UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIAS COMPUTACIONAIS
PARA O AGRONEGÓCIO - PPGTCA

ERMINIO PITA JASSE

APPLICATION PROGRAMMING INTERFACE - API PARA INTEGRAÇÃO DE DADOS EM AGRICULTURA DE PRECISÃO

DISSERTAÇÃO

MEDIANEIRA-PR
2017

ERMINIO PITA JASSE

APPLICATION PROGRAMMING INTERFACE - API PARA INTEGRAÇÃO DE DADOS EM AGRICULTURA DE PRECISÃO

DISSERTAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio – PPGTCA – da Universidade Tecnológica Federal do Paraná – UTFPR – Campus Medianeira, como requisito parcial para obtenção do título de Mestre em Tecnologias Computacionais para o Agronegócio / Área de concentração: Tecnologias Computacionais Aplicadas à Produção Agrícola.

Orientador: Prof. Dr. Claudio L. Bazzi.
Co-Orientador: Prof Dr. Paulo Sérgio Graziano Magalhães

MEDIANEIRA

2017

Dados Internacionais de Catalogação na Publicação

J39a

Jasse, Ermínio Pita

Application programming interface - API para integração de dados em agricultura de precisão/Ermínio Pita Jasse–2017.
45f. : il. ; 30 cm.

Orientador: Claudio Leones Bazzi.
Coorientador: Paulo Sergio Graziano Magalhães.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio. Medianeira, 2017.
Inclui bibliografias.

1.Integração de dados (Computação). 2.Agricultura de precisão. 3.Tecnologias Computacionais- Dissertações. I.Bazzi, Claudio Leones,orient. II.Magalhães, Paulo Sergio Graziano, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio. IV. Título.

CDD: 004

Biblioteca Câmpus Medianeira
Marci Lucia Nicodem Fischborn 9/1219

TERMO DE APROVAÇÃO

APPLICATION PROGRAMMING INTERFACE - API PARA INTEGRAÇÃO DE DADOS EM AGRICULTURA DE PRECISÃO

Por

ERMINIO PITA JASSE

Essa dissertação foi apresentada às oito horas e trinta minutos, do dia treze de dezembro de dois mil e dezessete, como requisito parcial para a obtenção do título de Mestre em Tecnologias Computacionais para o Agronegócio, Linha de Pesquisa Tecnologias Computacionais Aplicadas à Produção Agrícola, no Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio - PPGTCA, da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Claudio Leones Bazzi (Orientador – PPGTCA)

Prof. Dr. Paulo Lopes de Menezes (Membro Interno – PPGTCA)

Dr. Luis Henrique Bassoi (Membro Externo – EMBRAPA INSTRUMENTAÇÃO)

A via original com as assinaturas encontra-se na secretaria do programa.

AGRADECIMENTOS

Um agradecimento especial vai para o meu orientador Prof. Dr. Claudio L. Bazzi, por ter acreditado em mim e me ter oferecido uma oportunidade mesmo sem razões para o efeito; por me acompanhar durante toda formação; pelo apoio incondicional.
Agradeço ao meu co-orientador Prof. Paulo Sérgio Graziano Magalhães pelo apoio durante a caminhada e pelos sábios ensinamentos.
Ao Ministério da Ciência e Tecnologia Ensino Superior e Técnico-Profissional pela bolsa e pelo apoio durante a formação.
A minha instituição Universidade Eduardo Mondlane (UEM), Moçambique, por me permitir continuar com a formação e por me conceder suporte. Agradeço aos meus colegas de trabalho no DMI (UEM – Faculdade de Ciências)
A minha mãe Serenia Fuleza Malapuza, por ser minha professora para a vida, e pelo exemplo que sempre foi e será, meu eterno agradecimento.
A minha esposa Omega Armando Mourinho pelo acompanhamento nos momentos difíceis, aos meus filhos, irmãos e familiares.
A meus amigos Alexssander Liesenfeld, João Metambo e Márcio Matté.
Aos professores e servidores da UTFPR, em particular aos guardas por me ter entendido mesmo quando ficava até tarde no laboratório.
A todos não mencionados, mas que de alguma forma tenha dado seu contributo.

"Aquele que não é corajoso o suficiente para correr riscos não conseguirá nada na vida."

Muhamad Ali

"Não é porque o cachorro está latindo que o elefante pára de caminhar."

Fungai Jasse

RESUMO

JASSE, Ermínio Pita. **Application Programming Interface - API para integração de dados em agricultura de precisão.** 2017. Dissertação (Mestrado em Tecnologias Computacionais para o Agronegócio) - Programa de Pós-Graduação em Tecnologias Computacionais para o Agronegócio, Universidade Tecnológica Federal do Paraná.

Dada a importância do setor agrícola na economia nacional, e com a crescente necessidade de aumento da produtividade das culturas, o desenvolvimento de tecnologias que possibilitem a integração de aplicações computacionais aplicadas ao segmento agrícola se faz necessário, considerando a necessidade de obtenção de informações de forma rápida e segura. A Agricultura de Precisão (AP) fundamenta-se na aplicação de tecnologias para melhorar a tomada de decisão, principalmente em relação a aplicação de insumos e ao manejo dos recursos existentes. O presente estudo teve como objetivo o desenvolvimento de uma ferramenta computacional em ambiente Web que permite realizar o armazenamento, a integração e o gerenciamento de dados agrícolas por meio de softwares especialistas integrados. A ferramenta desenvolvida visa fornecer uma interface para que outros softwares enviem e recebam dados agrícolas criando uma integração entre diversas aplicações por meio do ambiente Internet e requisições/respostas. A ferramenta se mostrou eficiente, oferecendo vantagens quanto ao menor tempo de desenvolvimento de aplicações e na integração destas que podem ser desenvolvidas em diferentes linguagens de programação e para funcionar em diferentes ambientes.

Palavras-chave: Tecnologia agrícola, Web API, Integração de dados agrícolas.

ABSTRACT

JASSE, Ermínio Pita. **Application Programming Interface - API for data integration in Precision Agriculture.** 2017. Dissertation (MSc in Computational Technologies for Agribusiness) – Post - Graduate Program in Computational Technologies for Agribusiness, Federal Technological University of Paraná.

Given the importance of the agricultural sector in the national economy, and with the growing need to increase productivity, the development of technologies that allow the integration of computational applications applied to the agricultural environment is necessary, considering the need to obtain information quickly and safely. The Precision Agriculture (AP) is based on the application of technologies for better decision-making, mainly concerning the application of agricultural inputs and management of the existing resources. The present study aimed at the development of a computational tool in a Web environment that allows the storage, integration and management of agricultural data through integrated specialist software. Through the Internet environment and HTTP requests / responses, the tool aims to provide an interface for other software to send and receive agricultural data creating an integration between several applications. The tool has proven to be efficient, offering several advantages in terms of the shortest development time of applications and the integration of them, which can be developed in different programming languages and for different environments.

Key-words: Agricultural technology, Web API, Agricultural data integration.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AgDataBox	Agricultural Data Box
AgDataBox API	API Agricultural Data Box
AP	Agricultura de Precisão
API	*Application Programming Interface*
APP	Aplicação ou aplicativo
CACHEABLE	Dados que podem ser temporariamente guardados
EAI	*Enterprise Application Interface*
GPS	*Global Positioning System*
GNSS	Global Navigation Satellite System
HTTP	*Hypertext Transfer Protocol*
JSON	*JSON Web Token*
JWT	*Java Script Object Notation*
PC	*Personal Computer*
REST	*Representational State Transfer*
SGBD	Sistema de Gestão de Bancos de Dados
SIG	Sistemas de Informações Geográficas
SQL	*Structured Query Language*
SOAP	*Simple Object Access Protocol*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
VANT	Veículo Aéreo Não Tripulado
WebApp	*Web Application* (aplicativo web)
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A procura por produtos agrícolas tende a aumentar nas próximas décadas, prevendo-se que a população mundial tende a superar 9 bilhões de pessoas até o 2050 (ALEXANDRATOS e BRUINSMA, 2012). Este fato faz com que se busque formas de melhorar e ampliar os sistemas produtivos, considerando problemas socioeconômicos e ambientais, que também tendem a aumentar, devido a redução da disponibilidade de água e de áreas produtoras (TILMAN *et al*., 2011).

A Agricultura de Precisão (AP) é vista como uma pratica otimizada de gerenciamento que visa proporcionar o aumento da produção, a redução dos custos e diminuição dos impactos ambientais causados pela prática agrícola (ANOUNCIA e AMALANATHAN, 2017; JAROLÍMEK *et al*., 2017; CAMBOURIS *et al*. 2014). Atrelado a esta forma de produção inteligente, está o uso de tecnologias, tais como o uso de sistemas de localização GNSS, sistemas de informação geográfica (SIG), técnicas de sensoriamento remoto, computação móvel e outros (STAFFORD, 2000).

Para Shen, Basist e Howard (2010), países emergentes precisam aplicar os princípios da AP para garantir seu desenvolvimento sustentável, tendo em vista que permite que agricultores identifiquem problemas e oportunidades e apliquem soluções com precisão (LINDBLOM *et al*., 2016), o que pode garantir uma agricultura sustentável tanto em termos financeiros como ambientais.

O objetivo da AP é gerenciar a variabilidade espacial e temporal do solo e das culturas, por meio da tomada de decisões mais acertadas (FOUNTAS *et al*., 2015; POLOJÄRVI *et al*., 2009), tendo-se disponibilidade de dados espaciais de qualidade e precisão. Para tal, o produtor precisa conhecer as características de cada parte da propriedade como o tipo de solo, sua fertilidade, relevo, vegetação, o histórico de uso, entre outros, para que possa tratar cada parte da área de produção de forma diferenciada, conforme suas necessidades.

Devido a sua difusão nos últimos anos, as tecnologias têm sido usadas com uma frequência cada vez maior no campo, para obtenção de dados agrícolas e monitoramento das lavouras, resultando em um excessivo volume de dados de diferentes tipos (FOUNTAS, PENDERSEN e BLACKMORE, 2005), o que dificulta seu gerenciamento e utilização. Autores como Steinberger, Rothmund e

Auernhammer (2008) ou Nash, Korduan e Bill (2009) comentam que o uso efetivo de dados requer que tanto os dados gerados internamente (dispositivos, equipamentos e softwares da propriedade) como os gerados externamente (ex. dados meteorológicos) sejam facilmente integrados e partilhados entre diferentes hardwares, softwares e sistemas de informação.

Com vista a melhorar o gerenciamento, diversos softwares têm sido usados para tratar os dados coletados de forma a agrega-los valor para posterior uso no processo de tomada de decisões na produção agrícola (BAZZI, SOUSA e BETZEK, 2015), e arquiteturas cada vez mais sofisticadas têm sido propostas, fazendo uso de plataformas web e outras tecnologias. Softwares de gestão agrícola desenvolvidos com serviços baseados na web facilitam a pesquisa colaborativa pela Internet por conectar dados geograficamente dispersos (SCHWEIK, 2009).

Uma API[1] (*Application Programming Interface*) pode auxiliar em problemas de integração e padronização de formatos para o compartilhamento de dados no setor agrícola, considerando que uma API possibilita que duas aplicações ou mais troquem dados entre si (GUILLAUD, 2011).

Neste contexto, o presente trabalho baseia-se no desenvolvimento de uma Web API para armazenamento, integração, gerenciamento e disponibilização de dados agrícolas para que possa ser utilizada para o desenvolvimento de aplicações voltadas a AP, possibilitando que desenvolvedores de aplicações deste segmento desenvolvam aplicações que funcionem de forma integrada e compartilhando dados em formatos simples e flexíveis.

[1] API é um conjunto de aplicativos que tem como principal função intermediar a comunicação entre dois ou mais outros aplicativos. Tem sido usada como uma janela para as organizações exporem seus recursos ou serviços para seus clientes e parceiros.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver e disponibilizar uma Web API que possa ser utilizada em Agricultura de Precisão permitindo a integração de dados entre aplicações, tendo como foco o armazenamento, gerenciamento e disponibilização de dados agrícolas.

