21:41:51",
    "id_type_data": 1,
    "mongoref": null,
    "geometry": {
      "type": "MultiPolygon",
      "coordinates": [
        [
          [
            [
              -52.8785047820016,
              -28.2494564298971,
              0
            ],
            [
              -52.877675692815,
              -28.2500636329405,
              0
            ],
            [
              -52.8767439236686,
              -28.2502816671219,
              0
            ],
            [
              -52.8757167211961,
              -28.2509631752485,
              0
            ],
```


Figura 15. Consulta a uma geometria shape do banco de dados, utilizando a URI: http://dominio.com/api/gis/v0/shape/geom/1

```
CREATE OR REPLACE FUNCTION AgoGIS_Raster2Geom(rast raster)
  RETURNS json AS
$BODY$
   DECLARE
      j json;
      BEGIN
         j := (SELECT row_to_json(fc)
            FROM
               (SELECT 'FeatureCollection' as type, array_to_json(array_agg(feats)) AS features
                  FROM
                     (SELECT
                        'Feature' as type,
                           ST_AsGeoJson((gv).geom)::json as geometry,
                           row_to_json((SELECT props FROM (SELECT (gv).val AS pixel_value) AS props ))
                                AS attributes
                        FROM
                           (SELECT
                              ST_PixelAsPolygons(rast) as gv
                           ) json
                     ) feats
               ) fc);
               RETURN j;
      END;
$BODY$
LANGUAGE plpgsql;
```

Figura 16. Função AgroGIS_Raster2Geom(rast raster) para conversão de dados raster para geometria

cada célula da matriz raster, constituindo um conjunto de elementos do tipo "*Polygon*"onde cada um destes é composto pelas coordenadas que constituem o polígono. Vinculado a cada polígono, também

Tabela 6. Exemplo de URI's para consulta aos dados das tabelas "raster_files"e "raster_layers"da plataforma AgroNET GIS

Consulta	URI
Obter a listagem dos dados das tabelas "raster_files" e "raster_layers"	http://dominio.com/api/gis/v0/raster
Obter geometria de um raster	http://dominio.com/api/gis/v0/raster/geom/:id
Obter a listagem dos dados das tabelas "raster_files" e "raster_layers"de uma data especificada	http://dominio.com/api/gis/v0/raster/date/ dd-mm-yyyy
Obter a listagem dos dados das tabelas "raster_files" e "raster_layers"de um intervalo de datas	http://dominio.com/api/gis/v0/raster/date/ range/dd-mm-yyyy/dd-mm-yyyy
Obter a listagem dos dados das tabelas "raster_files" e "raster_layers"de uma fonte cadastrada na tabela "source"	http://dominio.com/api/gis/v0/raster/source/ :source
Obter a listagem dos dados das tabelas "raster_files" e "raster_layers"de um tipo cadastrado na tabela "type_data"	http://dominio.com/api/gis/v0/raster/typedata/ :typedata
Obter a listagem dos dados das tabelas "raster_files" e "raster_layers"de uma fonte cadastrada na tabela "source", de um tipo cadastrado na tabela "type_data" e de uma data especificada	http://dominio.com/api/gis/v0/raster/source/ :source/typedata/:typedata/date/dd-mm-yyyy

existe um atributo "*attributes*"que contém os valores associados a célula da matriz, e neste caso, contendo o valor de NDVI baixado da plataforma MODIS.

4.5 CONCLUSÃO

Com a crescente demanda por Sistemas de Informação Geográfica e por sua importância no cenário agrícola, se torna cada vez mais necessário o desenvolvimento de soluções de armazenamento e disponibilização destes tipos de dados. Em razão disso e da necessidade da evolução de sistemas de AP com suas necessidades e sua grande área de abrangência demandam também a capacidade de integração com outros sistemas.

Neste sentido, uma plataforma *web* capaz de armazenar e disponibilizar dados geoespaciais por meio de um protocolo padrão se torna importante para o cenário da AP, visto que além de permitir a troca de informações com o usuário final, também permite a integração com diferentes sistemas de AP, o que se torna essencial para o mercado da AP . Para tanto, este trabalho buscou englobar de uma maneira geral, diferentes tipos de dados geoespaciais que estão presentes na AP, se fazendo importante para o armazenamento de mapas agrícolas entre outros tipos de dados espaciais, que são essenciais para o sucesso da cadeia produtiva.

```
"value": {
       "raster_files": 11,
       "description": "NDVI",
       "date_file": "2001-01-17T00:00:00.000Z",
       "raster_layers": 11,
       "raster_geometry": {
           "type": "FeatureCollection",
           "features": [
               {
                   "type": "Feature",
                   "geometry": {
                       "type": "Polygon",
                       "coordinates": [
                           [
                               [
                                   -53.11,
                                   -28.1
                               ],
                               [
                                   -53.1075,
                                   -28.1
                               ],
                               [
                                   -53.1075,
                                   -28.1025
                               ],
                               [
                                   -53.11,
                                   -28.1025
                               ],
                               [
                                   -53.11,
                                   -28.1
                               ]
                           ]
                       ]
                   },
                   "attributes": {
                       "pixel_value": 0.8632
                   }
               },
```


Figura 17. Saída JSON/GeoJSON de um raster convertido para o formato GeoJSON

5. AGRONET VIEW: UMA PLATAFORMA DE ACESSO, VISUALIZAÇÃO E GERAÇÃO DE DADOS DA AGRICULTURA DE PRECISÃO

5.1 RESUMO

Ferramentas de gestão agrícola podem ser aliadas para o acompanhamento e o gerenciamento das propriedades agrícolas, onde a visualização de dados fornecidos por sensores e equipamentos eletrônicos agrícolas auxiliam o processo de tomada de decisões. No entanto, a grande quantidade e variedade de tipos de dados da AP, dificultam a manipulação destes dados. Para tanto, o objetivo deste trabalho é apresentar uma plataforma para acompanhamento e visualização dos dados da AP, disponibilizados pelas plataformas AgroNET Services e AgroNET GIS. No que se refere a resultados, obteve-se uma plataforma *web* para acesso, visualização e geração de dados consumidos pela AP.

