Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Sistema para Coleta e Avaliação de Relatórios Individuais de Trabalho

Douglas Baumgratz de Carvalho

JUIZ DE FORA
FEVEREIRO, 2022

Sistema para Coleta e Avaliação de Relatórios Individuais de Trabalho

Douglas Baumgratz de Carvalho

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Sistemas de Informação

Orientador: Igor de Oliveira Knop

JUIZ DE FORA
FEVEREIRO, 2022

Sistema para Coleta e Avaliação de Relatórios Individuais de Trabalho

Douglas Baumgratz de Carvalho

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Igor de Oliveira Knop
D.Sc. Modelagem Computacional

Luciana Conceição Dias Campos
D.Sc. Engenharia Elétrica

André Luiz de Oliveira
D.Sc. Ciência da Computação

JUIZ DE FORA

15 DE FEVEREIRO, 2022

Resumo

O Relatório Individual de Trabalho (RIT) é o meio pelo qual o professor do magistério superior apresenta, à sua chefia imediata e à sociedade, as atividades desenvolvidas durante o ano letivo. As atividades são categorizadas nos eixos: ensino, pesquisa, extensão e gestão acadêmica. Realizar a coleta e a posterior análise desses dados não é uma tarefa fácil em função de grande variedade das atividades e, principalmente, a divulgação e cruzamento dos dados ao longo dos anos não estão disponíveis para ferramentas de pesquisa. Este trabalho de conclusão de curso consiste em automatizar o processo de coleta e avaliação de RITs, para uma melhor organização das instituições de ensino superior, através de um sistema *online* com uma interface amigável para humanos cujo os dados são exportados na forma de serviços. Através das análises automatizadas dos resultados, espera-se apresentar uma visão gerencial mais acertada sobre os RITs e dessa forma, contribuir para um aumento na eficiência da gestão do ensino público.

Palavras-chave: desenvolvimento de software; gestão pública; gestão acadêmica.

Abstract

The Individual Work Report (IWR) is how the higher education professors in Brazil presents, to his immediate supervisor and society, the activities developed during year. The activities are categorized in the following axes: teaching, research, extension, and academic management. Carrying out the collection and subsequent analysis of these data is not an easy task due to the great variety of activities and, mainly, the dissemination and crossing of data over the years are not available for research tools. This course conclusion work consists of automating the process of collecting and evaluating IWRs, for a better organization of higher education institutions, through an online system with a human-friendly interface whose data are exported in the form of services. Through the automated analysis of the results, it is expected to present a more accurate managerial view of the IWRs and, in this way, contribute to an increase in the efficiency of the management of public education.

Keywords: software development; public management; academic management.

Agradecimentos

Aos meus pais, pela confiança no meu progresso e pelo apoio emocional.

A todos os meus parentes e amigos, pelo encorajamento e apoio.

Ao Professor Doutor Igor de Oliveira Knop pela orientação, amizade, ensinamentos e dedicação do seu escasso tempo ao meu projeto de pesquisa.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de alguma forma direta e indiretamente para o nosso enriquecimento pessoal e profissional.

"A vitalidade é demonstrada não apenas pela persistência, mas pela capacidade de começar de novo".

F. Scott Fitzgerald

Conteúdo

Lista de Figuras

Lista de Abreviações e Siglas

API *Application Programming Interface*. 6, 14, 21, 23, 25, 26, 28, 32, 36, 37, 39, 44, 49

CEPE Conselho de Ensino, Pesquisa e Extensão. 10

CPAD Comissão Permanente de Avaliação Docente. 13

CRUD *Create, Read, Update, Delete*. 31, 32

CSS *Cascading Style Sheet*. 14, 16, 17

DCC Departamento de Ciência da Computação. 6, 11, 20, 33, 35, 36, 48, 49

HTML *HyperText Markup Language*. 14, 16, 17

HTTP *Hypertext Transfer Protocol*. 13, 14, 26

IaaS *Infrastructure as a Service*. 19

ICE Instituto de Ciências Exatas. 12, 13

JS *JavaScript*. 14–18

JSON *JavaScript Object Notation*. 25, 27, 33

JSX *JavaScript Sintaxe Extension*. 15

JWT *Json Web Token*. 14, 26

LDB Lei de Diretrizes e Bases da Educação Nacional. 12

MEC Ministério da Educação. 12, 13

NoSQL *Not Only SQL*. 25

ODM *Object Document Mapping*. 18, 25

ORM *Object Relational Mapping*. 18

OS *Operating System*. 15, 16

PaaS *Platform as a Service*. 19, 21

PIT Plano Individual de Trabalho. 10, 13, 49

REST *Representational State Transfer*. 14, 21, 23, 26

RIT Relatório Individual de Trabalho. 1, 10, 11, 13, 21, 23, 28, 33, 39, 48, 49

SaaS *Software as a Service*. 19

SDK *Software Development Kit*. 16

SGBD Sistema de Gerenciamento de Banco de Dados. 18, 25

SQL *Structed Query Language*. 18

UFJF Universidade Federal de Juiz de Fora. 10–13, 20, 24, 39, 48

WEB *World Wide Web*. 13, 14, 16

WWW *World Wide Web*. 13

XML *Extensible Markup Language*. 15, 33

1 Introdução

Os dados são muito importantes para qualquer organização, independentemente do seu ramo de atuação e porte. Todas as informações inter-relacionadas permitem aos gerentes, diretores ou superiores estarem cientes dos fatores preponderantes, facilitando a análise para melhoria de seus resultados, com tomadas de decisões mais assertivas. A coleta de dados é um processo de obtenção por diversos meios como, por exemplo, técnicas de questionário, entrevista, observação, análise de conteúdo, entre outros (IBC, 2017).

Na esfera pública, as instituições não são diferentes e devem sempre fazer um acompanhamento de seus índices para melhor atender as suas finalidades. Nas instituições de ensino superior, por conseguinte, na Universidade Federal de Juiz de Fora (UFJF), todos os docentes devem apresentar um Plano Individual de Trabalho (PIT) tal qual definido na Resolução 46/95 do Conselho de Ensino, Pesquisa e Extensão (CEPE) (UFJF, 1995). Cada docente é responsável pela elaboração do seu próprio PIT, onde deve constar um detalhamento de seu planejamento de atividades para o próximo ano. Ao final do período, um RIT detalha em quais atividades seu esforço foi desprendido e quais resultados foram alcançados durante o ano. De posse desses relatórios, a assembleia docente e a chefia imediada avaliam se o professor, historicamente, cumpriu suas atividades de acordo com seu PIT e se sua produção está condizente com o plano de carreira.

Mais recentemente, adiciona-se a demanda no serviço público, na forma da Lei nº 14.129 de 29 de março de 2021, que estabelece as diretrizes para o Governo Digital (BRASIL, 2021). Essa lei determina a busca pela transparência na execução dos serviços públicos e o monitoramento da qualidade, bem como a prestar contas diretamente à população sobre a gestão dos recursos públicos e a interoperabilidade de sistemas e a promoção de dados abertos.

Entretanto, por mais que essa seja uma das atividades previstas nas instituições de ensino superior, não há uma plataforma unificada para envio e avaliação dos relatórios dos professores nem um acompanhamento da evolução dos dados ao longo dos anos. Uma plataforma para coletar e apresentar os dados, considerando a mutabilidade dos processos

e critérios da instituição e departamentos, se faz mais do que necessária.

Neste trabalho foi realizada uma pesquisa qualitativa em paralelo à proposta e desenvolvimento de software de apoio à melhoria do processo de coleta e avaliação de RITs individuais no ensino superior, bem como dar uma visualização do esforço e produção discente ao nível de, individual, departamento e instituição. Complementarmente, este trabalho também realiza um estudo de caso envolvendo os RITs dos docentes do Departamento de Ciência da Computação (DCC) da UFJF, modelando e importando os dados ao nível de departamento e instituto.

Portanto, o objetivo geral deste trabalho é a automatização do processo de coleta, validação, análise, e transparência dos RITs.

Para atingi-lo, serão necessários os seguintes objetivos específicos:

1. Realizar uma modelagem dos dados a serem colhidos, bem como sua categorização e inter-relações;
2. Propor um método auxiliado por computador para coleta de dados;
3. Apresentar uma série de relatórios que vão servir de apoio a tomada de decisão departamental;
4. Prover uma interface para que outros aplicativos possam consumir os dados coletados;
5. Realizar o estudo de caso sobre os dados públicos e relatórios de um departamento real.

Através da conclusão desses objetivos específicos, espera-se que o sistema forneça informações sobre os dados com uma maior clareza e transparência para a instituição.

Este trabalho está organizado em cinco capítulos. Além desta Introdução, o Capítulo 2 traz a fundamentação teórica com os conceitos aqui tratados. O Capítulo 3 discute o método utilizado para investigar os resultados encontrados. O Capítulo 4 traz o relato do desenvolvimento completo e apresenta quais foram os resultados obtidos. Por fim, o Capítulo 5 traz as considerações finais, limitações e sugestões para trabalhos futuros.