2.2 OBJETIVOS ESPECÍFICOS

- Modelar uma estrutura para a integração de diferentes tipos de dados, obtidos em ambiente agrícola;
- Desenvolver uma API de armazenamento e gerenciamento de dados agrícolas, que permita integrar diferentes softwares voltados ao setor agrícola;
- Testar e disponibilizar a API em um servidor baseado na web.

3 REVISÃO DE LITERATURA

3.1 AGRICULTURA DE PRECISÃO

Agricultura de Precisão (AP) corresponde a uma estratégia de gestão que utiliza tecnologias de informação para trazer dados de múltiplas fontes e tomar decisões associadas à produção agrícola (NATIONAL RESEARCH COUNCIL, 1997). Para Molin (2008), AP é um conceito moderno de gestão agrícola utilizando técnicas digitais para monitorar e otimizar os processos de produção. Para Fountas, Pedersen e Blackmore (2005), o objetivo da AP é o gerenciamento do solo e das culturas para aumentar a produtividade e reduzir os impactos ambientais, sendo que a coleta, o armazenamento, o processamento de dados e posterior aplicação de insumos em taxa variável acabam figurando como suas principais atividades.

Alguns estudos (NATIONAL RESEARCH COUNCIL, 1997; BLACKMORE e GRIEPENTROG, 2002) apontam que os avanços tecnológicos têm permitido aos produtores agrícolas coletar um grande volume dados sobre seus talhões e, já é sabido que gerenciar um volume elevado de dados é sempre um desafio, principalmente quando não se tem as ferramentas necessárias.

De modo geral, para se praticar a AP com sucesso, tanto na perspectiva econômica como ambiental, considera-se dentre outros componentes, a necessidade de um sistema que armazene os dados de campo, que permita realizar os processos de análise e tomada de decisão, e que permita a aplicação das estratégias traçadas a partir das análises obtidas com o processamento dos dados (MOLIN, 2008).

A atividade agrícola está relacionada a um sistema complexo e dinâmico, composto de inúmeras variáveis que podem influenciar o desempenho produtivo das culturas e sua qualidade. Neste sentido, com vistas a se obter um melhor controle e maior segurança produtiva, dados de diversas fontes e de diferentes estruturas de dados têm sido coletados de diferentes formas a fim de avaliar a influência de cada variável no contexto da produção. Esses dados normalmente são obtidos a partir de amostragem de solo ou plantas e posterior análise laboratorial

(RODRIGUES, CORA e FERNANDES, 2012; SANTI *et al*., 2012) ou por meio de sensores, tais como medidores de condutividade elétrica, sensores de índices de vegetação (SOARES e CUNHA, 2015) e monitores de colheita (MOORE, 1997), que têm surgido de forma expressiva nos últimos anos, considerando o barateamento de dispositivos eletrônicos e a facilidade no desenvolvimento de tecnologias utilizando-se destes recursos.

Valente *et al*. (2011) demonstra resultados positivos com o uso de sensores de umidade em áreas irrigadas para avaliar necessidades específicas de irrigação em cada área específica das lavouras. Isso demonstra a importância que os sensores têm tido na produção agrícola.

Outra técnica de obtenção de dados utilizado na agricultura corresponde ao sensoriamento remoto, e que corresponde ao processo de obtenção de dados de campo, sem haver um contato direto com tais objetos (WÓJTOWICZ, WÓJTOWICZ e PIEKARCZYK, 2016). Essa técnica tem sido aplicada no acoplamento de sensores em satélites e veículos aéreos não tripulados - VANTs.

Desta forma, verifica-se que diversas fontes de dados, obtidas de distintas formas estão presentes no campo, objetivando a interpretação para se ter uma tomada de decisão mais acertada.

Em um estudo realizado nos EUA e Dinamarca, Fountas *et al.* (2005) analisou a forma como os produtores armazenam e tratam seus dados. O estudo constatou que a maioria dos produtores guardam dados de produtividade em computadores pessoais, enquanto que dados de análise de solo são armazenados tanto em seus computadores como nos computadores dos consultores agrícolas, que fizeram a amostragem.

Neste sentido, verifica-se a necessidade de se ter um ambiente que tenha a capacidade de integrar diversas aplicações de forma fácil. Desta forma, o produtor deixaria de se preocupar com a forma ou técnica de processamento e passaria mais tempo analisando os dados coletados, podendo ainda, compartilhar seus dados com outros produtores, realizar o gerenciamento eficiente e realizar análises mais precisas.

Fountas *et al*. (2005), concluiu que os agricultores investem pouco tempo para analisar os dados coletados, por demandar muito tempo, considerado muito elevado pela maioria dos agricultores envolvidos na pesquisa. O estudo apela então, para

que em futuros trabalhos, seja dado foco em uma forma melhor e mais fácil de armazenamento e tratamento do elevado volume de dados, como forma de reduzir o tempo requerido para sua análise.

3.2 TECNOLOGIAS PARA DESENVOLVIMENTO DE SOFTWARE

Como forma de se manterem competitivas, as organizações têm investido cada vez mais em tecnologias de diferentes tipos, tendo aplicativos que rodam em diferentes níveis do negócio da instituição e que podem ter sido desenvolvidos usando ferramentas diversificadas. As organizações possuem sistemas que são baseados geralmente em tecnologias modernas, fazendo uso de bancos de dados e aplicativos desenvolvidos em diferentes linguagens de programação. De forma a integrar diferentes tecnologias, várias técnicas de integração têm sido desenvolvidas, tais como a Enterprise Application Interface (EAI), que corresponde a combinação de processos, softwares, padrões e tecnologias, com o objetivo de integrar diferentes aplicações, de modo que possam atuar de forma transparente, como se fossem apenas uma única aplicação (FENNER, 2015).

Para Soomro e Awan (2012), a integração de aplicativos pode ser feita em quatro níveis de integração: (1) em nível de dados; (2) em nível de interface aplicacional; (3) em nível de métodos; e (4) em nível de interface com o usuário.

Independentemente do nível de integração, decidido em função dos objetivos e caraterísticas do projeto, qualquer integração de aplicativos deve considerar o tipo de plataforma a ser utilizada (desktop ou web), considerando seus pontos positivos e negativos.

Sheriff (2002) chama atenção para o fato de a plataforma web centralizar o gerenciamento, facilitando a manutenção e atualização da aplicação. Para uma plataforma que deve atender a vários usuários, especialmente quando eles se encontram geograficamente dispersos, a plataforma web é mais adequada, considerando que a partir de qualquer dispositivo computacional conectado a internet, um usuário pode acessar a aplicação, armazenar, gerenciar, analisar e disponibilizar dados e recursos. Já o desktop tem a vantagem de ser considerado

mais seguro por corresponder a um ambiente mais restrito.

Em um estudo comparativo entre aplicativos web e desktop, Pop (2000) realizou uma comparação empírica entre dois aplicativos, sendo um desktop e outro em ambiente web. Ainda que em termos de performance as vantagens tenham sido a favor das aplicações desktop, o autor concluiu apontando alguns pontos chave, que dão conta para uma tendência maior no desenvolvimento de aplicações web. Um aplicativo desktop precisa ser descarregado e instalado no computador do cliente antes de ser utilizado. Esta visão também é mencionada por outros autores (SHERIFF, 2002; ELIZABETH, 2015), que destacam a facilidade em convencer um usuário a escolher uma aplicação que não precisa ser instalada em detrimento de uma que para ser usada precisa ser instalada e configurada. Ainda, considerando um ambiente geograficamente disperso, a plataforma desktop torna-se inviável para qualquer projeto.

3.2.1 API - *APPLICATION PROGRAMMING INTERFACE*

API é o acrônimo de *Application Programming Interface* (Interface de Programação de Aplicativos), e segundo Jacobson, Brail e Woods (2012), corresponde a um ou a um conjunto de softwares com a função de intermediar a comunicação entre aplicações. Uma API permite que uma ou mais aplicações se comuniquem de forma transparente para o usuário, como se fosse só uma aplicação. Sua integração agrega a integração em nível de dados onde esses são enviados ou recebidos entre as aplicações envolvidas (ex. facebook ou tweeter), por meio de uma API. A consulta de taxas de câmbio ou previsão de tempo é outro exemplo de uso de API do dia-a-dia.

Para uma organização, uma API é um ponto de conexão entre ela e seus parceiros e clientes, partilhando processos de negócio, serviços e conteúdo; conectando equipes de trabalho internas a desenvolvedores independentes, de uma forma fácil e segura (RICHARDSON e RUBY, 2007).

Uma API pode ser desenvolvida com o propósito de ser usada por um grupo restrito de usuários (desenvolvedores) ou pode ser desenvolvida tendo como alvo o

público (qualquer desenvolvedor). Assim, as APIs podem ser classificadas em públicas ou privadas (DOERRFELD, 2016), cabendo a organização decidir o tipo de API que pretende, com base nos seus objetivos.

No desenvolvimento dessas soluções (Web APIs), dois estilos arquiteturais têm dominado o mercado nos últimos anos: 1) REST (*Representation State Transfer*) e 2) SOAP (*Simple Object Access Protocol*). REST corresponde a um estilo arquitetural para sistemas de hipermídia distribuídos (FIELDING, 2000a), ele não é um protocolo. Já o SOAP corresponde a um protocolo para troca de dados geralmente usando HTTP como protocolo de transferência de dados. Ainda que SOAP tenha sido a abordagem mais usada no desenvolvimento de serviços para web por muito tempo, o estilo REST possui representatividade atualmente de mais de 70% das APIs públicas (BORA e BEZBORUAH, 2015) disponíveis.

Antes de se iniciar com o desenvolvimento de uma API, é importante avaliar os dois estilos arquiteturais mais utilizados (REST e SOAP). No entanto, é importante notar que o próprio significado de SOAP deixa claro a intenção de ser um protocolo enquanto que “REST é definido apenas como um estilo arquitetural” (FIELDING, 2000a), o que faz com que não seja adequado estabelecer uma comparação direta entre os dois estilos.

Para realizar a comunicação entre as diversas aplicações e uma API, verifica-se a necessidade de se ter padrões de comunicação e a definição do formato no qual os dados serão transmitidos. Atualmente existem dois formatos largamente utilizados para transportar e apresentar dados pela Internet: 1) Formato JSON (*Java Script Object Notation*); e 2) Formato XML (*eXtensible Markup Language*).

O formato JSON possui características de ser disponibilizado com tamanho menor que o XML devido ao maior número de *tags* (caracteres especiais de formação) utilizadas ao escrever um arquivo XML, sendo que o formato JSON tem sido o formato de apresentação de dados escolhido na maioria das APIs desenvolvidas.

4 MATERIAIS E MÉTODOS

Devido à complexidade envolvida no desenvolvimento da API, buscou-se fazer uso de tecnologias já consagradas na literatura, atendendo ao quesito referente a gratuidade, tendo em vista que a API será distribuída gratuitamente aos seus usuários (desenvolvedores de software).

4.1 LINGUAGEM DE PROGRAMAÇÃO

Devido a ampla abrangência técnica apresentada pela literatura, a linguagem Java foi selecionada, considerando que é uma linguagem de alto nível e que possui uma grande quantidade de usuários, facilitando a busca por auxílios quando necessário (SCHILDT, 2007). Suas caraterísticas de atender aos conceitos de orientação a objetos, ser multiplataforma, atendimento a critérios de segurança, e performance (TUTORIALSPOINT, 2017), foram considerações importantes a favor no momento da avaliação.

A linguagem Java possui ainda diversos *frameworks*, tais como o VRaptor, Struts e Spring MVC, que correspondem a conjuntos de classes desenvolvidas com a finalidade de auxiliar e acelerar a programação (codificação) de aplicativos baseados na web. Hibernate, Spring Data e Castor são outros exemplos de *frameworks* da linguagem Java, usados para realizar a persistência (armazenamento/recuperação) de dados de forma fácil, simples e segura.

Como servidor de aplicações e web container, foi utilizado o Tomcat, versão 8.3, considerando a recomendação de McClanahan e Maucherat (2016), que é o servidor web indicado para aplicações web robustas desenvolvidas em Java.