5.2 INTRODUÇÃO

A Agricultura de Precisão tem sido cada vez mais decisiva na cadeia produtiva. Um grande número de sistemas e dispositivos são disponibilizados no mercado visando o aumento da produtividade no campo, assim como o aumento da qualidade do produto e a redução dos impactos ao meio ambiente. Estes sistemas buscam beneficiar os produtores rurais, permitindo um melhor controle na aplicação dos insumos e uma melhor distribuição de produtos, evitando perdas e por consequência reduzindo os custos de produção [5].

Neste sentido, o desenvolvimento de um software personalizado, com uma interface de usuário customizada e que permita a integração com outros sistemas, pode ser uma alternativa viável para o cenário agrícola. Neste aspecto, uma solução de interface gráfica integrada a serviços *web*, pode facilitar a expansão de recursos da ferramenta[5].

Neste caso, uma interface gráfica seguindo especificações e critérios de usabilidade e comunicação visual se tornam importantes para qualquer sistema. Assim, o uso de elementos gráficos, cores, fontes, tamanhos e disposição dos elementos devem ser observados para facilitar o seu uso, visto que as interfaces gráficas (principalmente *web*) tendem a mudar de uma máquina para a outra devido a variáveis como tipo de sistema e tamanho da tela, por exemplo [85] [86].

Para tanto, estão disponíveis na *web* um conjunto de *frameworks* que facilitam o desenvolvimento da interface do software, como Bootstrap [87] e Angular Material [88]. Estas ferramentas facilitam a criação de interfaces com componentes padronizados e que oferecem recursos de responsividade, visando o desenvolvimento rápido do software [89] [90].

Neste sentido, este capítulo tem por objetivo o desenvolvimento de uma interface gráfica para validação e gerenciamento dos dados das plataformas AgroNET Services 3 e AgroNET GIS 4. Para

tanto, foi desenvolvida uma interface gráfica *web* utilizando recursos de *frameworks* de interface gráfica e de desenvolvimento de software *web front-end*, capaz de trocar mensagens com as plataformas por meio de serviços *web* REST.

Para a melhor organização, este trabalho está dividido em 3 partes principais. A seção 5.2 apresenta os conceitos iniciais deste capítulo. A seção 5.3 apresenta os recursos utilizados para a elaboração do trabalho, e por fim, as seções 5.4 e 5.5 apresentam os resultados e as conclusões do trabalho, respectivamente.

5.3 MATERIAIS E MÉTODOS

Este trabalho tem por objetivo validar as plataformas AgroNET Services e AgroNET GIS. Para tanto, buscou-se a utilização de *frameworks* de desenvolvimento *web front end* para priorizar o desenvolvimento ágil. As seções abaixo descrevem as tecnologias e os métodos utilizados para o desenvolvimento do trabalho apresentado neste capítulo.

5.3.1 AngularJS

O angularJS foi utilizado nessa plataforma pois, além de ser um framework JavaScript para construção de aplicações web de código aberto e um framework estrutural para aplicativos web dinâmicos, o AngularJS permite que você use o HTML como linguagem de modelo e permite estender a sintaxe do HTML para expressar os componentes do aplicativo de forma clara e sucinta. Ele funciona como uma extensão do HTML, diferente de outros frameworks JavaScript, não é necessário manipular o DOM. Isso possibilita a criação de um front end mais organizado e sem a necessidade de manipular o html e os dados.

5.3.2 Elementos de interface gráfica

O uso de softwares e bibliotecas de interface gráfica são fundamentais na construção de um *web* site, pois trazem componentes gráficos estilizados e padronizados, prontos para serem aplicados no desenvolvimento *web* [87].

Para construir o fornt-end do projeto, foi utilizado a biblioteca gráfica Angular Material, uma ferramenta que fornece um conjunto de componentes de interface de usuário reutilizáveis, testados e acessíveis. Os componentes do Angular Material ajudam na construção de páginas web atraentes, consistentes e funcionais, ao mesmo tempo em que aderem aos princípios modernos de web design, como a portabilidade do navegador e a independência de dispositivos. Ele ajuda na criação de sites mais rápidos, bonitos e responsivos [88].

Além do Angular Material, utilizou-se do Bootstrap, com a mesma ideia de reutilizar componentes prontos mas que não estavam disponíveis no Angular Material, então fazendo a união dos dois foi possível criar uma interface mais completa e acessível para o usuário [87].

5.4 RESULTADOS

Visando a integração deste sistema, com a plataforma AgroNET Services, apresentada no capítulo 3 deste trabalho, o trabalho foi desenvolvido objetivando a construção de uma interface gráfica com o usuário para manipulação dos dados presentes na plataforma AgroNET Services.

Para tanto, foram utilizados recursos de desenvolvimento de software *web front-end*, com vistas a facilitar a interação do usuário com o sistema, além de servir como um sistema de validação da plataforma AgroNET Services. Dessa forma, A troca de dados com o sistema AgroNET Services foi feita por meio de serviços REST com troca de dados JSON, de acordo com a especificação com a troca de dados da plataforma AgroNET Services.

5.4.1 Componentes do Sistema

O sistema consiste de páginas para fazer cadastro de dispositivos, maquinas, propriedades, novos usuários e um painel administrativo onde é possível cadastrar novas doenças, produtos, solos e variedades. O acesso a essas diferentes interfaces de cadastro dependem do nível de permissão que o usuário possui dentro do sistema.

5.4.1.1 *Login* no sistema

Para acessar o sistema o usuário deverá entrar com suas credenciais em uma tela de *login* (Figura 18), cuja autenticação é feita por meio da entrada de usuário e senha, previamente cadastrados na plataforma.

5.4.1.2 Dispositivos

Foram criadas interfaces para manutenção de tipos de dispositivos e dispositivos. Os tipos de dispositivos, referem-se a aplicabilidade do dispositivo, onde por exemplo, todos os dispositivos de plantio, podem ser agrupados por um tipo denominado de "Monitor de Plantio". Para tanto, informações como sensores e comandos que podem ser executados pelos dispositivos, podem ser armazenados.

Para os dispositivos, isto é, qualquer dispositivo eletrônico agrícola que pode transmitir dados para a plataforma AgroNET Services 3, foi criada uma interface para manter as informações destes. Para tanto, dados como descrição, número de série (o qual é utilizado para identificação do dispositivo no sistema, e portanto deve ser único) e o tipo do dispositivo podem ser inseridas.