2 Fundamentação Conceitual

Este capítulo apresenta uma revisão conceitual e fundamentação do magistério superior através da Lei de Diretrizes e Bases da Educação Nacional (LDB) (Brasil, 2021), resoluções do Ministério da Educação (MEC), da UFJF e do Instituto de Ciências Exatas (ICE) e abordar todas as tecnologias utilizadas no desenvolvimento do aplicativo proposto.

2.1 Magistério no Ensino Superior

A educação superior tem por finalidade estimular a criação cultural, do desenvolvimento do espírito científico e do pensamento reflexivo (Brasil, 2021, art. 43). É condição para habilitação no Magistério Superior ao docente de nível superior ter concluído a formação em programas de pós-graduação, na modalidade de mestrado ou doutorado (*stricto sensu*), ou o notório saber em casos particulares (Brasil, 2021, art. 66). Ao corpo docente é atribuído atividades de extensão, planos de trabalho, programas e projetos de pesquisa científica (Brasil, 2021, art. 53). São atividades dos docentes do magistério superior a aplicação de ensino e pesquisa, exercendo funções de monitoria de acordo com rendimento e plano de estudos (Brasil, 2021, art. 84).

Segundo Drucker (1989), todas as organizações passam pelo constante problema de uma mediocridade segura. Sendo assim, para a saúde organizacional há uma demanda por alto desempenho. Há uma grande concorrência entre as instituições a procura por resultados rápidos, visto que há demandas a serem atendidas e metas a serem cumpridas. É necessário às instituições estarem sempre a procura de melhores soluções para seus problemas. Uma simples análise incompleta, pode levar a tomada de decisões incorretas ou fora dos parâmetros esperados, acarretando prejuízos que podem colocar em risco seus rendimentos. A qualidade dos serviços são essenciais para o desenvolvimento tanto individual quanto institucional (REIFSCHNEIDER, 2010).

Ao se falar em avaliação de desempenho, espera-se que o processo sistemático de coleta de dados, que se baseia em critérios pré estabelecidos e conhecidos por aqueles

que são avaliados, permita a formação de valores baseados em evidências (REIFSCHNEIDER, 2010). Seguindo as determinações do MEC, a resolução do ICE da UFJF de 02/2016 normatiza a elaboração, acompanhamento e avaliação do Plano Individual de Trabalho (PIT) e do Relatório Individual de Trabalho (RIT) dos docentes, sendo a principal forma de planejamento e acompanhamento das atividades dos professores (ICE, 2016).

2.2 Atividades desenvolvidas pelos docentes

A proposta de PIT submetida pelo(a) docente deverá ser avaliada pela Comissão Permanente de Avaliação Docente (CPAD) que irá verificar os dados inseridos e se corresponde com o plano de metas estabelecido pelo departamento (ICE-UFJF, 2016, art. 6). O PIT é uma distribuição de esforço em cinco eixos (ensino, pesquisa, extensão, gestão e afastamento/capacitação), normalmente preenchido utilizando um equivalente de horas semanais, acompanhando o regime de trabalho do docente em 40 ou 20 horas.

O RIT, por sua vez, é um relato detalhado de cada atividade, acompanhado de documentos comprobatórios quando necessário. As atividades também seguem os mesmos eixos de ensino, pesquisa, extensão, gestão e afastamento/capacitação. O RIT é avaliado pela CPAD que irá emitir um parecer se o mesmo encontra-se em consonância ou não com o PIT apresentado pelo professor no início do ano (ICE-UFJF, 2016, art. 7). Os pareceres PIT e RIT serão submetidos à aprovação pela assembleia departamental para aprovação (ICE-UFJF, 2016).

2.3 Desenvolvimento Web

O desenvolvimento web é a área da Tecnologia da Informação voltada para construção de sites na *World Wide Web* (WWW) ou rede privada (MOZILLA, 2022). Ferreira (2021) a *World Wide Web* (WEB) foi criada em 1989 por Tim Berners-Lee, quando se tornou a figura central no processo de desenvolvimento da internet por fazer a primeira comunicação bem sucedida entre cliente e servidor via *Hypertext Transfer Protocol* (HTTP)[1].

[1] O site MDN Web Docs encontra-se disponível em ⟨https://developer.mozilla.org/pt-BR/docs/Web/HTTP⟩

O modelo cliente-servidor, trata a respeito da separação de responsabilidades, ou seja, o cliente não implica em como os dados são armazenados e os servidores não se importam com qual aplicativo o usuário está acessando desde que os protocolos de comunicação e autenticação sejam respeitados.

Segundo Marques (2018), as APIs, ou uma tradução livre, interface para programação de aplicações, são responsáveis pela comunicação entre uma aplicação cliente do usuário e a aplicação em execução no servidor Web, que por sua vez pode fazer acesso à base de dados, bem como a gestão de todo conteúdo e tratamento estatístico dos dados.

Os serviços *Representational State Transfer* (REST) são implementações cliente-servidor que utilizam os métodos do protocolo HTTP para prover uma API (FIELDING, 2000). Para garantir a segurança entre a comunicação das APIs, uma das técnicas é o uso do *Json Web Token* (JWT) que realiza a autenticação entre duas partes através de um *token* assinado que autentica uma requisição web (MONTANHEIRO; CARVALHO; RODRIGUES, 2017).

Segundo Loudon (2010), as ferramentas padrão para a criação de páginas WEB são: *HyperText Markup Language* (HTML), *Cascading Style Sheet* (CSS) e *JavaScript* (JS) . O HTML é uma linguagem de marcação que fornece a estrutura base para as aplicações web. O CSS é uma linguagem de marcação responsável pela estilização das páginas. O JS é uma linguagem de programação dinâmica de alto nível que permite a manipulação de diversos elementos da página, tornando-a mais interativa e com uma experiência de usuário melhorada. Segundo OpenJS Foundation (s.d.), em 2009 o *Node.js* foi criado sobre o motor de JavaScript do Google Chrome como sendo o primeiro ambiente de execução JS com o propósito de uma linguagem *back-end*, permitindo, pela primeira vez, se utilizar uma única linguagem tanto no servidor quanto no navegador, o *front-end*.

Com o passar dos anos, as exigências por novas funcionalidades foram aumentando e com isso diversas bibliotecas e *frameworks* para *front-end* foram desenvolvidos. Os *frameworks* são responsáveis por dar mais praticidade no desenvolvimento de interfaces com usuário. Através do reuso de código, tarefas básicas são automatizadas permitindo aos desenvolvedores focarem mais tempo nas regras de negócio. Entre as mais populares

no momento da escrita deste trabalho, estão o AngularJS[2], o VueJS[3] e o ReactJS[4], que foi o framework usado neste trabalho.

O ReactJS foi desenvolvido pelo Facebook e passou a ser utilizada na interface do mural de notícias da rede social. Anos depois, o código foi aberto para comunidade, o que colaborou para sua grande popularização. Ele possui suporte a componentes, logo favorece a reusabilidade que é um fator preponderante no desenvolvimento de sistemas. Os componentes são escritos em *JavaScript Sintaxe Extension* (JSX), que é uma extensão de sintaxe opcional para JavaScript parecida com *Extensible Markup Language* (XML), mas permite declarar uma estrutura para o componente em meio ao código imperativo do JS (REACTJS, 2022).

2.4 Dispositivos Móveis

Com o avanço da internet e a popularização dos computadores nos anos de 1990 e 2000, diversos dispositivos vem sendo criados para facilitar o dia a dia das pessoas. Tais dispositivos estão alterando a forma de viver, pois, com a liberdade de se desprender de uma infra-estrutura fixa de comunicação de dados, permite usufruir de toda tecnologia disponível de qualquer lugar a qualquer momento (MATHEUS; LOUREIRO, 1998). Em 2020 o Brasil registrou 234,07 milhões de acessos, representando um aumento de 7,39 milhões em relação a 2019 (Ministério das Comunicações, 2021). Com o aumento de acessos, o desenvolvimento de novas aplicações para dispositivos móveis tem crescido exponencialmente. As principais plataformas de *smartphones* são Android[5] e IOS.

O Android é um *Operating System* (OS) de software *open-source*[6] para dispositivos móveis mantido pelo *Google* que é considerada a plataforma mais utilizada no mundo dos *smartphones*, principalmente no Brasil. Com sua arquitetura baseada em GNU/Linux[7], aproveitam a utilização de diversos recursos como gerenciamento de

[2]O site oficial do AngularJS encontra-se disponível em ⟨https://angular.io/⟩

[3]O site oficial do VueJS encontra-se disponível em ⟨https://vuejs.org/⟩

[4]O site oficial do ReactJS encontra-se disponível em ⟨https://reactjs.org/⟩

[5]O site oficial da plataforma Android encontra-se disponível em ⟨https://www.android.com/intl/pt-BR_br/⟩

[6]Quando um software possui seu código fonte aberto. Dependendo da licença, em geral, é permitido fazer alterações diretamente no código fonte.

[7]O site oficial do GNU encontra-se disponível em ⟨https://www.gnu.org/⟩

memória, processos, automatização de rede e drivers (GANDHEWAR; SHEIKH, 2009). O desenvolvimento mobile Android nativo ocorre por meio do *Software Development Kit* (SDK) do Android, onde as linguagens mais utilizadas são Java[8] e Kotlin[9].