Java possui uma diversidade de opções de IDE *(Interface Development Environment*) como o NetBeans, IntelliJ IDEA ou Eclipse. Para o projeto foi usado o IDE NetBeans pela disponibilidade de *plug-ins* (aplicativos complementares) disponibilizados pelo desenvolvedor da IDE e facilidade de integração com outras ferramentas de desenvolvimento.

4.2 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

O Sistema de Gerenciamento de Banco de Dados escolhido foi o PostgreSQL que é um sistema gratuito e que possui mecanismos de extensibilidade para incorporar capacidades adicionais de forma a torná-lo mais flexível, principalmente as funcionalidades espaciais, incorporadas pela extensão PostGIS (OLIVEIRA *et al*., 2007). O PostgreSQL possui interfaces de programação nativas, para linguagens como Java, C/ C ++, .Net, Perl, Python e Ruby, permitindo a programação de funcionalidades no próprio banco de dados (POSTGRESQL, 2016).

4.3 ESTILO ARQUITETURAL E FORMATO DE DADOS

Tendo em vista a comunicação entre os sistemas a serem desenvolvidos e a API, foi adotado o padrão de dados no formato JSON para o transporte e apresentação de dados entre clientes (aplicações diversas) e a API. O uso do formato JSON foi considerado como mais adequado, considerando seu formato simples (fácil de se entender) e por ser menor, quando comparado com outros formatos como o padrão XML.

Como estilo arquitetural, o desenvolvimento foi realizado fazendo uso da estrutura REST, por possuir boa performance na utilização de recursos do protocolo HTTP (FIELDING, 2000b), que é o principal protocolo a ser utilizado no funcionamento da API desenvolvida.

4.4 SEGURANÇA

Tendo em vista o gerenciamento de dados em nuvem, a disponibilização de dados e recursos deve se manter restrita e acessível apenas por intervalos de tempo limitados, em que há a troca de informações; o ambiente foi desenvolvido considerando recursos de autenticação e autorização, fazendo com que apenas usuários devidamente credenciados tenham acesso aos dados sobre os quais possuam permissão. Esse processo foi baseado no uso de um *token* que é gerado no ato de efetivação do *login* do usuário e permanece válido por até 24 horas.

Para a geração e verificação do *token* foi utilizada a tecnologia JWT (JSON Web Token), fazendo uso do protocolo OAuth (protocolo aberto que serve para autorização segura de uma forma simples e padronizada para aplicações web, mobile ou desktop (OAUTH, 2017)) como protocolo de autorização. JWT corresponde a um padrão aberto (definido pelo RFC 7519) que define uma forma compacta e autocontida para transmitir de forma segura, dados entre duas partes como objeto JSON (JWT, 2017).

Uma vantagem importante em usar esta tecnologia é o fato de já contar com ferramentas para criptografar os dados, sendo que os dados mais críticos já são, por padrão criptografados antes de serem enviados. Havendo necessidade, a ferramenta permite que mais dados sejam criptografados.

4.5 DOCUMENTAÇÃO

Devido a importância de se ter uma documentação adequada para disponibilização para a comunidade de desenvolvedores usuários da API, foi utilizada a ferramenta Swagger, que corresponde a um conjunto de aplicativos *open-source* desenvolvido com base na OpenAPI - Specification (SWAGGER, 2017), para desenvolver a documentação. OpenAPI -Specification define uma linguagem de programação padrão para descrição de interface para APIs REST, que permite tanto

humanos como máquinas localizar e entender recursos/serviços sem precisar acessar o código fonte.

4.6 CONTEXTO DO PROJETO AGRICULTURAL DATA BOX

Agricultural Data Box (ou AgDataBox) corresponde a um projeto que vem sendo desenvolvido por pesquisadores da Universidade Tecnológica Federal do Paraná (UTFPR) – Medianeira, em parceria com a Universidade Estadual do Oeste do Paraná e apoio do Ministério da Agricultura, Pecuária e Abastecimento - MAPA, Capes, CNPq e Fundação Parque Tecnológico Itaipu (PTI). O projeto tem sido desenvolvido com objetivo de desenvolver e disponibilizar ferramentas computacionais gratuitas para produtores rurais, pesquisadores e prestadores de serviço, focados em agricultura de precisão (AP), na tentativa de viabilizar o ramo agrícola no país por meio de tecnologias livres.

De forma sucinta, o projeto AgDataBox visa o desenvolvimento de um ambiente integrado de dados, obtidos de diferentes fontes e com diferentes tipos de dados (mapas temáticos, dados amostrais de solo, relevo, imagens georreferenciadas, precipitação, operações de campo, custos, pessoal, máquinas/equipamentos, entre outros). O projeto possui vários aplicativos voltados para o setor agrícola, dos quais constam AgDataBox Mobile (versão para dispositivos móveis), AgDataBox Web (versão web) que funcionam como clientes, e AgDataBox API que corresponde ao servidor das demais aplicações.

O presente trabalho consistiu no desenvolvimento do AgDataBox API, que é o software que corresponde a uma plataforma de armazenamento dos dados que os softwares clientes irão enviar e permitir a comunicação entre os softwares desenvolvidos na base dele (AgDataBox API). A ferramenta permitie integrar diferentes aplicações em um único local de forma centralizada, auxiliando e facilitando o trabalho de desenvolvedores de software que podem usufruir de recursos complexos implementados na API, de forma facilitada.

A estrutura básica da API AgDataBox (servidor), integra diferentes softwares, desenvolvidos em diferentes linguagens, conectadas ao servidor por meio de uma

estrutura de rede compatível e que permite a comunicação pelo protocolo HTTP, Figura 1.

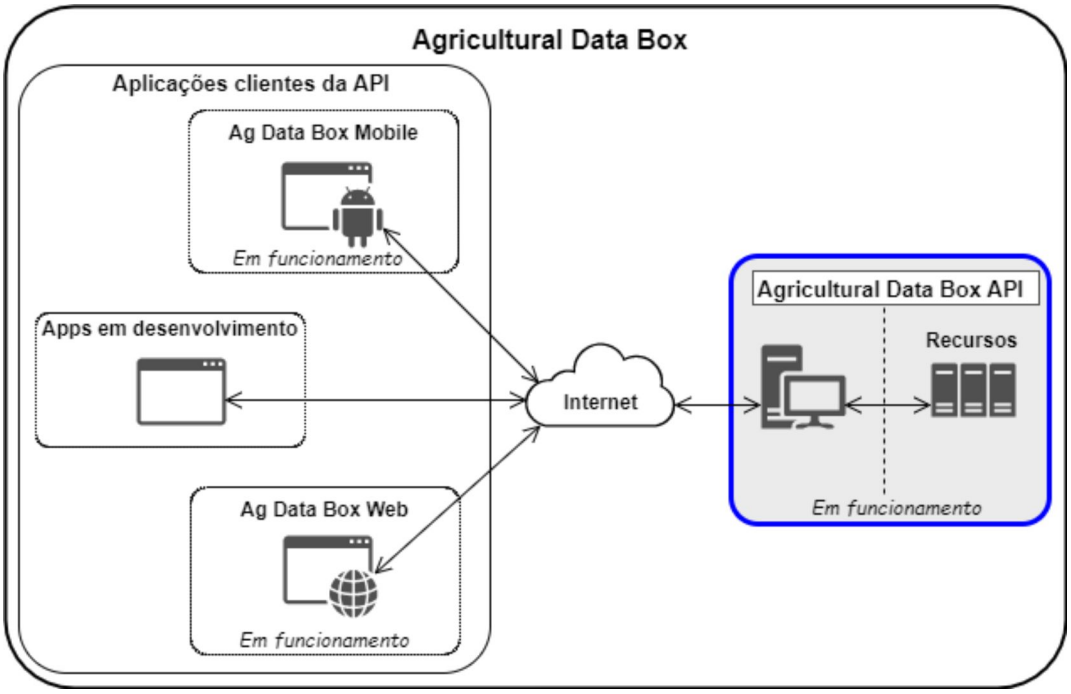

Figura 1 – Estrutura de funcionamento básico da plataforma AgDataBox-API, com diferentes aplicações conectadas

Fonte: Autoria própria.

Desta forma, a estrutura criada permite, além de outras características, viabilizar:

1. Integração de dados – em que diversas aplicações podem trabalhar de forma simultânea, armazenando, gerenciando e disponibilizando dados;
2. Centralização do gerenciamento e manutenção – em que dados e regras de negócio passam a estar apenas na API, que são disponibilizadas a outras aplicações. As aplicações clientes, por exemplo, não precisam se preocupar com as regras de negócio e nem questões relacionadas ao banco de dados central;
3. Manutenção e gerenciamento de recursos – em que se tem um ambiente facilitado para o controle de falhas por ser centralizado, permitindo que correções sejam realizadas e serviços corrigidos sejam disponibilizados à diferentes aplicações de forma simultânea e rápida;

4. Agilidade no processo de desenvolvimento de aplicações cliente – em que se percebe que o desenvolvimento de aplicações é viabilizado de forma mais rápida, devido a possibilidade de uso de recursos já implementados na API;
5. Escalabilidade – em que acréscimo de novas aplicações no projeto, pode ser feito, sem, no entanto, ter que realizar grandes alterações na estrutura geral;
6. Multiplataforma – cada desenvolvedor fica livre para trabalhar com ferramentas/linguagens que possui maior conhecimento/viabilidade.

4.7 ESTRUTURA LÓGICA DA API

O desenvolvimento da API foi realizado considerando três camadas, transparentes para o usuário, sendo: 1) camada de recursos (dados); 2) camada da lógica de negócio; e 3) camada de persistência.

A camada de recursos permite o armazenamento de todos os dados previstos, correspondendo aos registros dos produtores agrícolas ao longo dos anos, em suas áreas produtoras. Encontra-se nesta camada o banco de dados, gerenciado por um sistema de gerenciamento de bancos de dados (PostgreSQL).

A camada Lógica de negócio define todas as regras para acesso aos dados armazenados. A verificação de autenticidade de um usuário bem como as autorizações que ele possui são algumas das tarefas fundamentais desta camada. A Figura 2 ilustra a estrutura da API, contemplando seus principais componentes em nível técnico.

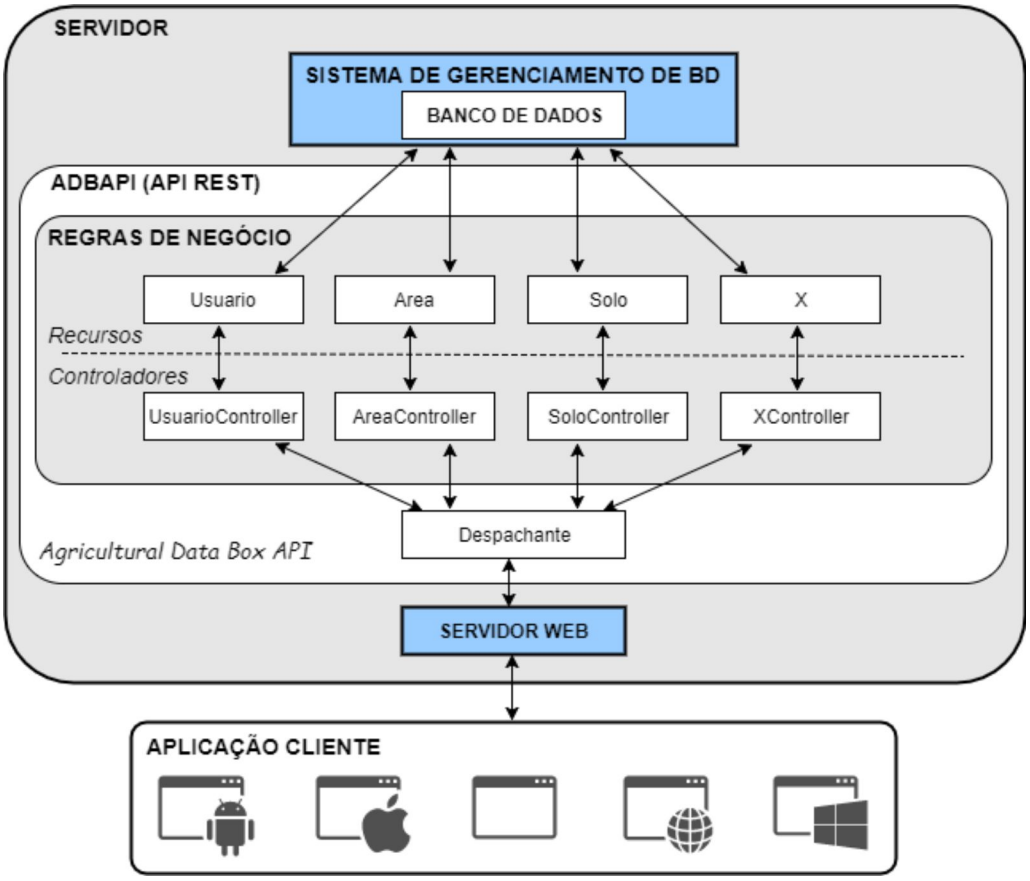

Figura 2 – Estrutura técnica geral da API.