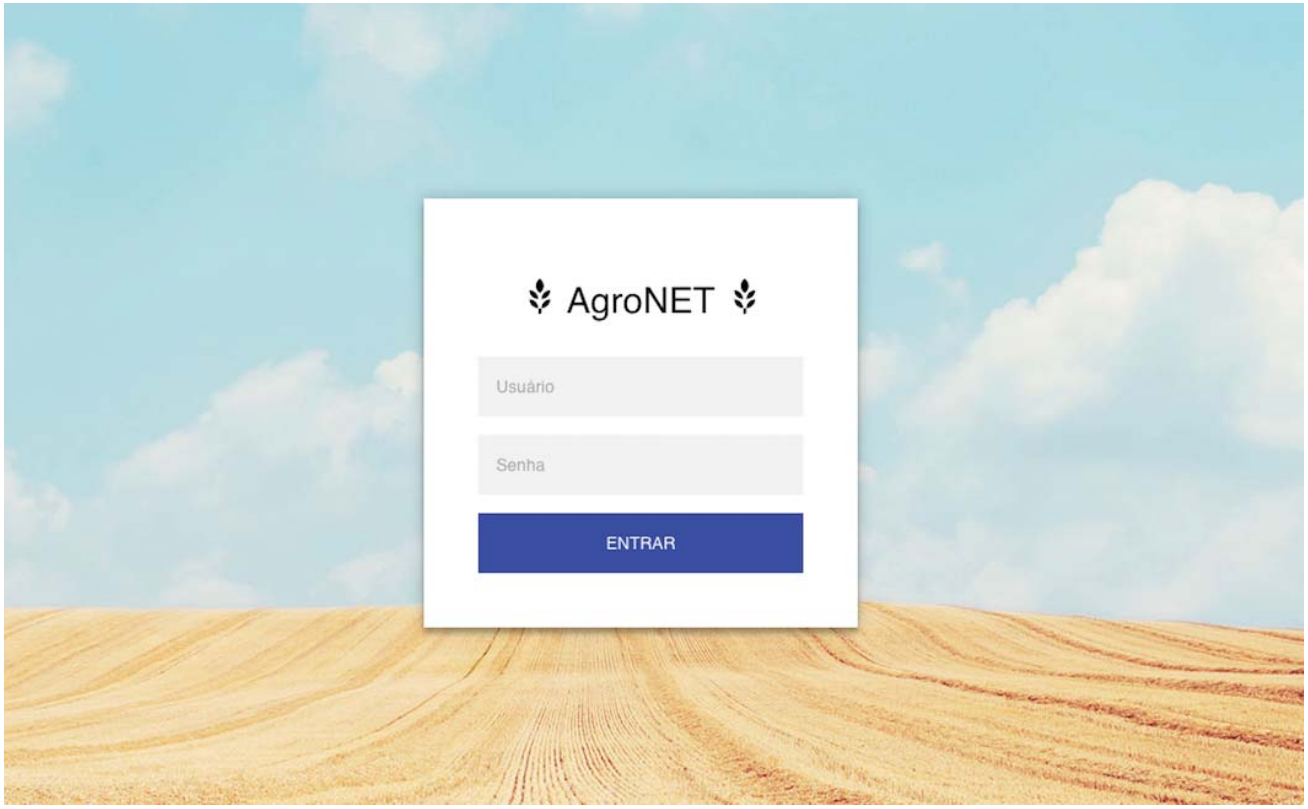

Figura 18. Tela Inicial do Projeto AgroNET-View

5.4.1.3 Visualização dos dados dos dispositivos

Para a visualização dos dados coletados pelos dispositivos, o sistema mostra um mapa com marcadores que indicam a localização geográfica de cada dispositivo, por meio dos dados de GPS que foram recebidos.

Ao clicar sobre um marcador é possível visualizar os dados do dispositivo que estão sendo recebidos pela plataforma. A Figura 19 apresenta o registro de dados de um dispositivo selecionado. Para tanto, são exibidas as informações do último registro de dados do dispositivo selecionado, porém, por meio da seleção de um intervalo de datas, é possível visualizar os demais registros do dispositivo. Após a seleção do intervalo de datas desejado, um mapa é apresentado, indicando todos os registros de dados dos dispositivos para a data selecionada. Assim, como apresentado na Figura 20, é possível visualizar todas as informações das coletas.

5.4.1.4 Maquinário

Nesta área do sistema é cadastrada a frota agrícola na qual a fazenda possuí. São cadastrados os veículos agrícolas, tais como tratores e máquinas colheitadeiras e implementos agrícolas, como sameadoras e pulverizadores. A frota pode ser cadastrada, informando atributos como: descrição, tipo e dispositivos vinculados ao veículo ou maquinário.