O iOS é um sistema operacional desenvolvido exclusivamente para dispositivos móveis de propriedade da empresa Apple[10], ou seja, *hardware* de terceiros não são permitidos. É muito utilizado e possui como público alvo as classes sociais mais altas, o que inclui usuários de maior poder aquisitivo. As principais linguagens de programação para o desenvolvimento nativo são Objective-C[11] ou Swift[12].

O desenvolvimento nativo é programado na linguagem de cada OS. Possuem a vantagem de maior desempenho e perdem em outros fatores como, a implementação de inúmeros códigos, tornando o processo lento (CERQUEIRA; BITTENCOURT, 2014). Cada plataforma possui suas linguagens de programação específicas, o que justifica o alto custo de mão de obra especializada com competências técnicas necessárias. Outro problema é o não reaproveitamento de códigos existentes em tais plataformas, por exemplo, o mesmo código fonte desenvolvido em Android nativo não funciona no iOS nativo e vice-versa. O desenvolvimento híbrido surgiu para resolver o problema de uma plataforma de desenvolvimento que abrange todos os sistemas operacionais (VENTEU; PINTO, 2018).

2.5 Desenvolvimento Híbrido

O desenvolvimento híbrido para dispositivos móveis consiste na mistura de tecnologias WEB como HTML, JS e CSS em conjunto com um *framework* que tenha acesso às funções nativas do aparelho. São desenvolvidas sem o uso de linguagens e SDKs nativos, por isso não apresentam todo o desempenho que o dispositivo pode oferecer (CERQUEIRA; BITTENCOURT, 2014).

Alguns *frameworks* conhecidos foram desenvolvidos para o desenvolvimento híbrido, são eles: React Native[13]; Ionic; MoSync e; Apache Cordova.

[8] O site oficial da linguagem Java encontra-se disponível em ⟨https://www.java.com/pt-BR/⟩
[9] O site oficial da linguagem Kotlin encontra-se disponível em ⟨https://kotlinlang.org/⟩
[10] O link oficial da empresa Apple encontra-se disponível em ⟨https://www.apple.com/br/⟩
[11] O site oficial da linguagem Objective-C encontra-se disponível em ⟨http://www.objective-cloud.com/⟩
[12] O site oficial da linguagem Swift encontra-se disponível em ⟨https://swift.org/⟩
[13] O site oficial do React Native está disponível em ⟨https://reactnative.dev/⟩

React Native é um *framework open-source*, baseado no React desenvolvido pela equipe do Facebook, utilizando a linguagem JS. Seu lema é "aprenda de uma vez (React.js) e escreva em todo lugar (Web, Android, IOS)" (BEZERRA; VIANNA, 2021). O React Native permite que desenvolvedores web utilizem seus conhecimentos adquiridos no desenvolvimento de aplicativos móveis nativos.

O *framework* Expo[14], é composto por recursos que permitem desenvolver, construir, implantar e iterar nos aplicativos, tornando o processo de construção de aplicações Android, iOS e Web mais descomplicado (EXPO, s.d.). O aplicativo pode ser encontrado na Google Play[15] e na App Store[16].

Ionic[17] é um *framework open-source*, para construções de aplicações móveis. Ele utiliza o *Angular* porém com suporte para o *React.js* e *Vue.js*. É construído nas tecnologias web (HTML, CSS e JS), por isso se comporta como um site executando no *WebView* acoplado ao código nativo (BEZERRA; VIANNA, 2021). Assim como no React Native, permite que desenvolvedores web utilizem seus conhecimentos no desenvolvimento de aplicativos móveis.

MoSync[18] é um framework que permite o desenvolvimento de uma aplicação tanto nas linguagens web, quanto na linguagem *C/C++* (CERQUEIRA; BITTENCOURT, 2014). Se divide em dois kits de desenvolvimento diferentes: MoSync SDK com suporte C/C++, HTML e JS e MoSync Reload com suporte apenas a HTML e JS.

Apache Cordova[19] é um *framework* de desenvolvimento mobile que inicialmente foi desenvolvido pela Apache[20] para que fosse utilizado nas aplicações internas da empresa. O Cordova surgiu a partir do código fonte do PhoneGap[21] criado pela Adobe[22]. A equipe da Apache criou várias melhorias nas funções JS do PhoneGap, tanto que hoje o PhoneGap faz uso do Cordova em sua plataforma de desenvolvimento. O Cordova utiliza como

[14] O site oficial do Expo encontra-se disponível em ⟨https://expo.io/⟩

[15] O link oficial para o aplicativo Expo na Google Play encontra-se disponível em ⟨https://play.google.com/store/apps/details?id=host.exp.exponent⟩

[16] O link oficial para o aplicativo Expo na App Store encontra-se disponível em ⟨https://apps.apple.com/us/app/expo-go/id982107779⟩

[17] O site oficial do Ionic Framework encontra-se disponível em ⟨https://ionicframework.com/⟩

[18] O site oficial do MoSync encontra-se disponível em ⟨https://github.com/MoSync/MoSync⟩

[19] O site oficial do Apache Cordova encontra-se disponível em ⟨https://cordova.apache.org/⟩

[20] O site oficial da Apache encontra-se disponível em ⟨ https://www.apache.org/⟩

[21] O site oficial do Phonegap encontra-se disponível em⟨https://phonegap.com/⟩

[22] O site oficial da Adobe encontra-se disponível em ⟨ https://www.adobe.com/⟩

principal linguagem de programação o JS, o que permite ao desenvolvedor acessar as funções nativas de dispositivos que executam Android, IOS ou BlackBerry.

O Sencha Touch[23] inicialmente era uma biblioteca de interface, porém hoje em dia com suporte ao Apache Cordova e ao Adobe PhoneGap Build, tornou-se um concorrente do PhoneGap. Traz consigo uma API completa de interfaces, com isso, não há a necessidade de frameworks externos para gerenciar a responsividade dos aplicativos (CERQUEIRA; BITTENCOURT, 2014).

2.6 Banco de dados

Diante do grande volume de dados gerado pelas aplicações, houve a necessidade da criação de um sistema para que todas informações possam ser armazenadas. Diversos Sistema de Gerenciamento de Banco de Dados (SGBD) foram criados com propósitos diferentes. Por exemplo Oracle[24], SQL Server[25], MongoDB[26], Redis[27], Neo4J[28], DynamoDB[29], dentre outros.

Para trabalhar com SGBD, existem ferramentas que permitem escrever consultas sem a utilização da linguagem nativa dos SGBDs. Uma das ferramentas tem por objetivo mapear os dados da aplicação em uma linguagem orientada a objetos e realizar operações em tabelas relacionais via *Structed Query Language* (SQL) e é chamada de *Object Relational Mapping* (ORM) (ALCARAZ, 2016). Já o *Object Document Mapping* (ODM), que podemos citar a ferramenta *Mongoose*[30] como solução para mapeamento objeto-documento *MongoDB* para *NodeJS* através de *schemas*. O *MongoDB* é um dos bancos de dados orientado a documentos mais populares que agrupa todos os dados em um único documento estruturado (SOUZA; OLIVEIRA, 2019).

[23] O site oficial do Sencha Touch encontra-se disponível em ⟨ https://www.sencha.com/products/touch/ ⟩

[24] O site oficial do Oracle encontra-se disponível em ⟨https://www.oracle.com/br/database/⟩

[25] O site oficial do SQL Server encontra-se disponível em ⟨https://www.microsoft.com/pt-br/sql-server/sql-server-downloads⟩

[26] O site oficial do MongoDB encontra-se disponível em ⟨https://www.mongodb.com/pt-br⟩

[27] O site oficial do Redis encontra-se disponível em ⟨https://redis.io/⟩

[28] O site oficial do Neo4j encontra-se disponível em ⟨https://neo4j.com/⟩

[29] O site oficial do DynamoDB encontra-se disponível em ⟨https://aws.amazon.com/pt/dynamodb/⟩

[30] O site oficial do *Mongoose* encontra-se disponível em ⟨https://mongoosejs.com/⟩

2.7 Computação na Nuvem

De acordo com Ruschel, Zanotto e Mota (2010) a computação em nuvem (*cloud computing*) é a ideia de utilizar os mais variados tipos de aplicações através da internet com a mesma facilidade de tê-las instaladas no próprio computador.

A computação em nuvem distribui os recursos na forma de serviços e pode ser dividida em três modelos: *Software as a Service* (SaaS), *Platform as a Service* (PaaS) e *Infrastructure as a Service* (IaaS). Com o SaaS é possível gerenciar os problemas técnicos potenciais como dados, *middleware*, servidores e armazenamento. Exemplos comuns de SaaS são: *Google Workspace*[31], *Dropbox*[32], *Salesforce*[33] e *Cisco*[34]. O PaaS é semelhante ao SaaS, exceto que em vez de entregar o software pela internet, ele fornece uma plataforma para criação de software. Exemplos comuns de PaaS são: *Heroku*[35], *MongoDB Atlas*[36] e *OpenShift*[37]. Já o IaaS são compostos por recursos de computação automatizados e altamente escalonáveis. Exemplos comuns de IaaS são: *DigitalOcean*[38], *Amazon Web Services (AWS)*[39] e *Linode*[40].