Fonte: Autoria própria.

O despachante (*dispatcher*) é o módulo responsável por receber e analisar as requisições HTTP que os clientes enviam, e reencaminha cada requisição para o controlador apropriado, dependendo da URI para a qual a requisição foi enviada. Assim, se uma requisição é enviada para /api/solo/32, o despachante reencaminhará essa requisição para o despachante SoloController, que é o controlador do recurso de solo. Feito isso, aguardará que uma resposta seja retornada pelo controlador para que uma resposta HTTP seja construída e retornada para o cliente.

Os Controladores (*controllers*), são módulos específicos para cada recurso disponível na aplicação. São responsáveis por receber a requisição encaminhada pelo despachante e verificar se a solicitação não contém erros. Se algum erro for encontrado, uma mensagem de erro HTTP correspondente é retornada para o despachante, caso contrário, o controlador assume o fluxo e a requisição é atendida, retornando uma confirmação ao despachante que, por sua vez, retorna a resposta correspondente ao requisitante (cliente).

Os recursos (Usuario, Solo, entre outros), são módulos específicos que representam cada modelo do sistema. São chamados pelos controladores e interagem com a lógica de negócio e executam ações com a camada de armazenamento de dados. Os dados recuperados do banco, são formatados para o formato JSON antes de serem retornados para o controlador.

5 RESULTADOS E DISCUSSÃO

5.1 ARQUITETURA DE FUNCIONAMENTO DO SISTEMA

Tendo em vista a complexidade do ambiente desenvolvido, foram criados alguns diagramas que visam orientar e facilitar desenvolvedores de novas aplicações, assim como a manutenção do próprio software desenvolvido. Dos diagramas desenvolvidos, constam os da UML (Unified Modeling Language), que segundo Schmuller (2004), é uma linguagem que define uma série de artefatos para auxiliar na tarefa de modelar e documentar sistemas desenvolvidos com no paradigma orientado a objetos. Para ilustrar o funcionamento geral do sistema, o diagrama de requisições/ respostas (arquitetura cliente - servidor) da API (incluindo a autenticação) é apresentado (Figura 3), em que é possível visualizar a arquitetura cliente-servidor, onde o cliente realiza requisições e o servidor encaminha uma resposta à chamada. O cliente, corresponde a qualquer aplicação que utiliza recursos da API por meio de requisições HTTP.

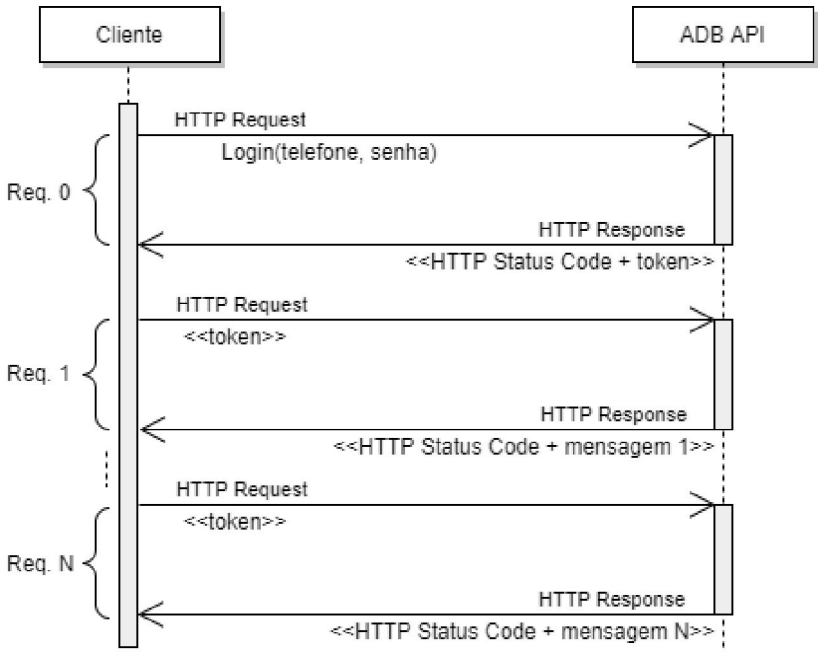

Figura 3 – Esquema de requisições/ respostas cliente-servidor da API.
Fonte: Autoria própria.

Na Figura 3 é possível verificar que na requisição 0, o cliente envia uma requisição HTTP com suas credencias para a API e ela responde retornando um *token* que deverá ser enviado para efetuar qualquer outra operação na API. Assim que a API recebe a requisição para *login*, ela verifica as credenciais do usuário. No caso de a informação ser válida, o *token* é encaminhado ao cliente. Caso contrário uma mensagem de erro é retornada para o cliente.

Após realizado o *login*, novas requisições podem ser realizadas. Na Figura 3 pode-se verificar na requisição 1, uma requisição é realizada anexando-se o *token* obtido na requisição 0 (de *login*), juntamente com o tipo de operação a ser realizada. Ao receber a requisição, a API, verifica se o *token* é válido e então encaminha a resposta ao cliente.

Tendo em vista que a API corresponde a uma estrutura para atendimento multiusuário, recursos de armazenamento de dados foram apresentados com duas características distintas: 1) dados do tipo compartilhado; 2) dados com restrições de compartilhamento. Os dados compartilhados correspondem aos cadastros de dados que ao serem cadastrados podem ser de uso de outros usuários de forma compartilhada, restringindo inclusão de cadastros iguais no banco de dados. Um exemplo de tipo de dados compartilhado corresponde a “Cultura”, em que um registro (“Soja” por exemplo) poderá ser utilizado por qualquer usuário que trabalha com esta cultura. Na Tabela 1, são apresentados todos os recursos de armazenamento disponíveis, separados de acordo com sua característica.

Tabela 1. Classificação de Entidades quanto a característica

Fonte: Autoria própria.

Compartilhados	Não compartilhados		
CategoriaNoticia	AbrangenciaNoticia	GeometriaOcorrencia	PixelAmostra
Cultura	AgendaOperacao	GradeAmostral	PixelMapa
Empresa	Amostra	InsumoOperacao	PixelZonaManejo
Insumo	Area	Mapa	PontoAmostral
Marca	Atributo	MapaZonaManejo	Precipitacao
Solo	Classe	Maquina	Safra
TipoOperacao	Classificacao	MaquinaOperacao	SafraArea
UnidadeMedida	Entidade	Noticia	Semente
Variedade	Entrega	Ocorrencia	TipoOcorrencia
Veiculo	Funcionario	OperacaoCampo	Usuario
	FuncionarioOperacao	Permissao	ZonaManejo

De forma a organizar os recursos disponibilizados pela API, desenvolveu-se uma estrutura de chamadas, com uma interface uniforme para todos os recursos (entidades) do sistema, similar a apresentada na Figura 4, sendo que, o cliente não necessita alterar o endereço das chamadas (*endpoint*), considerando que basta o mesmo realizar a escolha do método (POST - Inserção, PUT - Atualização, GET - Seleção, DELETE - Deleção) e informar o recurso que deseja.

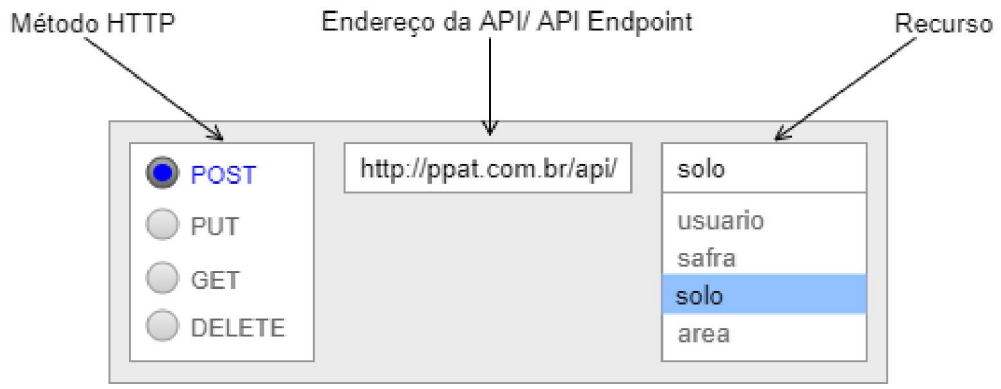

Figura 4 – Interface uniforme REST, para chamadas na API.

Fonte: Autoria própria.

5.2 MODELAGEM DO SISTEMA

Devido a similaridade nas operações sobre os diversos recursos necessários na API, na Figura 5 é apresentado um diagrama de casos de uso integrado e que contempla a interação entre atores (aplicações cliente) e a API.

No diagrama apresentado na Figura 5, pode-se verificar ações que podem ser executadas pelo usuário. Dentre as operações previstas estão a criação de usuário, efetuar *login* e criar, alterar, recuperar e remover recursos (dados), permitindo toda a manipulação do banco de dados disponível e gerenciado pela API.

Tendo em vista a interação entre objetos do sistema, foram criados diagramas de sequência de eventos, visando ilustrar a dinâmica da interação entre objetos, baseado no tempo (SCHMULLER, 2004).

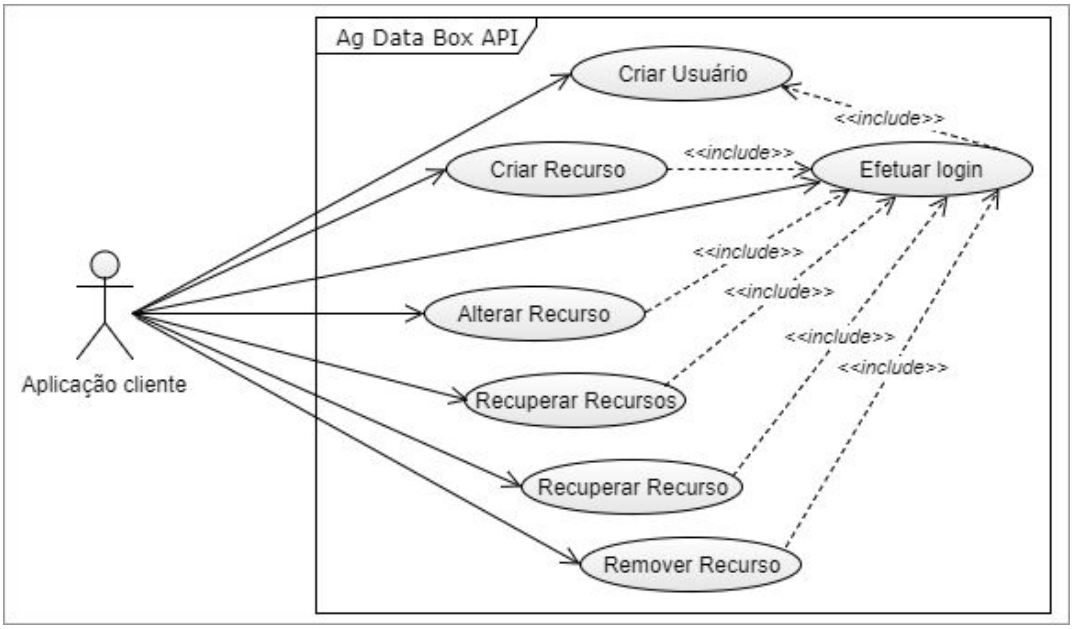

Figura 5 – Diagrama de Casos de Uso resumido.