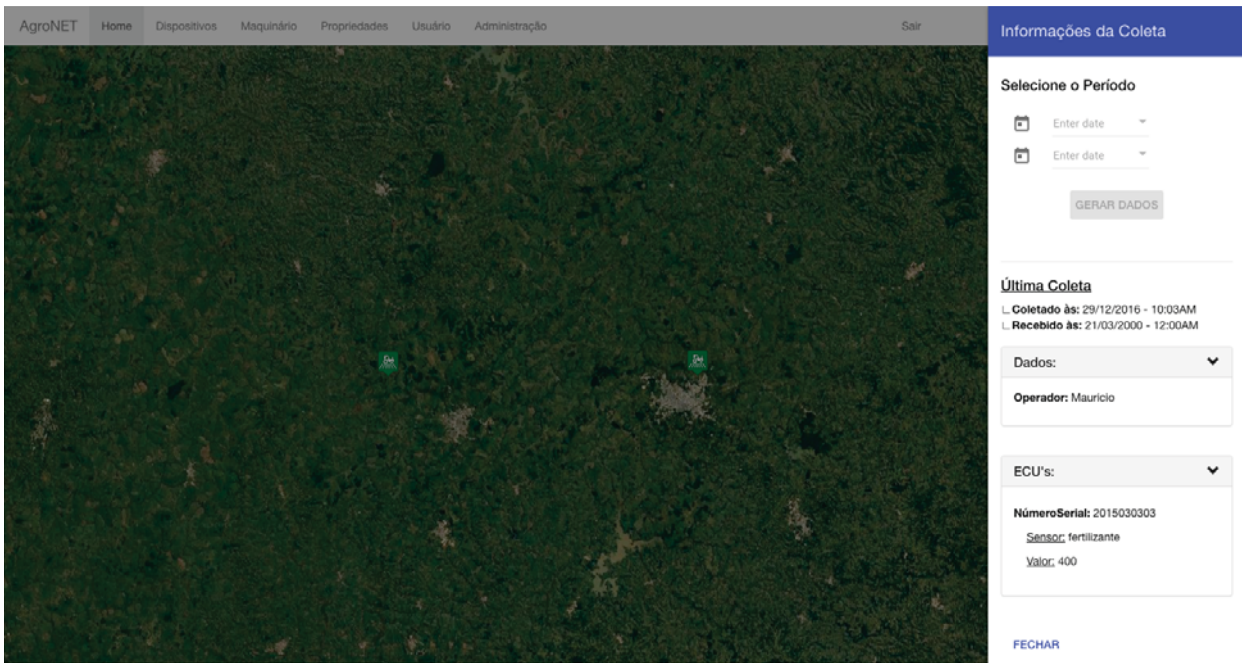

Figura 19. Exemplo da visualização dos dados dos dispositivos pela plataforma AgroNET View

Figura 20. Visualização dos dados coletados por dispositivos agrícolas

5.4.1.5 Propriedades

Nesta etapa do sistema, o usuário pode fazer o cadastro de uma propriedade, fornecendo seus dados e os dados de localização da mesma. É dentro da interface "propriedades"que o usuário poderá cadastrar novos talhões da propriedade e safras, além de fazer o controle das mesmas.

Após o cadastro da propriedade propriamente dita, ela passa a ser listada na tela junto a opção de visualizar os detalhes da propriedade, a qual redireciona o usuário para uma janela que lista suas informações. Nesse momento, é possível visualizar os talhões dessa propriedade ou criá-los. Ao criar um novo talhão, o usuário poderá informar dados como: descrição, tamanho da área e o solo da área (previamente cadastrado na seção de Administração 5.4.1.7). Após o cadastro do talhão, ficam disponíveis ao usuário informações e acessar os registros de safras dp.

Para registrar uma safra o usuário pode informar uma descrição, a data de plantio e a cultivare utilizada, também podendo acrescentar a data de colheita e a produtividade colhida. Ainda, nesta seção, o usuário poderá informar aplicações de produtos e ocorrência de patologias que observou-se durante a safra. Para cadastrar uma aplicação, o usuário informa uma descrição, a quantidade de produto a ser aplicada, qual o produto para a aplicação, a data de início e a data final, e, por fim, para cadastrar uma ocorrência de patologia o usuário poderá vincular uma patologia, previamente cadastrada no sistema na seção de Administração 5.4.1.7.

5.4.1.6 Usuário

Na tela de cadastro de usuários, é possível criar e gerenciar usuários e tipos de usuários, é aqui que são criados os usuários para acesso ao sistema. O tipo de usuário define as permissões que o usuário terá dentro do sistema e é atrelado ao usuário na hora de sua criação.

Para criar um usuário é necessário informar campos como: Nome, Sobrenome, Telefone, E-mail, Endereço, Cidade, Estado, País, Login, Senha e o seu "Tipo de Usuário", onde esta última determina os privilégios que o novo usuário terá no sistema.

5.4.1.7 Administração

Para manter as informações de cadastros gerais do sistema, sendo estes: patologias, produtos químicos, informações de solo e variedades de cultivare, determinou-se uma área administrativa para a interface do sistema. Estas informações são utilizadas na seção de gerenciamento de safras das propriedades, onde podem ser registradas, por exemplo, aplicações de produtos químicos e cultivare plantada na safra.

5.4.2 Simulação de dados da Agricultura de Precisão - AgroNET Sim

Para a disponibilização de dados da plataforma, foi desenvolvido um sistema que simula a geração de dados de sensores e dispositivos eletrônicos agrícolas. Este software permite a configuração de diferentes parâmetros para que a simulação seja feita, sendo eles: modo de operação, data, operador da máquina, identificação do dispositivo de geração de dados (número serial) e velocidade da simulação.

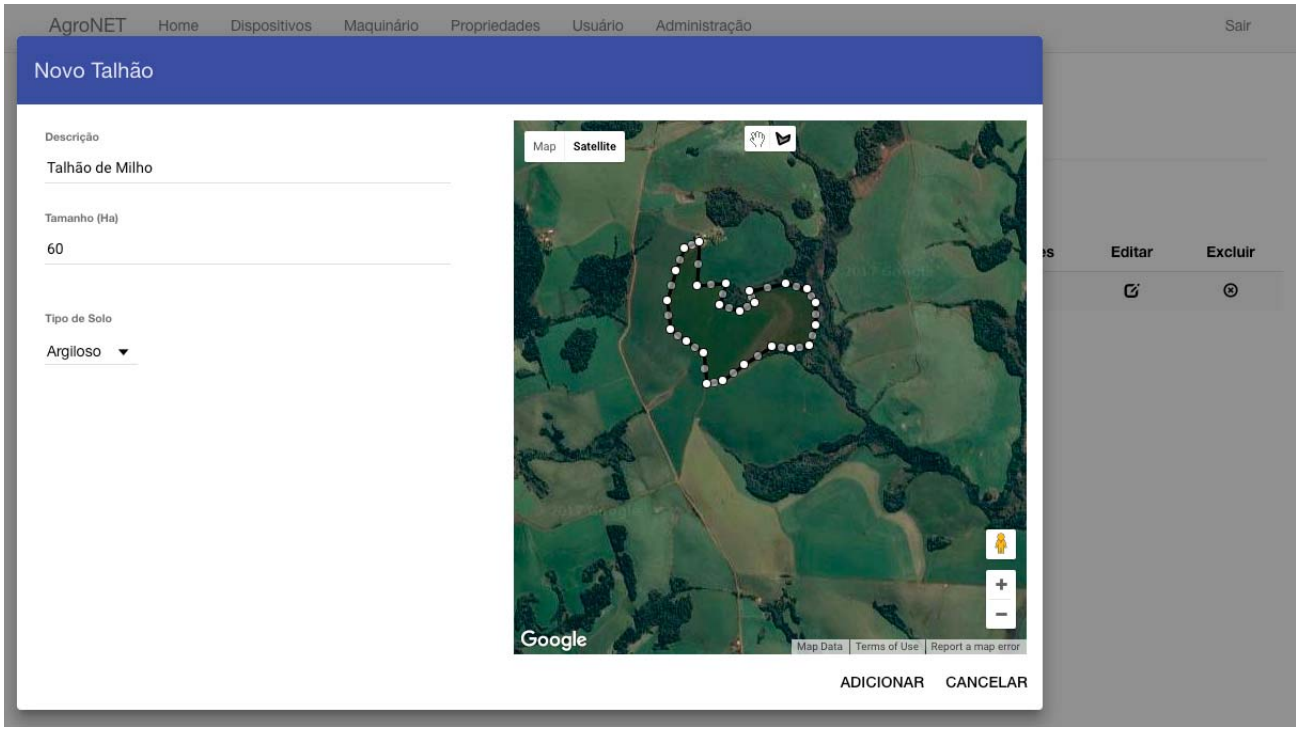

Figura 21. Tela de cadastro de propriedades/talhões da plataforma AgroNET View

Para a definição da área geográfica na qual o sistema realizará a operação, utilizou-se a biblioteca angular-google-maps [91], que oferece recursos de integração com o mapa da Google por meio do AngularJS, onde um equipamento agrícola virtual (trator, colheitadeira, etc.) realiza a interação, utilizando dados de posicionamento geográfico (latitude, longitude) e realizando a simulação de variação de variáveis como velocidade, estado da máquina (parada, em operação), tipos de sensores (fertilizante, semente) e valores gerados pelos sensores.