[31] O site oficial do Google Workspace encontra-se disponível em ⟨https://workspace.google.com/intl/pt-BR/⟩

[32] O site oficial do Dropbox encontra-se disponível em ⟨https://www.dropbox.com/⟩

[33] O site oficial do Salesforce encontra-se disponível em ⟨https://www.salesforce.com/br/⟩

[34] O site oficial do Cisco encontra-se disponível em ⟨https://www.cisco.com/⟩

[35] O site oficial do Heroku encontra-se disponível em ⟨https://www.heroku.com/⟩

[36] O site oficial do MongoDB Atlas encontra-se disponível em ⟨https://www.mongodb.com/pt-br/atlas/database⟩

[37] O site oficial do OpenShift encontra-se disponível em ⟨https://docs.openshift.com/⟩

[38] O site oficial do DigitalOcean encontra-se disponível em ⟨https://www.digitalocean.com/⟩

[39] O site oficial do Amazon Web Services (AWS) encontra-se disponível em ⟨https://aws.amazon.com/pt/⟩

[40] O site oficial do Linode encontra-se disponível em ⟨https://www.linode.com/pt/⟩

3 Método

Este capítulo descreve os processos utilizados neste trabalho para atingir os objetivos. Na Seção 3.1 é apresentado a caracterização da pesquisa, e na Seção 3.2 apresenta uma visão geral da estrutura do trabalho desenvolvido.

3.1 Caracterização da Pesquisa

A pesquisa deste trabalho de conclusão de curso é caracterizada como desenvolvimento de software, no qual um processo será capturado por um conjunto de serviços web e um cliente multiplataforma.

O sistema será desenvolvido, com o objetivo de gerar uma aplicação web para realizar a coleta através de formulários, das planilhas do DCC e de arquivos baixados da plataforma *Lattes*.

De posse dos dados será feito a análise dos dados através de diversos gráficos de relatórios individuais de trabalho de professores. Espera-se que através das análises, as instituições de ensino superior possam ter uma visão geral mais detalhada sobre os dados e que aumente a eficiência em suas gestões.

Adicionalmente, espera-se realizar um estudo de caso no DCC da UFJF realizando uma coleta de dados, adequando o projeto aos processos definidos em resoluções institucionais ou definições federais.

3.2 Visão Geral

Este trabalho foi dividido em cinco etapas. As etapas são:

1. Modelagem inicial dos dados para um banco de documentos;
2. Um mapeamento objeto-documentos, com as restrições necessárias das regras de negócio;

3. Implementação de um servidor para *back-end* que irá prover uma API de serviços;

4. Implementação de um cliente multi-plataforma para servir de *front-end* utilizando uma das soluções de desenvolvimento híbrido;

5. A implantação na nuvem (PaaS) para um estudo de caso.

Uma modelagem inicial é realizada, objetivando destacar os principais conceitos e relações. Em linhas gerais, a solução desenvolvida se baseia na arquitetura cliente-servidor, onde o servidor fornece os pontos terminais (em uma tradução livre de *endpoints*) e disponibiliza uma interface para acesso por aplicações REST. O servidor se comunica com o banco de dados de documentos para controle das informações e persistência sendo acessado pelo cliente por um aplicativo via API REST. Os arquivos de importação de dados são armazenados temporariamente no servidor.

O servidor, banco de dados e aplicativo encontram-se em uma plataforma PaaS, onde se dá a comunicação entre as partes. Para o envio de imagens de perfil, foi utilizado um serviço em nuvem que oferece uma solução para todo o gerenciamento de envio e armazenagem através da API disponibilizada.

O servidor também se conecta a uma API dedicada para interação com e-mails no momento do cadastro inicial na aplicação onde é enviado um e-mail de verificação de cadastro. Um serviço em nuvem de armazenamento de arquivos foi contratado para comunicação com a API e armazenamento dos dados. Adicionalmente, há o acesso ao serviço do Google Docs e plataforma *Lattes* da aplicação para importação de dados que são armazenados temporariamente no servidor. A comunicação entre as partes pode ser visualizada na Figura 3.1.

Este capítulo apresentou as etapas de desenvolvimento a visão geral para a construção de uma solução para a coleta e análise de RITs. No próximo capítulo o processo de desenvolvimento e os resultados serão apresentados.

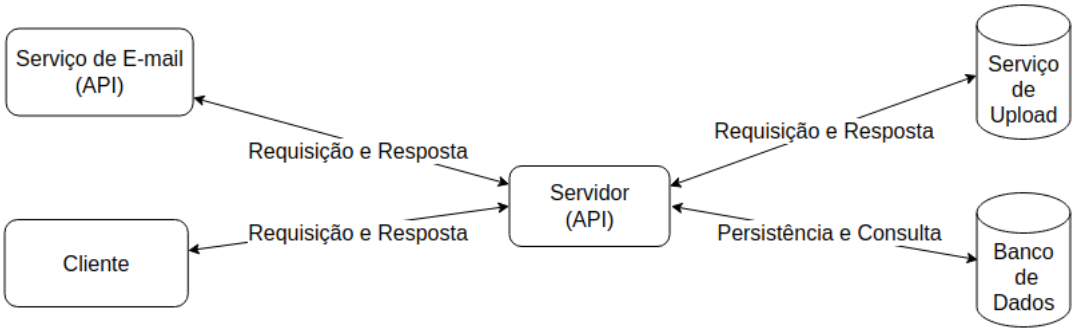


Figura 3.1: A comunicação com o servidor e integração com os serviços de email, upload de imagem, persistência, e importação de dados.

4 Desenvolvimento

Neste capítulo é exposto o processo e os artefatos criados que compõem o sistema de coleta dos RITs. Na Seção 4.1 é apresentado os principais pontos que compõem o sistema. Na Seção 4.2 é descrito os principais conceitos e relações das coleções do banco de dados. A Seção 4.3 traz as principais ferramentas desenvolvidas para a criação do servidor. A Seção 4.4 discute as principais ferramentas responsáveis pela criação do cliente e também apresenta como é feita a autenticação no sistema, a coleta e importação dos dados, as estatísticas dos professores, departamentos e instituição. Deste modo, é possível ter uma visão geral das funcionalidades disponíveis para que os professores possam submeter seus relatórios.

4.1 Visão Geral

Os testes dos *endpoints* da API no servidor foram feitos através da ferramenta cliente de API REST Insomnia (2021.3.0), o que permitiu verificar o retorno das requisições enviadas ao servidor sem a necessidade de rodar o aplicativo móvel. Já os testes do aplicativo móvel foram feitos utilizando a ferramenta Expo através do seu aplicativo Expo Client.

A implantação do servidor Express (4.17.2), para fins de testes de aceitação, está em um ambiente de *cloud computing* e o provedor de serviço de virtualização utilizado foi o Heroku, que se comunica com o banco de dados MongoDB implantado no serviço MongoDB Atlas a partir de clusters. O aplicativo React Native (SDK-42.0.0) também foi implantada no Heroku que acessa via API REST as informações na aplicação. Para o envio das imagens de perfil foi utilizado o serviço Cloudinary (1.22.0). E para o envio dos e-mails de verificação de cadastro foi implantado utilizando a API SendGrid (7.2.1). A comunicação entre as partes pode ser visualizada na Figura 4.1.

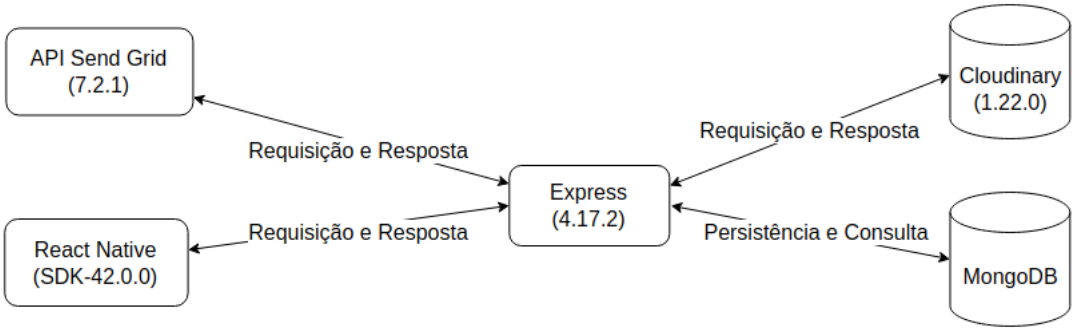


Figura 4.1: A comunicação com o servidor detalhada e integração com os serviços de email, upload de imagem, persistência, e importação de dados.

4.2 Modelagem dos Dados

A modelagem dos dados foi feita utilizando um banco de documentos ao invés de uma estrutura relacional para os dados. Nesta abordagem, os dados são descritos em uma árvore com objetos podendo ter estruturas diferentes uns dos outros. As Figuras 4.2 e 4.3 ilustra as coleções com seus relacionamentos e atributos. A coleção `Years` representa os anos de atividades exercidas pelos professores; a coleção `Departments` representa todos os departamentos da UFJF; a coleção `Axes` representa todos os eixos: ensino, pesquisa, extensão, atividade administrativa, capacitação e representação; a coleção `Categories` representa todas as categorias existentes da universidade; a coleção `Activities` representa todas as atividades desenvolvidas pelos professores do departamento; a coleção `tokens` representa o *token* de acesso do usuário para cada *endpoint* da aplicação; por fim, temos a coleção `Users`, que representa os professores usuários da aplicação.