Fonte: Autoria própria.

Para exemplificar e demonstrar a lógica utilizada para realização das ações previstas para o sistema, serão apresentados os diagramas de sequência de eventos para autenticação (*login*) (Figura 6) e para inserção de dados de tipo de solo na API (Figura 7).

Após a criação de um usuário na API, para qualquer requisição o cliente necessita efetuar *login* (autenticação) para obter o *token* necessário em todas as requisições. Para efetuar o *login,* conforme apresentado na Figura 6, o cliente envia uma requisição HTTP com método POST para a URI "/usuario/login", com dados de telefone e a respectiva senha no corpo da requisição. O despachante ao receber a requisição, faz as validações e então, encaminha a solicitação para o controlador passando os dados de *login*. O Controlador, (UsuarioController), chama o recurso correspondente e verifica se este (usuário) existe no banco de dados com as credenciais encaminhadas. Caso a validação dos dados seja verdadeira, um *token* é gerado e retornado ao cliente. Caso contrário, um erro é retornado para o Controlador que por sua vez, repassa o resultado para o Despachante que, constrói a resposta HTTP de erro e retorna para o cliente.

Na Figura 7 é apresentado o fluxo para a inserção de um dado referente ao tipo de solo. O cliente envia uma requisição HTTP com método POST para a URI

"/solo", incluindo no corpo da requisição os dados de tipo de solo e o *token* no cabeçalho. A requisição é recebida pelo Despachante o qual extrai a URI, o cabeçalho e o corpo que contém os dados do novo tipo de solo a ser cadastrado. Em seguida, a requisição é encaminhada para o Controlador apropriado (no caso SoloController), tendo em conta a URI recebida. O Controlador faz as verificações e solicita o recurso Solo, o qual corresponde aos dados de tipo de solo na base de dados. No fim de todo o processo, uma resposta de confirmação correspondente, é encaminhada para o cliente.

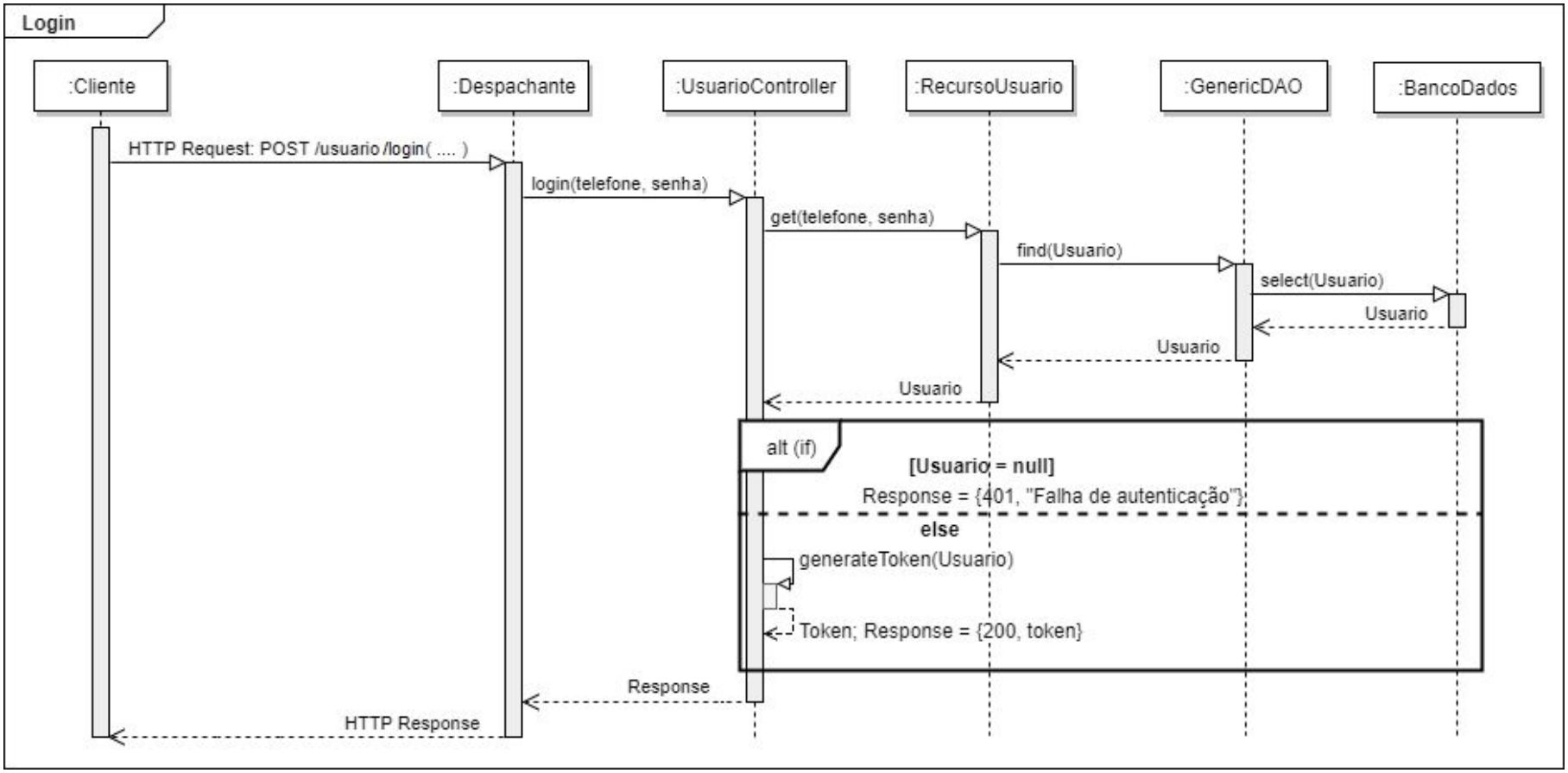

Figura 6 – Diagrama de sequência de eventos para autenticação (*login*)

Fonte: Autoria própria.

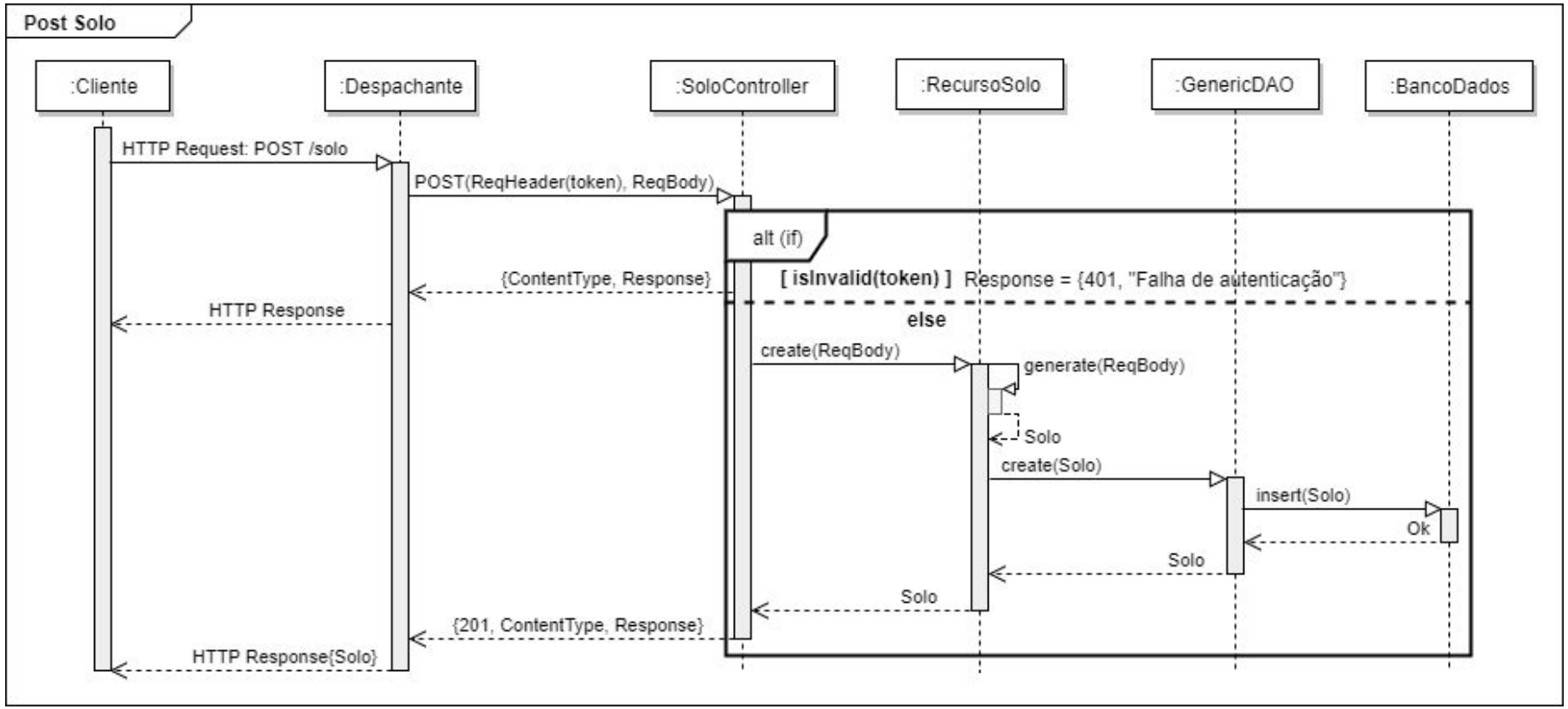

Figura 7 – Diagrama de sequência de eventos para criar solo

Fonte: Autoria própria.

5.3 SEGURANÇA

A autenticação e autorização para acesso a qualquer funcionalidade da API são realizadas por meio de um *token*, que é gerado e retornado ao usuário mediante uma requisição correspondente a operação *login*. Na Figura 8, é apresentado um esquema de como é gerado o *token* e como é realizada a autenticação e autorização na API, para cada requisição recebida.

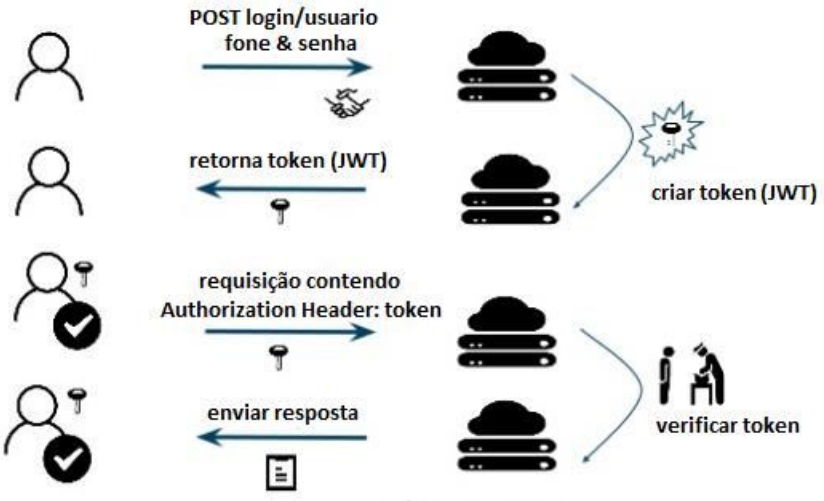

Figura 8 – Esquema de autorização e autenticação usando JWT.
Fonte: Autoria própria.

Os passos a serem seguidos pelo cliente, correspondem a:

a) O cliente já registrado na API envia uma requisição com seus dados (telefone e senha) para a API;

b) Os dados são validados na API, sendo então gerado um *token* que é retornado para o cliente.

c) De posse do *token*, o cliente pode realizar requisições na API anexando o *token* no *header* da requisição.

d) A API recebe e verifica todos os parâmetros necessários para conferir se o *token* é válido (integridade do *token*, prazo de validade, permissão, entre outros). Validado o *token* e verificada a permissão para execução de uma

determinada operação solicitada, a API executa a operação e retorna a resposta correspondente para o requisitante.