No instante em que a opção “Iniciar operação” é selecionada, utilizando o desenho de uma linha vermelha, o software realiza a marcação do caminho de onde o equipamento agrícola virtual deve percorrer. Para tanto, ao decorrer do trajeto traçado, o desenho de um retângulo é utilizado para realizar a marcação por onde a máquina virtual já passou. A Figura 22 apresenta a interface do simulador.

Para a transmissão dos dados, é definida uma distância em metros, para qual a máquina deverá fazer o envio dos dados coletados ao banco. Assim, por exemplo, se definido o valor 20, o sistema fará a transmissão dos dados a cada 20 metros percorridos. Os dados são transmitidos, utilizando o padrão REST, à um servidor *Web*, apresentado no capítulo 3 deste trabalho, onde os dados são armazenados.

5.5 CONCLUSÃO

O desenvolvimento de software *web* para a área da Agricultura de Precisão, pode facilitar a gestão dos dados que fazem parte desta área. Assim, o desenvolvimento de forma a possibilitar a

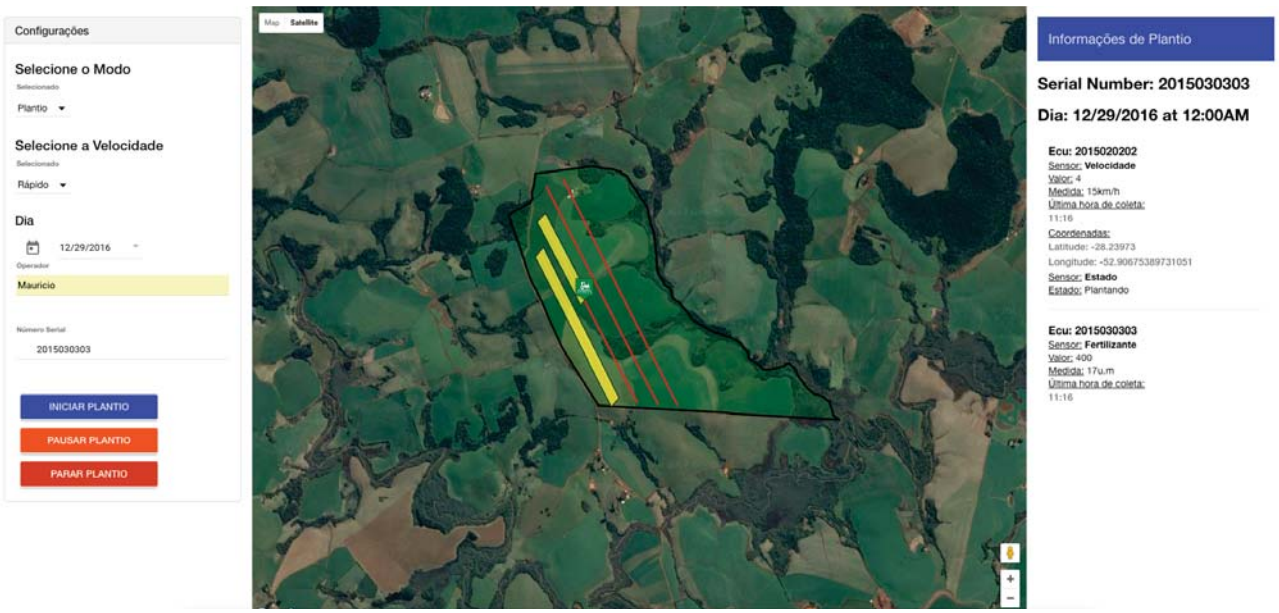

Figura 22. Interface do simulador AgroNET Sim

expansão de funcionalidades e a integração com outros sistema pode ser uma alternativa viável para as soluções de AP.

Para a elaboração do projeto apresentado neste capítulo, buscou-se utilizar tecnologias que possibilitassem a fácil modificação e expansão dos recursos disponíveis no projeto. Para tanto, o uso de *frameworks* de interface de desenvolvimento de software *web front-end* possibilitou o desenvolvimento do software a fins de torná-lo expansível, possibilitando a integração com novos recursos que podem ser futuramente desenvolvidos.

Dessa forma, o objetivo principal do trabalho, de desenvolver uma plataforma de interface gráfica, visando a integração e a troca de dados utilizando serviços *web* por meio do padrão arquitetural REST foi alcançado.