As coleções `Activities` e `Categories` possuem um atributo chamado `details` que é um mapa chave-valor de utilizado que permite a criação de documentos aninhados com chaves arbitrárias. Cada categoria possui seu próprio conjunto de `details` estático, sugerido pelo aplicativo conforme ilustra a Figura 4.4a.

O `details` da coleção `Categories` é utilizado quando o usuário cria uma nova atividade. Os dados vêm em branco para serem preenchidos conforme ilustra a Figura 4.4b. Já o atributo `details` na coleção `Activities` é utilizado para serem inseridas novas chaves e novos valores no mapa de detalhes. Isso permite armazenar no documento um conjunto de dados dependente da categoria, que pode ser estendido conforme a necessi-

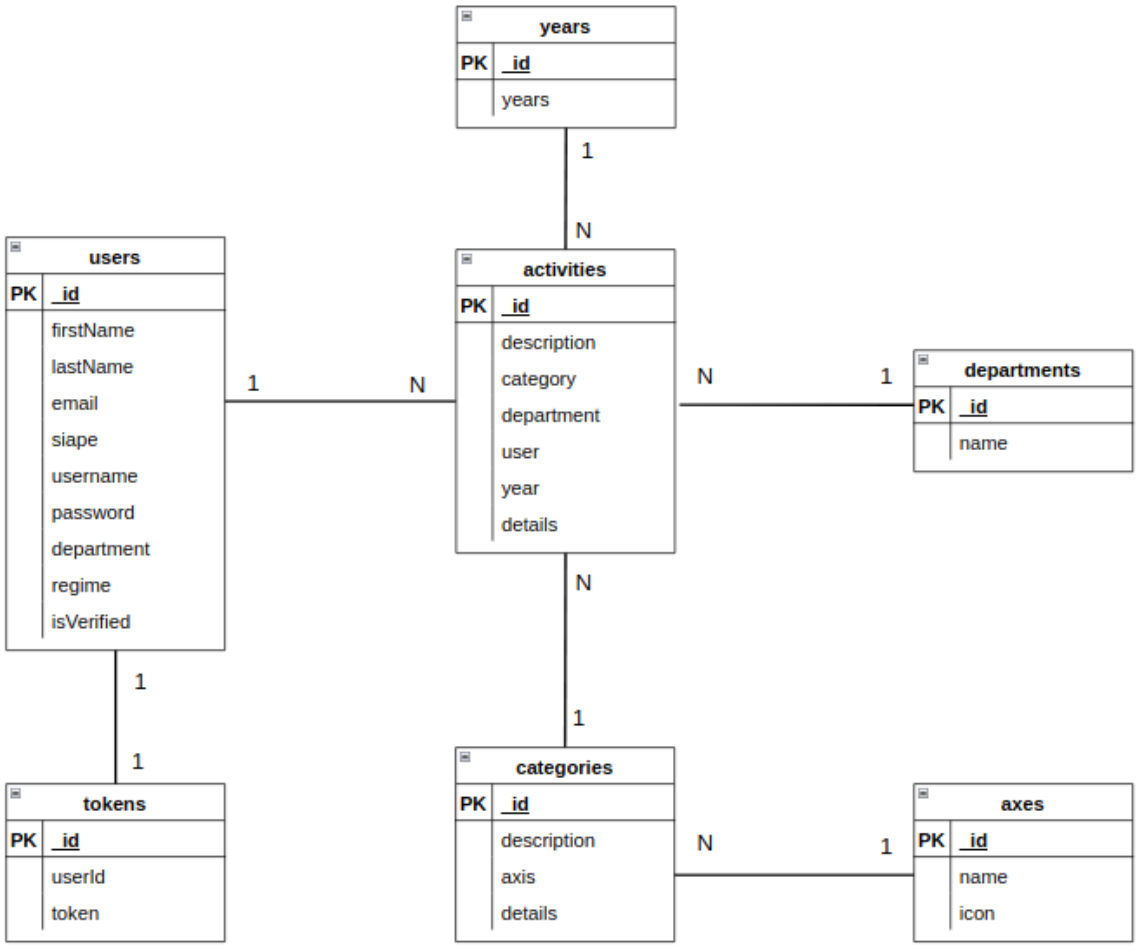


Figura 4.2: Diagrama representando as coleções, relações e cardinalidades entre documentos no MongoDB.

dade.

A Figura 4.5 ilustra o *JavaScript Object Notation* (JSON) do retorno da requisição a API através de uma visão melhorada das atividades pertencentes a uma categoria, onde é possível visualizar os detalhes na coleção `Category` e na coleção `Activities` conforme citado acima.

Foi utilizado a abordagem de banco de dados *Not Only SQL* (NoSQL) com o SGBD *MongoDB* que possui representação de objetos em formato de documentos JSON. Esses documentos possuem o gerenciamento de esquema através da biblioteca ODM *Mongoose* (5.9.22), permitindo um mapeamento objeto-documento e validações dos dados. A Figura 4.6 representa o *schema Mongoose* para as atividades.