Tendo em vista a necessidade de usuários realizarem operações em nome de outros usuários (quando um usuário concede permissão para outro), a API disponibiliza duas formas de tratar as autorizações; 1) Caso um usuário conceda permissão a outro, (usuário com código 1 (autorizante) concede um outro usuário com código 2 (autorizado) autorização para atuar no seu nome), então a requisição deverá incluir no *header* (cabeçalho), um campo chamado autorizante, com o valor do código do proprietário dos dados (autorizante). Nesse caso, o autorizante necessita conceder previamente uma permissão ao autorizado, indicando os recursos que ele pode acessar e as operações que ele pretende fornecer acesso.

A ausência do campo autorizante no *header* (cabeçalho), é interpretada como não havendo necessidade de tratar permissões entre usuários, o que faz com que cada usuário tenha permissão de efetuar requisições apenas sobre seus próprios dados.

5.4 TESTES EXECUTADOS COM APLICAÇÕES CLIENTE

Um dos resultados do trabalho realizado é uma plataforma que para além de armazenar e gerenciar dados agrícolas, permite integrar aplicações de forma simples e com um padrão de comunicação fácil de utilizar. Fazendo uso de requisições/respostas HTTP, a plataforma permite que aplicativos possam compartilhar dados tendo na API, um mediador da interação.

A estrutura final da plataforma é apresentada na Figura 9, sendo que “App cliente” representa qualquer aplicativo que faça uso da plataforma. A documentação da API, também corresponde a um tipo de cliente, visto que possui uma componente para enviar requisições e receber as respetivas respostas por parte da plataforma.

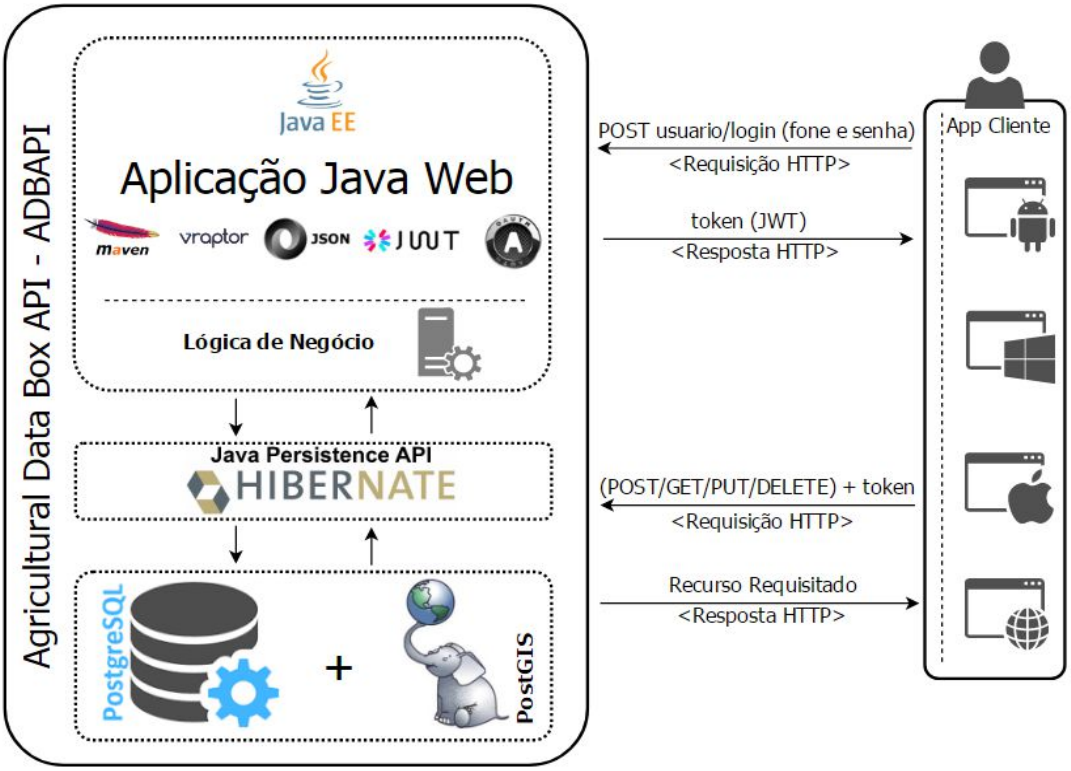

Figura 9 – Estrutura tecnológica da Web API proposta

Fonte: Autoria própria.

A solução corresponde a uma aplicação web, podendo receber e responder a requisições de qualquer plataforma de software, seja mobile, web ou até *desktop*. Ainda as aplicações cliente, podem ser desenvolvidas para diferentes sistemas operacionais, tais como Android, Windows Phone, iOS ou Symbian, se forem aplicações para dispositivos móveis; Windows, Mac Os ou Linux, para aplicações em ambiente desktop; ou fazer uso de diferentes navegadores web, tais como Mozilla Firefox, Internet Explorer ou Google Chrome, no caso de aplicações web.

Na Figura 9, também é possível identificar as principais tecnologias utilizadas para o desenvolvimento da aplicação, indicando ainda em que ambiente cada uma destas tecnologias foi utilizada. Integrando a base de dados (tradicional e dados geográficos) o banco de dados PostgreSQL e sua extensão PostGIS, tratam do armazenamento dos dados. Fazendo a integração do ambiente web com a base de dados, utilizou-se o Hibernate (um framework) e na aplicação web, utilizou-se a linguagem Java, fazendo uso das tecnologias como VRaptor e JWT.

Visando avaliar os aspectos funcionais e técnicos da aplicação, a mesma foi instalada e configurada em um ambiente público da internet (www.ppat.com.br/api). A partir deste ambiente, fazendo uso de ferramentas prontas (especializadas para testar aplicações desenvolvidas na arquitetura REST) de testes tais como Postman (Figura 10) e Insomnia (Figura 11), foram desenvolvidos testes. Essas ferramentas possuem vários recursos para facilitar o processo de teste tais como, salvar os dados, criar uma estrutura dos recursos da API, salvar os testes realizados ou importar/exportar testes de um aplicativo para outro. Para além dessas ferramentas, foram desenvolvidos três aplicativos integrados com a API (Figura 12, Figura 13 e Figura 14), que igualmente foram usados para efetuar os testes.

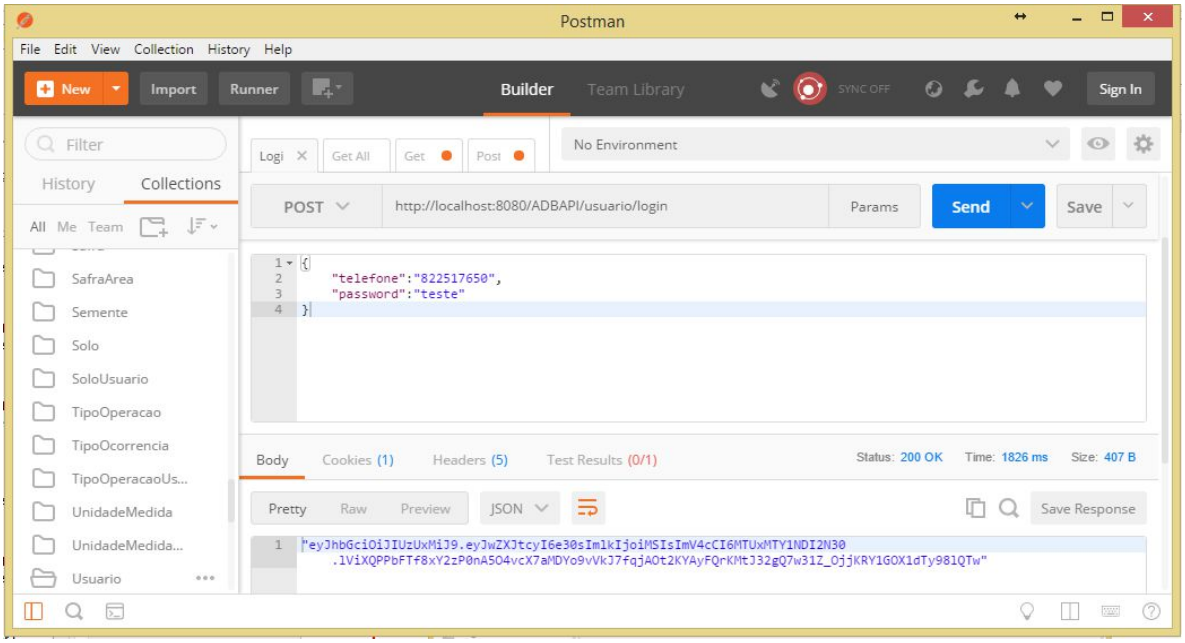

Figura 10 – Postman, aplicação cliente para APIs REST.
Fonte: Autoria própria.

O ambiente Postman é bastante simples e apresenta no lado esquerdo a lista dos recursos da API. Para encaminhar requisições para realização de testes da API, são necessárias configurações, tais como seu endereço, a URI do recurso e a seleção de um método HTTP (POST, GET, PUT, DELETE). Possui ainda opções para adicionar campos no cabeçalho (ex. autenticação) e a indicação do formato de dados, no caso o JSON. No exemplo apresentado na Figura 10, ocorre a solicitação de *login* na API, sendo que, como resposta, é retornado ao cliente o *token* que

deverá ser incluído no cabeçalho em todas próximas requisições.

Na Figura 11, a mesma solicitação de *token* é realizada, desta vez fazendo uso da ferramenta *desktop* Insomnia. A organização de seus componentes é similar a ferramenta Postman.

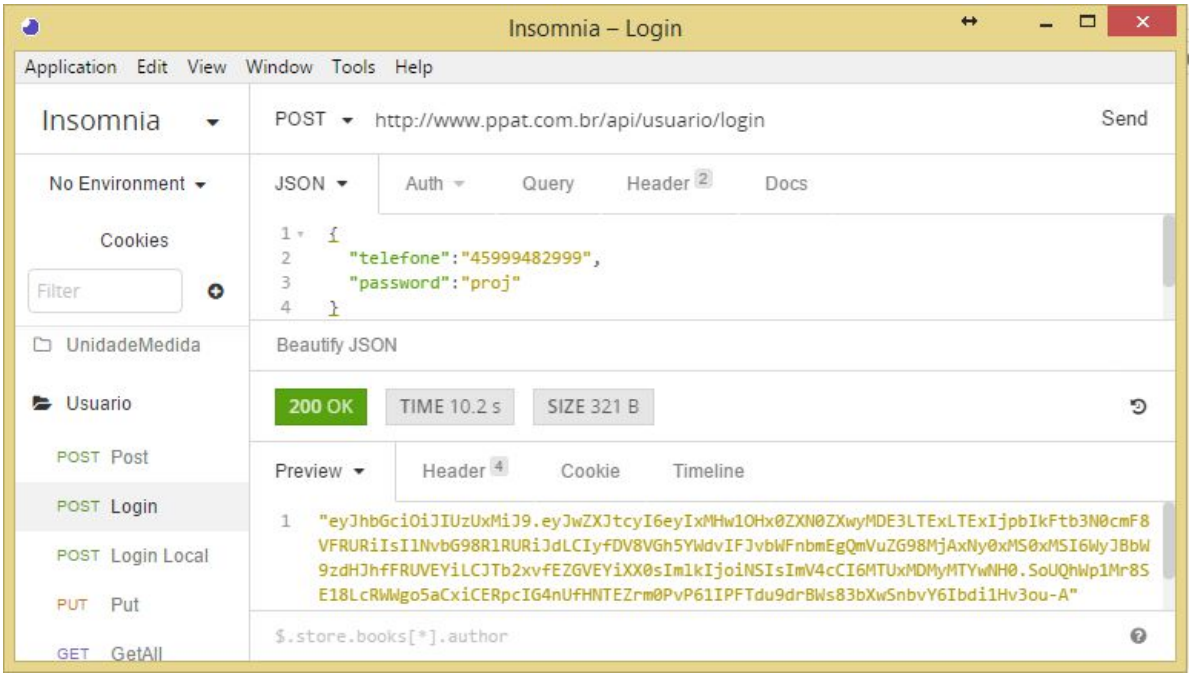

Figura 11 – Insomnia, aplicação cliente para APIs REST.
Fonte: Autoria própria.