6. CONCLUSÃO

A tecnologia empregada no cenário da Agricultura de Precisão, mostra-se como aliada no sucesso da produção agrícola. Para tanto, é visto a necessidade de tecnologias que que atendam o mercado, auxiliando a produção de alimentos quantitativa e qualitativamente. Este trabalho encontra-se inserido também neste cenário, visto que apresenta-se como um exemplo de solução que engloba a Agricultura de Precisão.

Neste sentido, podemos concluir que o objetivo do trabalho foi alcançado. Como resultado do trabalho, obteve-se uma plataforma de armazenamento e disponibilização de dados da Agricultura de Precisão utilizando um banco de dados flexível, capaz de armazenar grandes volumes de dados juntamente com sua heterogeneidade.

A plataforma AgroNET GIS, implementada para armazenamento de dados geoespaciais, agrega características inexistentes na plataforma AgroNET Services. Portanto as duas plataformas podem operar em conjunto, visto que foi criadas condições de relacionamento entre as plataformas.

O objetivo de determinar um padrão para troca de mensagens por meio de serviços *web* utilizando uma API também foi atingido. Com isso, é visto que tanto dispositivos da Agricultura de Precisão podem ser conectados a plataforma, bem como outros sistemas podem fazer o uso dos dados armazenados. Para exemplificar isto, a plataforma AgroNET View foi desenvolvida, provendo uma interface gráfica para o acesso aos dados das plataformas AgroNET Services e AgroNET GIS.

7. TRABALHOS FUTUROS

Na plataforma AgroNET Services, entende-se que ocorre a necessidade de implementação de um meio de não apenas recebimento, mas também transmissão de dados da plataforma para os dispositivos da Agricultura de Precisão.

Além disso, durante o desenvolvimento do trabalho constatou-se a necessidade de receber dados de dispositivos utilizando conectividade GPRS. Dessa forma, percebe-se que o desenvolvimento de um módulo capaz de efetuar troca de mensagens GPRS pode facilitar a troca de mensagens de dispositivos com a plataforma AgroNET Services.

Entende-se que a plataforma AgroNET GIS pode ser aprimorado. Durante o desenvolvimento do trabalho, percebeu-se que o acesso a dados do tipo raster tem sua performance comprometida quando o tamanho do arquivo raster é relativamente grande. Para tanto, estuda-se a implementação de técnicas de divisão do raster por meio de *tiles*, o que permite particionar um arquivo raster em inúmeras partes menores. Além disso, a implementação dos métodos REST POST, PUT e DELETE na API da plataforma deveriam ser incluídos.

A plataforma AgroNET View, foi desenvolvida apenas com o intuito de criar uma interface gráfica para exemplificar o acesso as plataformas AgroNET Services e AgroNET GIS. Portanto, também seria de suma importância para este trabalho a melhoria desta interface.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] SØRENSEN, C. et al. Information sources and decision making on precision farming. In: AMERICAN SOCIETY OF AGRONOMY. *Proceedings of the 6th International Conference on Precision Agriculture and Other Precision Resources Management, Minneapolis, MN, USA, 14-17 July, 2002.* [S.l.], 2003. p. 1763–1775.

[2] MCBRATNEY BRETT WHELAN, T. A. J. B. A. Future directions of precision agriculture. *Precision agriculture*, Springer, v. 6, n. 1, p. 7–23, 2005.

[3] WOLFERT, S. et al. Big data in smart farming–a review. *Agricultural Systems*, Elsevier, v. 153, p. 69–80, 2017.

[4] ZHANG, N.; WANG, M.; WANG, N. Precision agriculture—a worldwide overview. *Computers and electronics in agriculture*, Elsevier, v. 36, n. 2, p. 113–132, 2002.

[5] MURAKAMI, E. *Uma infra-estrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão.* Tese (Doutorado) — Universidade de São Paulo, 2006.

[6] LI, T. et al. A storage solution for massive iot data based on nosql. In: IEEE. *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. [S.l.], 2012. p. 50–57.

[7] ROBERT, P. C. Precision agriculture: a challenge for crop nutrition management. *Plant and soil*, Springer, v. 247, n. 1, p. 143–149, 2002.

[8] SILVA, P. A. V. Banco de dados orientado a objetos: Um estudo comparativo. 2004.

[9] ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados: Fundamentos e Aplicações. 3ª Edição*. [S.l.]: LTC, 2002.

[10] MEDEIROS, L. F. de. *Banco de Dados: Princípios e Práticas*. [S.l.]: Intersaberes, 2013. 185 p.

[11] PUGA, S.; FRANçA, E.; GOYA, M. *Banco de dados: Implementação em SQL, PL/SQL e Oracle 11g*. [S.l.]: Pearson, 2014. 352 p.

[12] HEUSER, C. A. *Projeto de banco de dados*. [S.l.]: Sagra Luzzatto, 2001. 254 p.

[13] ELMASRI, R.; NAVATHE, S. B. *Sistemas de Banco de Dados: Fundamentos e Aplicações. 4ª Edição*. [S.l.]: LTC, 2006.

[14] OLIVEIRA, M. et al. Um estudo comparativo entre banco de dados orientado a objetos, banco de dados relacionais e framework para mapeamento objeto/relacional, no contexto de uma aplicação web. *HOLOS*, Instituto Federal de Educacao Ciencia e Tecnologia do Rio Grande do Norte, v. 31, n. 1, p. 182, 2015.

[15] BONFIOLI, G. F. Banco de dados relacional e objeto-relacional: Uma comparação usando postgresql. *Departamento de Ciência da Computação, Universidade Federal de Lavras, Minas Gerais*, 2006.

[16] BISNETO, R. R. M. Uma ferramenta case para modelagem de bases de dados objeto-relacionais. 2011.

[17] SILBERSCHATZ, A. et al. *Database system concepts*. [S.l.]: McGraw-Hill Hightstown, 2010. v. 6. 1376 p.

[18] PINHEIRO, R. S. et al. Comparativo entre banco de dados orientado a objetos (bdoo) e bancos de dados objeto relacional (bdor). 2009.

[19] LAZZARETTI, A. T. *Integração de banco de dados e modelos de simulação de culturas para estimular o impacto de mudanças do clima no rendimento de grãos e na severidade da giberela em trigo*. 192 p. Tese (Doutorado) — PPGA, Passo Fundo, 2013.