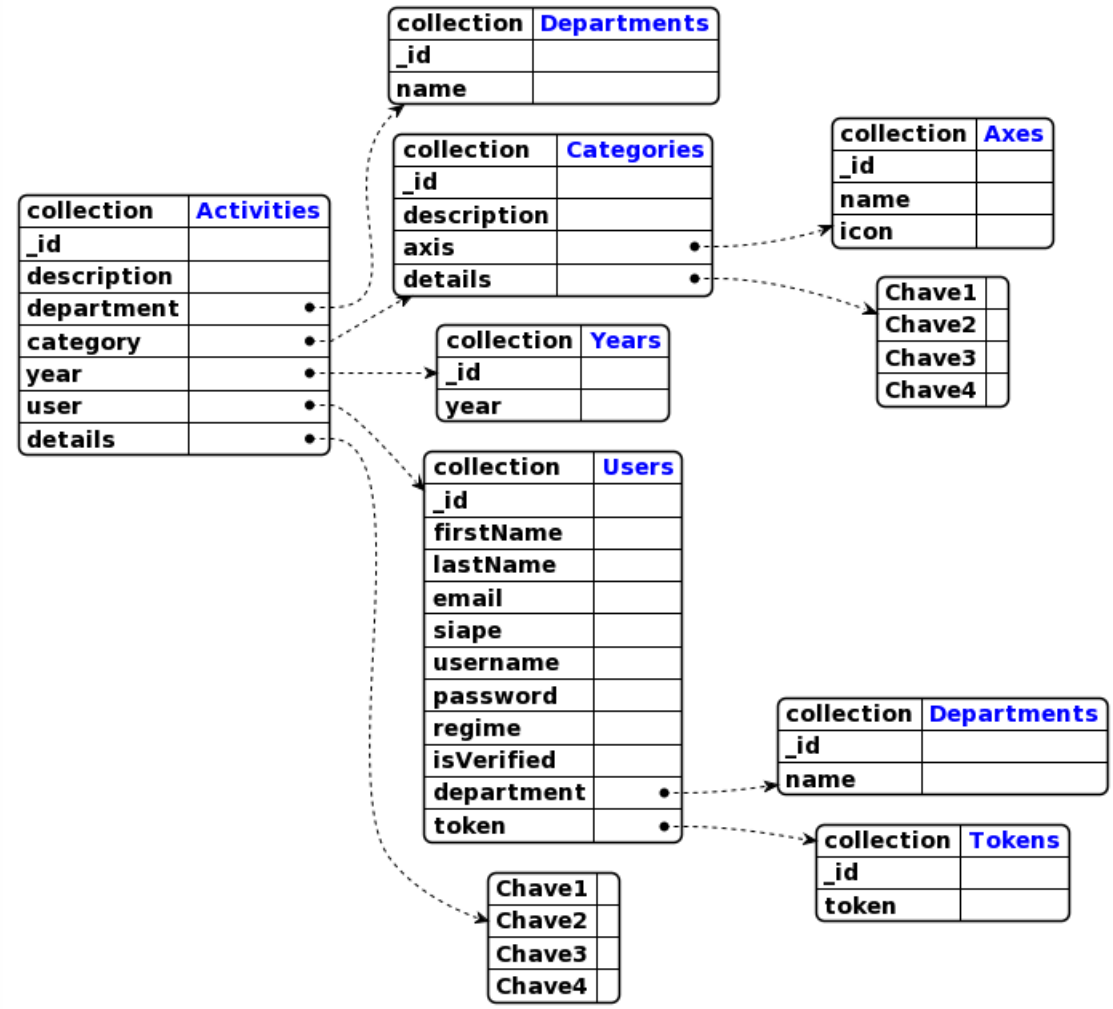


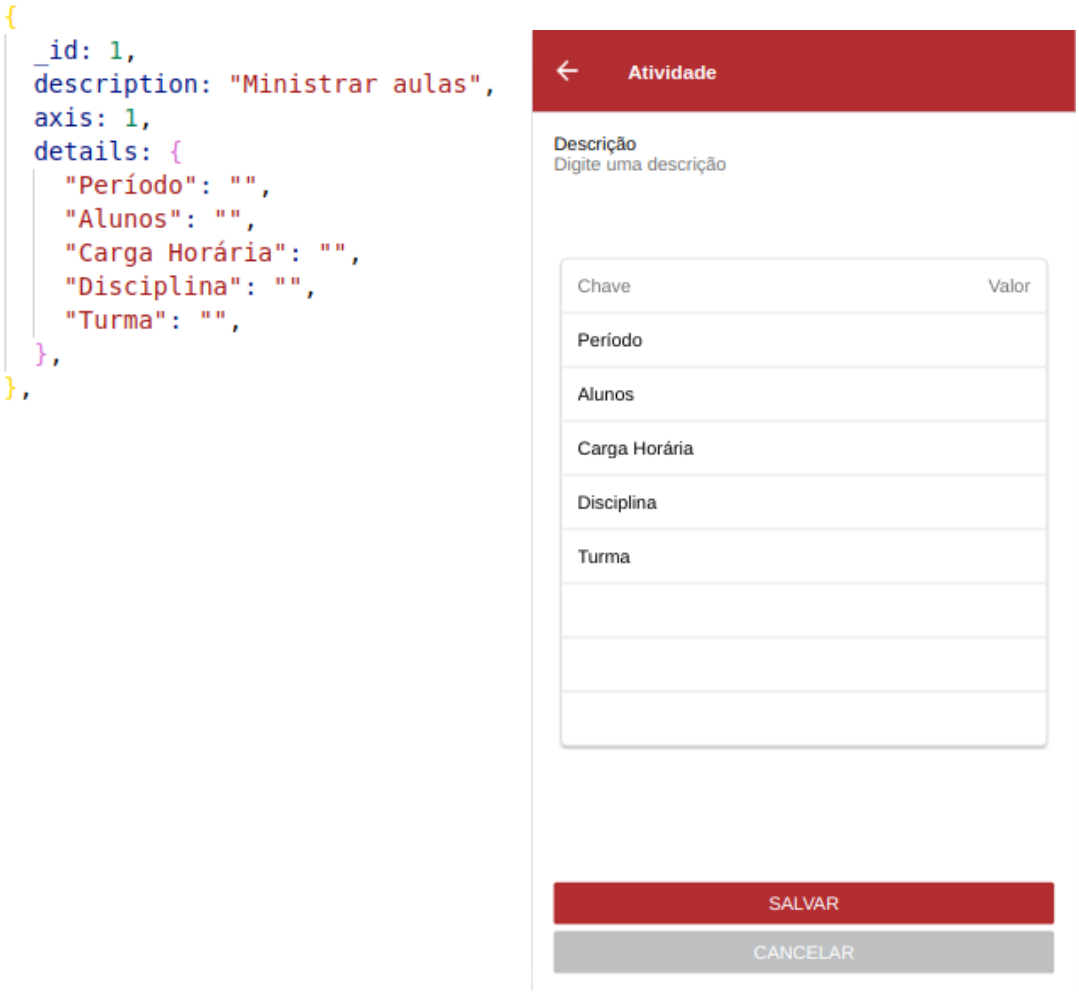
Figura 4.3: Diagrama representando as coleções e relações entre documentos no MongoDB.

4.3 Servidor

O servidor tem a responsabilidade de fazer o intermédio entre o aplicativo e o banco de dados, implementando todas regras de negócio e provendo a comunicação entre os clientes através de um padrão REST.

Para a construção do servidor foi utilizado o ambiente de execução *NodeJS* (14.18.1). Para as requisições oriundas dos clientes, todo o gerenciamento de rotas da aplicação, tratamento de exceções, gerenciamento das requisições HTTP, foram implementadas com o auxílio do *framework Express* (4.17.2).

Visando a segurança da aplicação, a API foi implementada com o esquema de autenticação JWT. Todas as rotas utilizam o *middleware* de autenticação *Passport* (0.4.1) e o módulo *Passport JWT* (4.0.0), que fica responsável por ler o JWT do cabeçalho da requisição HTTP através do *authorization* com esquema *bearer*, conforme ilustrado na



(a) JSON representando os detalhes de uma categoria.

(b) Captura da tela de criação de atividade.

Figura 4.4: Detalhes por categoria: a) mapa de detalhes e b) campos na tela de criação.

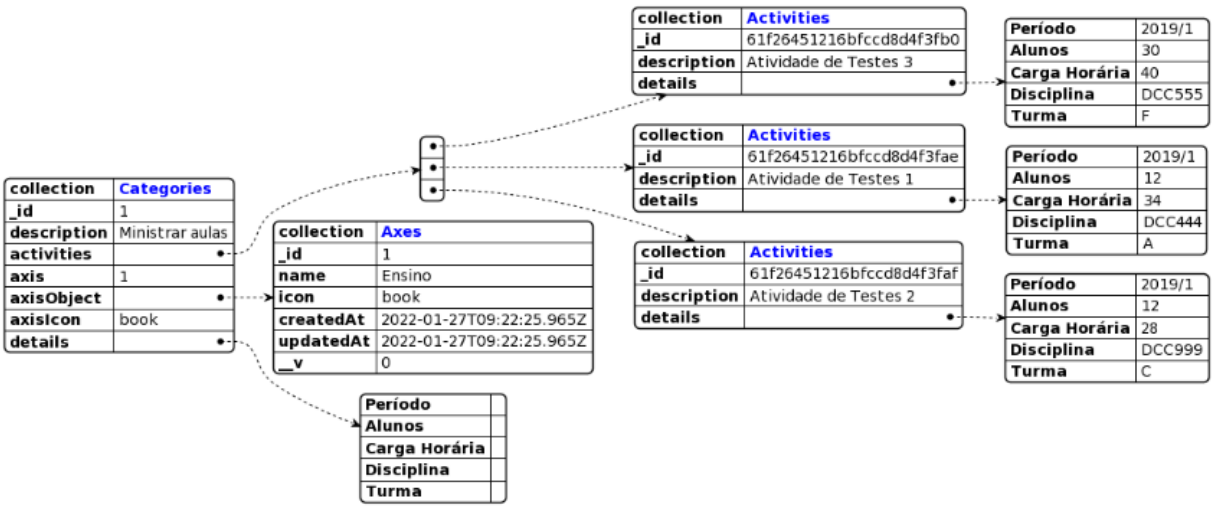


Figura 4.5: Diagrama representando o JSON retornado da API na consulta de atividades por categoria.

```
const mongoose = require("mongoose");

const ActivitySchema = new mongoose.Schema(
  {
    user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "User",
      required: true,
    },

    department: { type: mongoose.Schema.Types.Number, ref: "Department" },

    category: { type: mongoose.Schema.Types.Number, ref: "Category" },

    year: { type: mongoose.Schema.Types.Number, ref: "Year" },

    description: {
      type: String,
      required: "Description is required",
    },

    details: {
      type: Map,
      of: String,
    },

  },
  { timestamps: true }
);

module.exports = mongoose.model("Activity", ActivitySchema);

```

Figura 4.6: Mapeamento ODM em *schema* Mongoose para as atividades.

Figura 4.7.

4.4 Cliente

Com a necessidade de consumir e interagir com as funcionalidades e dados disponíveis pela API, foi criado um aplicativo móvel utilizando uma plataforma híbrida. A principal função do aplicativo é a coleta dos dados do RIT. Portanto, ao professor é disponibilizado uma interface que o permite navegar pelos eixos e relatar cada atividade, em uma categoria específica.

A visualização dos dados relatados, pode ser realizada diretamente pela navegação, ou agrupadas em uma série de gráficos. Os dados agrupados em gráficos por: categorias das atividades; de evolução ao longo dos anos; de quantidade de atividades por

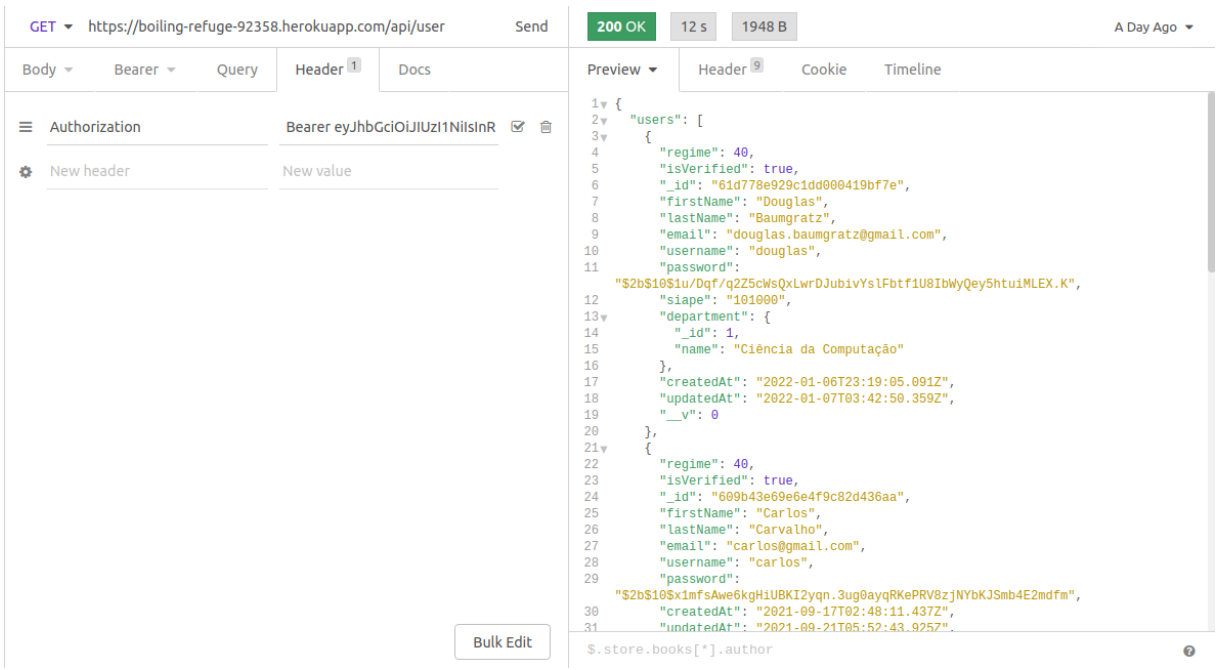


Figura 4.7: Captura da tela do Insomnia representando um exemplo de requisição com *bearer token*

departamentos e instituição; por detalhes (por exemplo, alunos que o professor lidou diretamente, por disciplinas ou orientações) e comparativos em relação a outros professores do seu departamento e instituição.