Todas as funcionalidades da API foram previamente testadas e validadas fazendo ainda uso de aplicações clientes desenvolvidas especificamente para trabalhar com recursos da API. A Figura 12 apresenta a ferramenta AgDataBox-Mobile, desenvolvida para o sistema operacional Android, a qual é totalmente integrada com a API e que foi desenvolvida para servir como ferramenta para produtores rurais para armazenar seus dados de campo, tais como, demarcação das áreas, ocorrências temporárias ou permanentes, operações de campo, entre outras, além de servir como ferramenta de apoio para o registro de precipitação pluviométrica e agendamento de tarefas. O AgDataBox Mobile foi também usado como ferramenta de testes da API em que, a partir dela, requisições são feitas na API e respostas são recebidas em função da operação escolhida.

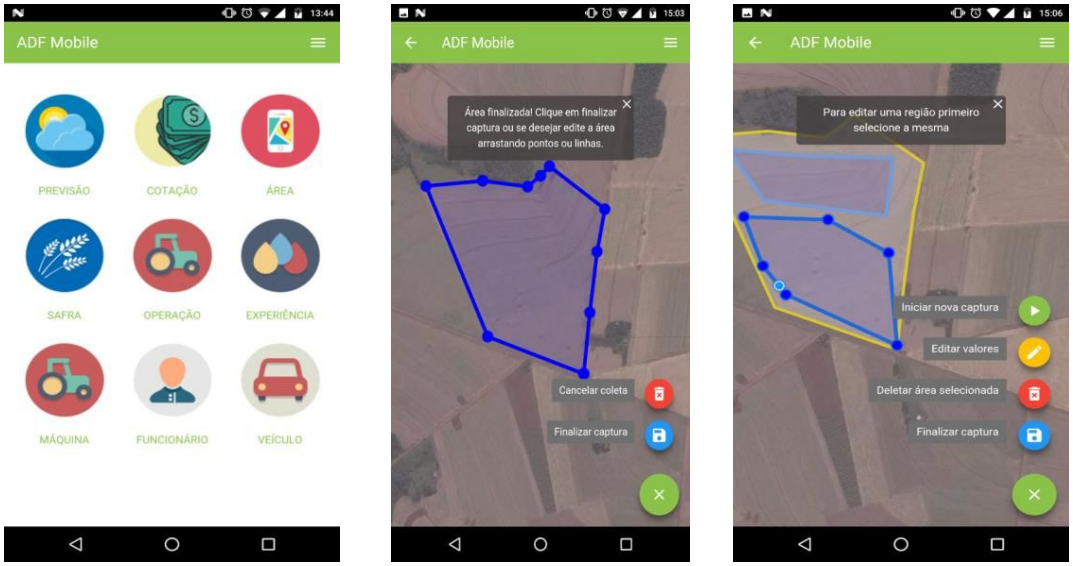

Figura 12 – AgDataBox Mobile, aplicação cliente da AgDataBox API.

Fonte: Autoria própria.

Fazendo uso da linguagem PHP, um módulo web (AgDataBox-Web; Figura 13) foi desenvolvido visando integrar todas as funcionalidades da API e permitir que dados obtidos pelo ambiente Mobile possam ser visualizados/alterados/excluídos por este ambiente e vice-versa.

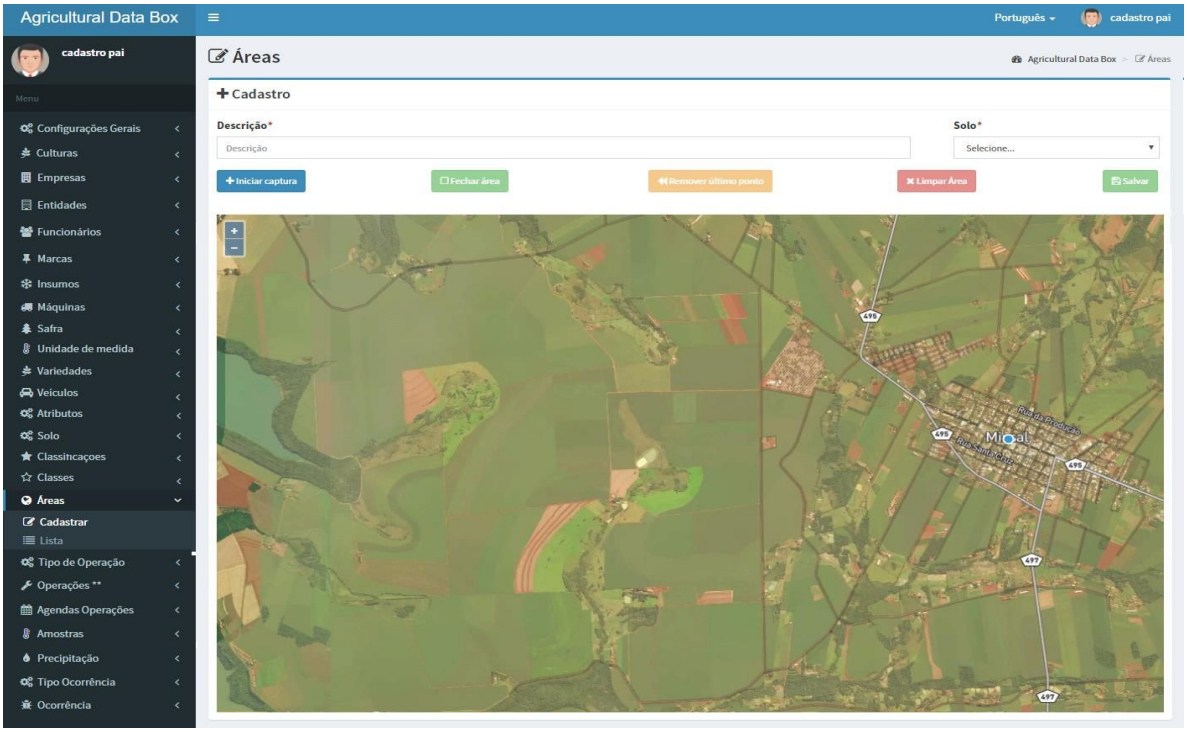

Figura 13 – AgDataBox Web, aplicação cliente da AgDataBox API.

Fonte: Autoria própria.

5.5 DOCUMENTAÇÃO AGRICULTURAL DATA BOX API

Como a API servirá para que outras aplicações façam uso dos recursos implementados, verificou-se a necessidade do desenvolvimento de um ambiente que proporcionasse uma fácil visualização e interpretação de cada um dos recursos implementados na API. Desta forma, uma página web foi desenvolvida para atender a este requisito de forma dinâmica e funcional (disponível em www.ppat.com.br/api/login). Neste ambiente todos os recursos de armazenamento e recuperação de dados estão descritos de forma clara e objetiva, Figura 14. A página web da documentação, incorpora uma aplicação de testes da API, através da qual os desenvolvedores de software do segmento agrícola podem testar a solução e se familiarizar, antes mesmo de começar a desenvolver sua própria aplicação.

Ao selecionar um recurso que se deseja obter informações de funcionamento, uma extensão apresenta os métodos disponíveis e na seleção deste, uma opção *try out* (testar) (Figura 15) é apresentada. Se essa opção for selecionada, os campos do recurso são habilitados. Pela documentação é possível realizar testes em qualquer recurso disponível na API. Possui também opções para visualizar outras ferramentas ligadas ao projeto. A Figura 14, mostra de uma forma geral o ambiente da documentação da API desenvolvida.

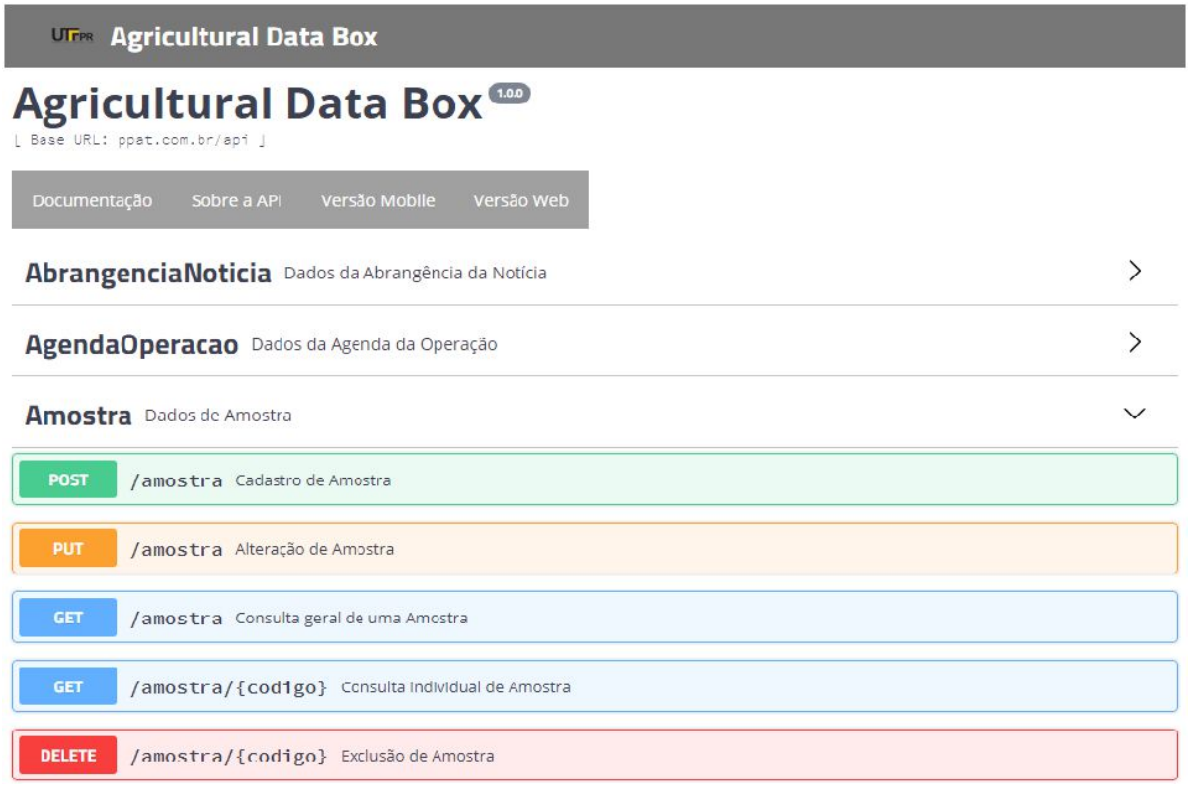

Figura 14 – Documentação e aplicação cliente para a AgDataBox API.

Fonte: Autoria própria.

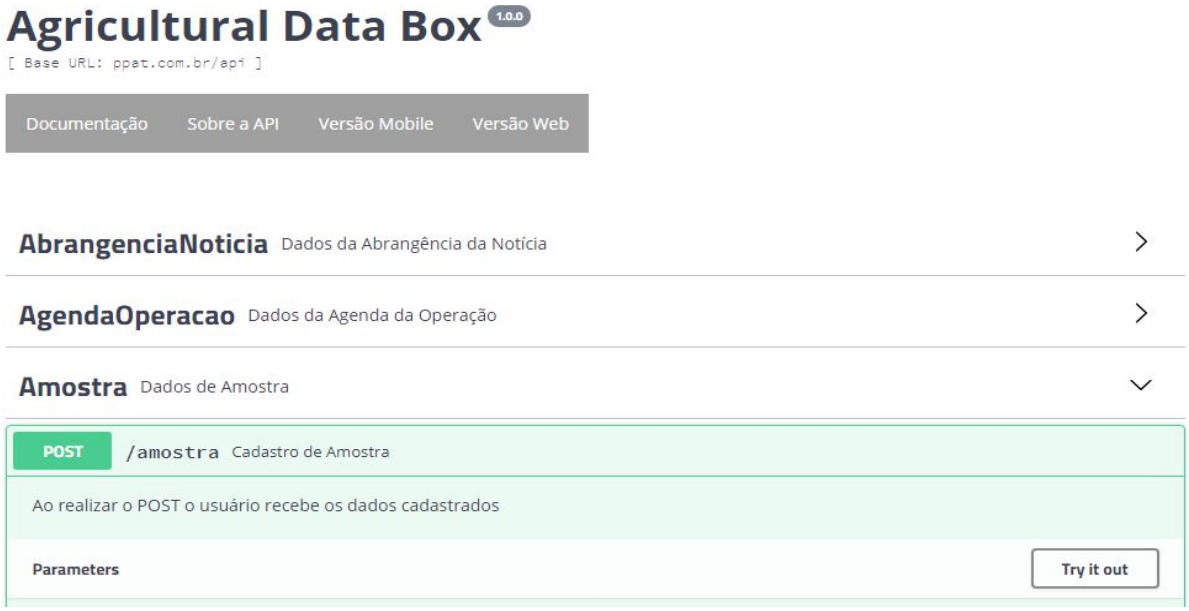

Figura 15 – Opção de teste pela documentação da API.