[20] BOSCARIOLI, C. et al. Uma reflexao sobre banco de dados orientados a objetos. In: *Congresso de Tecnologias para Gestão de Dados e Metadados do Cone Sul, Paraná, Brasil*. [S.l.: s.n.], 2006.

[21] MONGODB. *MongoDB*. Disponível em: <https://www.mongodb.org/>. Acesso em: Jun. 15, 2015.

[22] MEMBREY, P.; PLUGGE, E.; HAWKINS, D. *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. [S.l.]: Apress, 2010.

[23] ANDERSON, J. C.; LEHNARDT, J.; SLATER, N. *CouchDB: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2010.

[24] REDMOND, E.; WILSON, J. R. *Seven databases in seven weeks: a guide to modern databases and the NoSQL movement*. [S.l.]: Pragmatic Bookshelf, 2012.

[25] HEWITT, E. *Cassandra: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2010.

[26] CASSANDRA. *Cassandra*. Disponível em: <http://planetcassandra.org/>. Acesso em: Jun. 15, 2015.

[27] WORBOYS, M. F.; DUCKHAM, M. *GIS: a computing perspective*. [S.l.]: CRC press, 2004.

[28] MIRANDA, J. I. Publicando mapas na web: Servlets, applets ou cgi? *Embrapa Informática Agropecuária*, 2003.

[29] CASTRO, A. F. de. *Sistemas computacionais espaço-temporais para tomada de decisão em questões ambientais relacionadas à indústria de petróleo e gás*. Tese (Doutorado), 2007.

[30] SOARES, V. G. *GEOVISUAL–um ambiente de consultas visuais para bancos de dados geográficos*. Tese (Doutorado) — PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, Recife, 2002.

[31] JúNIOR, J. C. X. *NatalGIS: Um Sistema Multiagente de Recomendação de Informações Geográficas baseado em Agrupamento de Dados Relacionais*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE, 2012.

[32] QUEIROZ, R. d.; CÂMARA, G. Banco de dados geográficos. *Curitiba: MundoGeo*, 2005.

[33] RIBEIRO, G. P. *BANCOS DE DADOS DE IMAGENS DE SATÉLITES ARTIFICIAIS*. 37 p. Tese (Doutorado) — COPPE - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, 1995.

[34] CÂMARA, G. *Modelos, linguagens e arquiteturas para bancos de dados geográficos*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1995.

[35] LOURENÇO, P. M. B. Um estudo sobre recursos de tratamento de dados espaciais em sgbds geográficos. *Curso de Especialização em Geoprocessamento. Departamento de Cartografia do Instituto de Geociências da Universidade Federal de Minas Gerais*, 2008.

[36] FERREIRA, C. do S. *PostGIS*. [S.l.], 2006. 57 p.

[37] RICARDO, J.; PINTO, C. *PostGIS - Spatial Database Extension for PostgreSQL*. [S.l.], 2010. 22 p.

[38] TANAKA, P. S. Implementação de extensão de método de acesso para indexação de dados espaço-temporais no postgresql. *Universidade Estadual de Londrina*, 2013.

[39] POSTGIS. *PostGIS 2.1.8dev Manual*. Disponível em: <http://postgis.net/docs/manual-2.1/>. Acesso em: Jun. 15, 2015.

[40] OBE, R.; HSU, L. *PostGIS in action*. [S.l.]: Manning Publications Co., 2011.

[41] THOMPSON, M. *Getting Started with GEO, CouchDB, and Node. js*. [S.l.]: "O'Reilly Media, Inc.", 2011.

[42] POURABBAS, E. *Geographical Information Systems: Trends and Technologies*. Taylor & Francis, 2014. (A science publishers book). ISBN 9781466596931. Disponível em: <https://books.google.com.br/books?id=dcuSAwAAQBAJ>.

[43] EMER, J. C. Migrando aplicações web para plataformas abertas: um estudo de caso. 2013.

[44] VICTORINO, M.; BRÄSCHER, M. Organização da informação e do conhecimento, engenharia de software e arquitetura orientada a serviços: uma abordagem holística para o desenvolvimento de sistemas de informação computadorizados. *DataGramaZero–Revista de Ciência da Informação*, v. 10, n. 3, 2009.

[45] BROWN, A.; JOHNSTON, S.; KELLY, K. Using service-oriented architecture and component-based development to build web service applications. *Rational Software Corporation*, 2002.

[46] ALVES, F. V. de F. UtilizaÇÃo de web services para integraÇÃo de sistemas. *FATEC-SP Faculdade de Tecnologia de São Paulo*, 2012.

[47] RODRIGUEZ, A. Restful web services: The basics. *IBM developerWorks*, 2008.

[48] CHO, H.; RYU, S. Rest to javascript for better client-side development. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the companion publication of the 23rd international conference on World wide web companion*. [S.l.], 2014. p. 937–942.

[49] JUNIOR, F. d. A. R. *Programação Orientada a Eventos no lado do servidor utilizando Node. js*. 2012.

[50] HERRON, D. *Node Web Development*. [S.l.]: Packt Publishing Ltd, 2013.

[51] TORSTENSSON, D.; ELOFF, E. An investigation into the applicability of node. js as a platform forweb services. 2012.

[52] TYLER, L.; MORRIS, M. Choosing a restful framework for the rapdb. 2014.

[53] MONGOOSE. http://mongoosejs.com/index.html. Accessed: 2015-10-15.

[54] SEVILLEJA, C. *Easily Develop Node.js and MongoDB Apps with Mongoose*. 2015. Disponível em: <https://scotch.io/tutorials/using-mongoosejs-in-node-js-and-mongodb-applications>. Acesso em: Nov. 26, 2016.

[55] RYAN. *Simplifying PostgreSQL queries in Node.js using promises*. 2015. Disponível em: <http://blog.tomnod.com/nodejs-database-queries>. Acesso em: Out. 15, 2015.

[56] PG. Disponível em: <https://www.npmjs.com/package/pg>. Acesso em: Out.15, 2015.