Os dados podem ser entrados diretamente pela interface do aplicativo, ou serem importados via planilhas organizadas pelo departamento, ou através dos arquivos gerados da plataforma Lattes. Essas importações podem ser expandidas e adaptadas a cada caso, conforme a necessidade de customização do aplicativo para a instituição ou mudanças de regulamentos e leis.

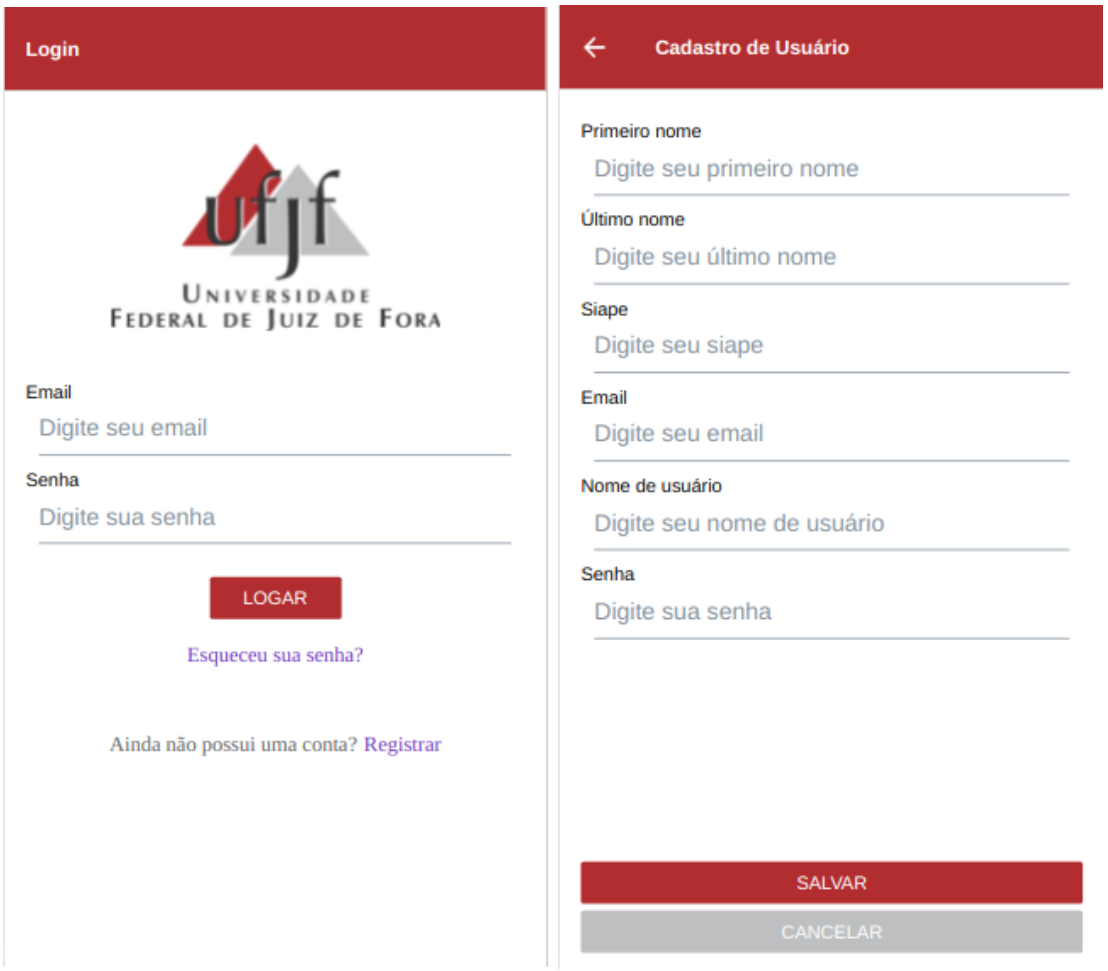
O cliente foi construído utilizando o *framework React Native* (*SDK*-42.0.0) através da ferramenta *Expo* (43.0.3). A biblioteca de componentes utilizada para criação dos gráficos dos dados foi a *Victory Native* (35.3.3)[41], que permite uso modular para *React* e *React Native*. A biblioteca responsável pelos componentes do aplicativo é a *React Native Elements*[42].

[41] O site oficial do Victory Native encontra-se disponível em ⟨https://formidable.com/open-source/victory/⟩

[42] O site oficial do React Native Elements encontra-se disponível em ⟨https://reactnativeelements.com/⟩

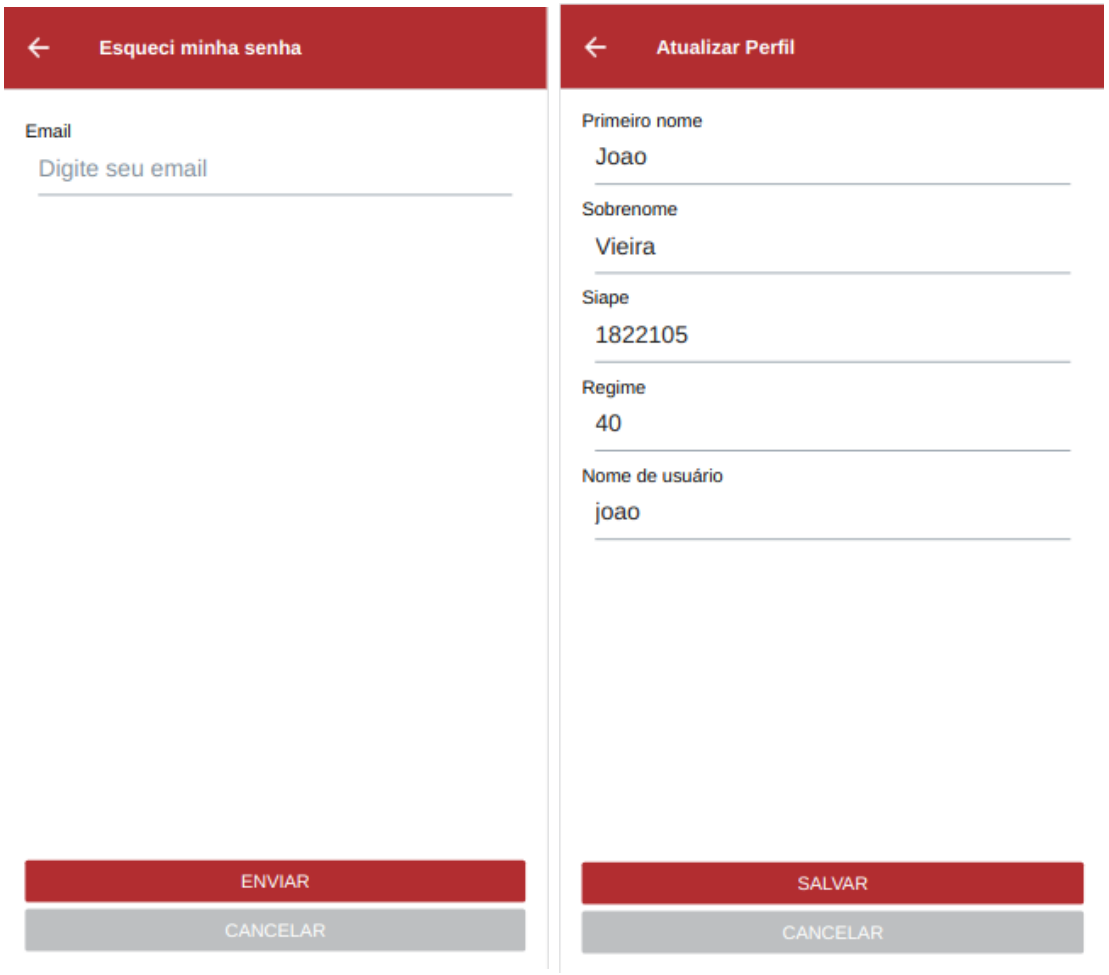
4.4.1 Autenticação

Como funcionalidades transversais, o registro e a identificação no aplicativo permitem a criação de um novo usuário com verificação de cadastro por e-mail, autenticação e recuperação de senha, atualização dos detalhes do perfil e navegação pelo aplicativo. A Figura 4.8a, representa a tela inicial do aplicativo, onde através do e-mail e senha o usuário faz o login na aplicação. Para o cadastro no aplicativo o usuário deve informar os campos: primeiro nome, último nome, siape, e-mail, nome de usuário e senha conforme ilustra a Figura 4.8b. Conforme a Figura 4.9a, através do e-mail do usuário é enviado um link para redefinição de senha no aplicativo. Os atributos de usuário como: primeiro nome, sobrenome, siape, regime e nome de usuário são editáveis, conforme ilustra a Figura 4.9b. As demais funcionalidades são abordadas nas seções subsequentes.