Fonte: Autoria própria.

CONCLUSÕES

A maioria das aplicações desenvolvidas para o setor agrícola são projetadas para funcionar de forma isolada em um único local. Aplicativos para mobile por exemplo, funcionam apenas em um dispositivo, sem que haja comunicação com outros dispositivos. A API AgDataBox, propõe um padrão de comunicação entre aplicativos do setor agrícola, de forma que eles possam funcionar de forma integrada. A solução permite que dois ou mais aplicativos compartilhem dados de maneira segura. Trata-se de uma ferramenta viável e acessível a qualquer aplicação que tenha acesso a Internet, podendo se comunicar com o servidor através do protocolo HTTP, realizando requisições e recebendo as respostas de forma simples.

Durante seu desenvolvimento, foram realizados testes de validação e segurança, para garantir que todas funcionalidades estejam de acordo com os objetivos do trabalho e que os dados armazenados estejam seguros. Todos os cinco (5) softwares usados (Postman, Insomnia, AgDataBox Mobile, AgDataBox Web e Documentação AgDataBox API) para realizar os testes, mostraram resultados satisfatórios. Pelos resultados dos testes, conclui-se que a solução funciona corretamente; ela está disponível em um servidor baseado na Internet e já está sendo usado. Os resultados obtidos mostraram também uma redução importante no tempo necessário para desenvolver aplicações clientes com base na plataforma, o que realça a importância da solução proposta para a indústria de desenvolvimento de softwares para o setor agrícola.

Por essas razões e pelas constatações obtidas, pode se dizer que as tecnologias aplicadas (REST, linguagem Java, PostgreSQL, PostGIS, Hibernate) foram adequadas para o desenvolvimento da aplicação. Todas tecnologias usadas são modernas e de acesso livre (grátis), o que favorece a disponibilização da solução sem que os usuários tenham que pagar. Verificou-se ainda que o desenvolvimento de aplicações de diferentes tipos e objetivos podem facilmente serem integradas ao ambiente da API, considerando a documentação disponível.

6 TRABALHOS FUTUROS

Alguns aspetos e recomendações são apontadas como medidas para ampliar e melhorar a solução desenvolvida:

1) A plataforma mantém suporte ao padrão HTTP de comunicação, sendo possível adicionar recursos de suporte ao padrão HTTPS para reforçar a segurança da solução;

2) Durante a fase atual, deverão também ser monitoradas as requisições mais frequentes para em seguida serem implementadas como *cacheable* de forma a melhorar a performance da API;

3) Com vista a prever a performance da plataforma quando estiver a ser utilizada em uma carga maior, deverão ser feitos testes simultâneos por vários usuários e monitorar sua performance. Se necessário, algumas ações para melhorar o desempenho poderão ser necessárias;

4) Novos recursos para diferentes tipos de dados poderão ser necessários e implementados, a fim de ampliar e melhorar o ambiente de integração de dados desenvolvido.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ANOUNCIA, S.M., AMALANATHAN, A. A Survey on Role of Computational Approaches in Precision Agriculture. **International Journal of Pharmacy & Technology**. 28503-28520. 2017.

BAZZI, C. L., SOUSA, G. S., BETZEK, N. M. **SDUM Software para definição e avaliação de unidades de manejo em agricultura deprecisão**.1° ed. Medianeira, Paraná, Brazil. 2015.

BLACKMORE, S; GRIEPENTROG, H-W. **A future view of precision farming.** KTBL Sonderveröffentlichung, 131-145. 2002.

BORA, A., BEZBORUAH, T. A Comparative Investigation on Implementation of RESTful versus SOAP based Web Services. **International Journal of Database Theory and Application**, 297-312. 2015.

CAMBOURIS A. N., ZEBARTH, B.J., ZIADI, N., PERRON, I. Precision Agriculture in Potato Production. **PotatoResearch,** 249–262, 2014.

DOERRFELD, B., WOOD, C., ANTHONY, A., SANDOVAL, K., LAURET, A. **The API Economy: Disruption and the Business of APIs Nordic APIs**, 1 ed. 2016.

ELIZABETH, L. **Should you develop a Desktop or Web App**? Sitepoint, 2015.

FENNER, J. **Enterprise Application Integration Techniques**. 2015.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese de Doutorado. University Of California, Irvine. 2000a.

FIELDING, R. **Hypertext Transfer Protocol -- HTTP/1.1**. 2000b. Disponivel em <https://www.w3.org/Protocols/rfc2616/rfc2616.html>

SOARES, R., CUNHA, J. Agricultura de Precisão: Particularidades de Sua Adoção no Sudoeste de Goiás–Brasil, pag. 1809-4430- **Eng. Agric**. v35. n4. p 689- 698, 2015.

FOUNTAS, S., BLACKMORE, S. ESS, D., HAWKINS, S. BLUMHOFF, G., LOWENBERG-DEBOER, J., SORENSEN, C. G. **Farmer Experience with Precision Agriculture in Denmark and US Eastern Corn Belt. Business Media**, Inc, 131-145. 2005.

FOUNTAS, S., CARLIB, G., SØRENSEN, C., G., TSIROPOULOS, Z., CAVALARISD, C., VATSANIDOU, A., LIAKOS, B., CANAVARI, M., WIEBENSOHN, J., TISSERYE, B. Farm management information systems: current situation and future perspectives. **Computers and Electronics in Agriculture**. 115, 40–50. 2015.

FOUNTAS, S., PEDERSEN, S., BLACKMORE, S. ICT in **Precision Agriculture – diffusion of technology**. 2005.

VALENTE, J., DAVID SANZ, ANTONIO BARRIENTOS, JAIME DEL CERRO, ÁNGELA RIBEIRO AND CLAUDIO ROSSI. **An Air-Ground Wireless Sensor Network for Crop Monitoring**, 2011.

GUILLAUD, P. H. **Comprendre les interfaces de programmation**. 2011.

JACOBSON, D., BRAIL, G., WOODS, D. **APIs: A Strategy Guide.** 1 ed. Gravenstein Highway North, Sebastopol, USA: O'Reilly, 2012.

JAROLÍMEK, J., STOČES, M., MASNER, J., VANĚK, J., ŠIMEK, P., PAVLÍK, J., RAJTR, J. User-Technological Index of Precision Agriculture. **Agris on-line Papers in Economics and Informatics**. 69 – 75. 2017.

JWT. **Introduction to JSON Web Tokens**. 2017. Disponivel em <https://jwt.io/introduction/>

LINDBLOM, J., LUNDSTRŐM, CH., LJUNG, M., JONSSON, A. Promoting sustainable intensification in precision agriculture: review of decision support systems development and strategies. **Precision Agriculture**, 2016.

MCCLANAHAN, C. R., MAUCHERAT, R. **Apache Tomcat®.** 2016. Disponivel em < https://tomcat.apache.org/download-80.cgi>

MOORE, M. An investigation into the accuracy of yield maps and their subsequent use in crop management, **Silsoe College Thesis**, 1997.

MOLIN, J.P. **Agricultura de precisão: o gerenciamento da variabilidade**. Piracicaba: ESALQ/USP, 83 p. 2008.

NASH, E. KORDUAN, P., BILL, R. Applications of open geospatial web services in precision agriculture: a review. **Precision Agric**. 546-560. 2009.

NATIONAL RESEARCH COUNCIL. **Precision Agriculture in the 21st century**. Washington, D.C., National Academy Press. 1997.

OAUTH. **About OAUTH**. 2017. Disponivel em< https://oauth.net/>

OLIVEIRA, T. C. A., CAMPOS, S. R. S., PRIETO, L. A. E., LASMAR, E. B. C. Indexação dos dados espaciais do Banco de Dados do Inventário de Minas Gerais. In: **Simpósio Brasileiro de Sensoriamento Remoto**, Florianópolis-SC. **Anais**. v. 9. p. 5973-5981. 2007.

POSTGRESQL. **Sobre o PostgreSQL**. 2016. Disponivel em< https://www.postgresql.org.br/sobre>

POLOJÄRVI, K., KOISTINEN, M., LUIMULA, M., VERRONEN, P., PAHKASALO,

M., TERVONEN, J. Distributed System Architectures, Standardization, and Web-Service Solutions in Precision Agriculture. **Geoprocessing**. 2009.

POP, O. **Comparing Web Applications with Desktop Applications: An Empirical Study. 2000**.

RODRIGUES, M. S.; CORA, J. E.; FERNANDES, C. **Soil sampling intensity and spatial distribution pattern of soils attributes and corn yield in no-tillage system**. Engenharia Agrícola, Jaboticabal, v. 32, n. 5, p. 852-865, set. /out. 2012.

RICHARDSON, L., RUBY, S. **RESTful Web Services**. 1 ed. Gravenstein Highway North, Sebastopol, USA: O'Reilly, 2007.

SCHILDT, H. **Java: The Complete Reference**, Seventh Edition. 7 ed. New York, United States: Mc Graw Hill, 2007.

SCHMULLER, J. **Teach yourself UML in 24 hours**. 3. ed. Indiana, USA: Sam Publishing, 2004.

SHEN, S., BASIST A., HOWARD, A. Structure of a digital agriculture system and agricultural risks due to climate changes. **Agriculture and Agricultural Science Procedia.** 42-51, 2010.

STAFFORD, J.V., EVANS, K., 2000. Spatial distribution of potatocyst nematode and the potential for varying nematicide application. **Proceedings of Fifth InternationalConference on Precision Agriculture**, July 16/19. Bloomington, MN, USA. 2000.

STEINBERGER, G. ROTHMUND, M., AUERNHAMMER, H. Mobile farm equipment as a data source in an agricultural service architecture. **Computers and electronics in agriculture**. 238-246. 2008.

SOOMRO, T. R., AWAN, A., H. Challenges and Future of Enterprise Application Integration. **International Journal of Computer Applications** (0975 – 8887). 2012.

SANTI, A. L.; AMADO, T. J. C.; CHERUBIN, M. R.; MARTIN, T. N.; PIRES, J. L.; DELLA FLORA, L. P.; BASSO, C. J. Análise de componentes principais de atributos químicos e físicos do solo limitantes à produtividade de grãos. **Pesquisa Agropecuária Brasileira**, Brasília, v. 47, n. 9, set. 2012.

SCHWEIK, C., M. **The Open Source Software "Ecosystem**". National Center for Digital Government. 2009.

SHERIFF, P. D. **Designing for Web or Desktop.** Microsoft. 2002. Disponivel em < https://msdn.microsoft.com/en-us/library/ms973831.aspx>

SWAGGER. **What is Swagger?** 2017. Disponivel em < https://swagger.io/docs/specification/about/ >

TILMAN D, BALZER C, HILL J, BEFORT B.L. Global food demand and the sustainable intensification of agriculture. **PNAS**. 108(50):20260–20264. Washington DC: Proceedings of the National Academy of Sciences of the United States of America. 2011.

TUTORIALPOINT. **Java, Simply easy Learning**. 2015. Disponivel em < http://www.tutorialspoint.com/java/java_tutorial.pdf>

WÓJTOWICZ M., WÓJTOWICZ A., PIEKARCZYK J. Application of remote sensing methods in agriculture. Communications in Biometry and Crop Science 11, 31–50. 2016.

ALEXANDRATOS, N., BRUINSMA J. World agriculture towards 2030/2050: The 2012 revision. ESA Working paper No. 12-03. Rome, FAO. 2012