[57] DEHAAN, P. *Connecting to a PostgreSQL database from Node.js using the pg module*. 2012. Disponível em: <http://nodeexamples.com/2012/09/21/connecting-to-a-postgresql-database-from-node-js-using-the-pg-module/>. Acesso em: Out. 15, 2015.

[58] HAHN, E. M. *Express in Action: Writing, building, and testing Node.js applications*. [S.l.]: "O'Reilly Media, Inc.", 2016.

[59] EXPRESSJS. *Express Framework web rápido, flexível e minimalista para Node.js*. Disponível em: <http://expressjs.com/>. Acesso em: Dez. 10, 2016.

[60] BROWN, E. *Web Development with Node and Express: Leveraging the JavaScript Stack*. [S.l.]: "O'Reilly Media, Inc.", 2014.

[61] HAMAD, H.; SAAD, M.; ABED, R. Performance evaluation of restful web services for mobile devices. *Int. Arab J. e-Technol.*, v. 1, n. 3, p. 72–78, 2010.

[62] PAUTASSO, C. Restful web service composition with bpel for rest. *Data & Knowledge Engineering*, Elsevier, v. 68, n. 9, p. 851–866, 2009.

[63] MONTEIRO, F. *Node. js 6. x Blueprints*. [S.l.]: Packt, 2016.

[64] JACYNTHO, M. D.; SCHWABE, D.; ROSSI, G. A software architecture for structuring complex web applications. *J. Web Eng.*, v. 1, n. 1, p. 37–60, 2002.

[65] XIANG, P.; HOU, R.; ZHOU, Z. Cache and consistency in nosql. In: IEEE. *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*. [S.l.], 2010. v. 6, p. 117–120.

[66] NEWCOMER, E.; LOMOW, G. *Understanding SOA with web services (independent technology guides)*. [S.l.]: Addison-Wesley Professional, 2004.

[67] TOTVS. *Talhões*. Disponível em: <https://www.totvs.com/mktfiles/tdiportais/helponlineprotheus/portuguese/agra010_talhoes.htm>. Acesso em: Novembro, 10 2015.

[68] TIBOLA, C. S. et al. *Sistema de Rastreabilidade Digital para Trigo*. [S.l.: s.n.], 2013. 90 p.

[69] BIHIS, C. *Mastering OAuth 2.0*. [S.l.]: Packt Publishing Ltd, 2015.

[70] BOYD, R. *Getting started with OAuth 2.0*. [S.l.]: Inc. O'Reilly Media, 2012.

[71] ALBA, J. F. Trabalhos acadêmicos de geoprocessamento desenvolvidos no laboratório de planejamento ambiental em 2007. *Embrapa Clima Temperado. Documentos*, Pelotas: Embrapa Clima Temperado., 2007.

[72] CHIECHELSKI, G. O.; BOGORNY, V. Uma extensão do postgis para geração automática de trajetórias semânticas. *Instituto de Informática da UFRGS, Porto Alegre, Brasil. Trabalho de conclusão de curso*, p. 61, 2008.

[73] LINSEISEN, H. Development of a precision farming information system. In: CITESEER. *Proceedings of the Third European Conference on Precision Agriculture, Montpellier*. [S.l.], 2001. p. 689–694.

[74] NIKKILÄ, R.; SEILONEN, I.; KOSKINEN, K. Software architecture for farm management information systems in precision agriculture. *Computers and electronics in agriculture*, Elsevier, v. 70, n. 2, p. 328–336, 2010.

[75] QUADROS, F. Metadados geoespaciais. *FossGis Brasil*, v. 4, n. 1, p. 4, 2012.

[76] OBE, R. O.; HSU, L. S. *PostGIS in action*. [S.l.]: Manning Publications Co., 2015.

[77] RESOURCES, D. of N.; MINES. *What is a Shapefile?* [S.l.], 2016. 8 p. Disponível em: <https://tools.ietf.org/html/rfc7946>. Acesso em: Janeiro 19, 2017.

[78] ESRI. *Shapefile Technical Description*. 1998. 34 p. Disponível em: <https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf>. Acesso em: Janeiro 19, 2017.

[79] RESOURCES, D. of N.; MINES. *Raster Graphic*. [S.l.]. Disponível em: <https://techterms.com/definition/rastergraphic>. Acesso em: Janeiro 09, 2017.

[80] ARCMAP. *How raster data is stored and managed*. [S.l.]. Disponível em: <https://techterms.com/definition/rastergraphic>. Acesso em: Janeiro 19, 2017.

[81] RFC 7946 - The GeoJSON Format. https://tools.ietf.org/html/rfc7946. Acesso em: Novembro 09, 2016.

[82] GEOJSON.ORG. *GeoJSON*. http://geojson.org/. Acesso em: Novembro 09, 2016.

[83] NASA. *MODIS - Moderate Resolution Imaging Spectroradiometer*. https://modis.gsfc.nasa.gov/about/. Acesso em: Novembro 11, 2016.

[84] POSTGIS. *Manual do PostGIS 2.0*. Disponível em: <http://postgis.org/docs/>. Acesso em: Novembro 11, 2016.

[85] ANDRADE, A. *Usabilidade de interfaces web: avaliação heurística no jornalismo on-line*. [S.l.]: E-papers, 2007.

[86] MARTINEZ, M. L. Usabilidade no design gráfico de web sites. In: *III Congresso Internacional de Engenharia Gráfica nas Artes e no Desenho*. [S.l.: s.n.], 2000.

[87] SILVA, M. S. *Bootstraep 3.3.5. Aprenda a usar o framework Bootstrap para criar layouts CSS complexos e responsivos*. [S.l.]: Novatec, 2013.

[88] ANGULAR Material. Disponível em: <https://material.angularjs.org>. Acesso em: Março 25, 2016.

[89] MEW, K. *Aprendendo Material Design: Domine o Material Design e crie interfaces bonitas e animadas para aplicativos móveis e web*. [S.l.]: Novatec, 2016.

[90] COCHRAN, D. *Twitter Bootstrap Web Development How-To*. Birmingham: Packt, 2012. 68 p.

[91] MAPS, A. G. *Angular Google Maps*. [S.l.]. Disponível em: <https://angular-ui.github.io/angular-google-maps/>. Acesso em: Março 25, 2016.