(a) Captura da tela de login. (b) Captura da tela de cadastro.

Figura 4.8: Telas das funcionalidades de: a) autenticação e b) cadastro de usuários.



(a) Captura da tela de redefinição de senha. (b) Captura da tela de edição de perfil.

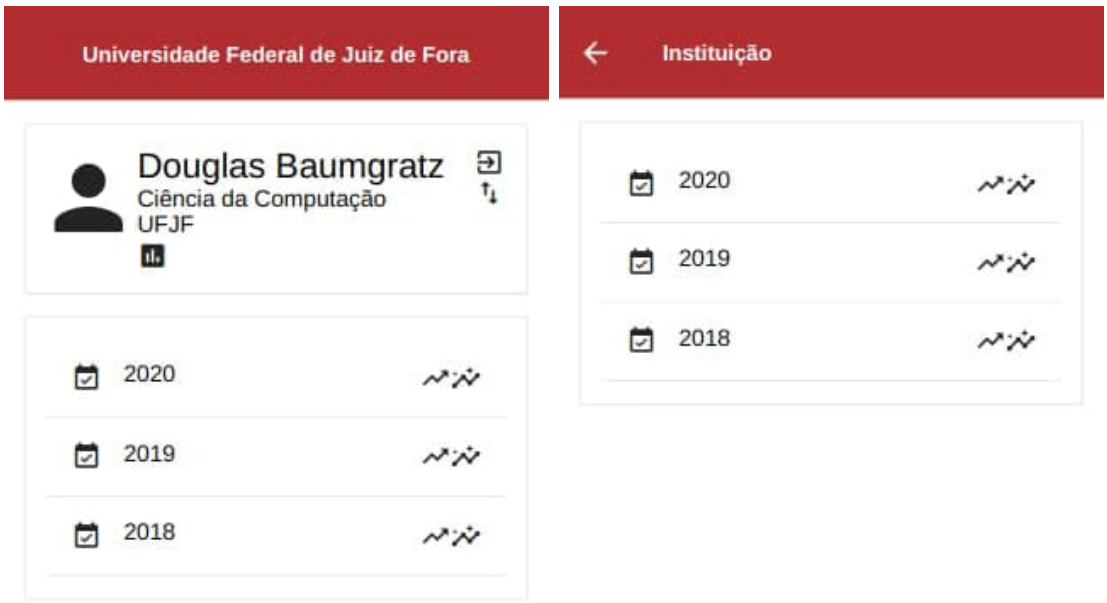
Figura 4.9: Telas das funcionalidades de: a) redefinição de senha e b) edição de perfil.

4.4.2 Coleta de Dados

A tela inicial do aplicativo possui uma listagem por anos das atividades desenvolvidas e permite ao usuário acessar as funcionalidades de importações, *Create, Read, Update, Delete* (CRUD) de atividades e estatísticas dos professores conforme ilustra a Figura 4.10a. A tela da instituição também possui um filtro por anos e permite ao usuário visualizar estatísticas de todos professores, departamentos e instituição conforme ilustra a Figura 4.10b.

A coleta dos dados, ou seja, o relato do professor das atividades no ano, é realizada a partir da escolha do ano corrente, que conduz para uma listagem de categorias, organizadas pelo eixo. Conforme pode ser observado pela Figura 4.11a, a direita do nome de cada categoria, há um botão para criação de uma atividade para a categoria em questão.

Ao adicionar uma atividade, uma nova tela é aberta para a entrada da descrição



(a) Tela inicial do aplicativo. (b) Tela da instituição.

Figura 4.10: Capturas de telas do aplicativo: a) tela inicial e b) tela instituição.

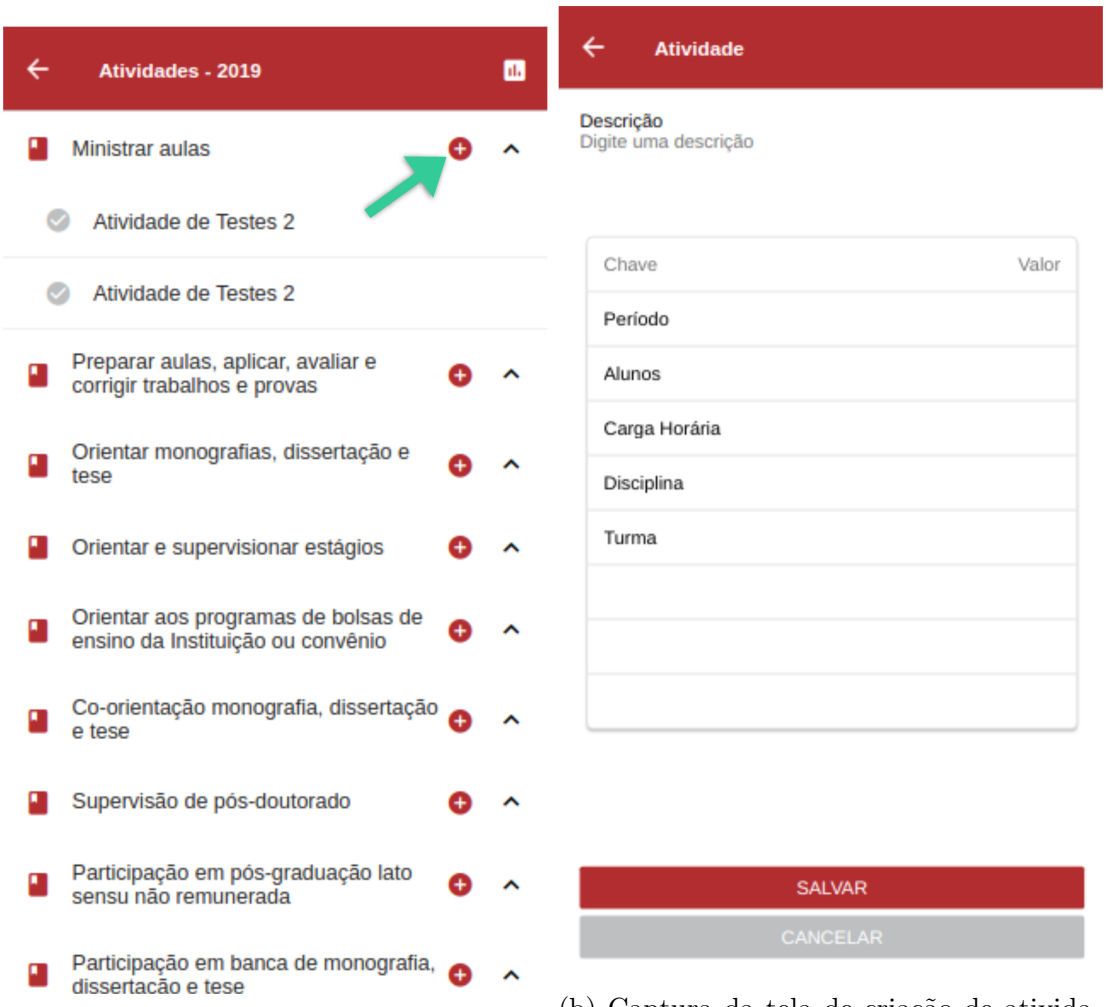
e dos detalhes conforme ilustra a Figura 4.11b. Os detalhes seguem o modelo chave-valor e são dependentes da categoria escolhida. Novas chaves podem ser adicionadas pelo usuário no momento da criação de uma atividade.

Para edição e exclusão das atividades foi utilizado a API de animação *React Native Gesture Handler* (1.10.2), através do componente *Swipeable*. O que permitiu criar um gesto de arrasto para esquerda e para direita, com as ações de edição e exclusão sendo exibidas, respectivamente. A Figura 4.12a ilustra a ação de edição de uma atividade e a Figura 4.12b ilustra a exclusão de uma atividade.

Existem um conjunto de *endpoints* responsáveis por realizar o CRUD de atividades de um professor. Um exemplo de resposta da API para as atividades de um professor em um determinado ano pode ser visualizado na Figura 4.14.

4.4.3 Importação de dados

A importação dos dados a partir de planilhas de anos anteriores, utiliza uma ferramenta de importação desenvolvida em JavaScript que utiliza a API do Google Drive para realizar a



(a) Captura da tela onde se encontra botão para adicionar uma categoria.

(b) Captura da tela de criação de atividades.

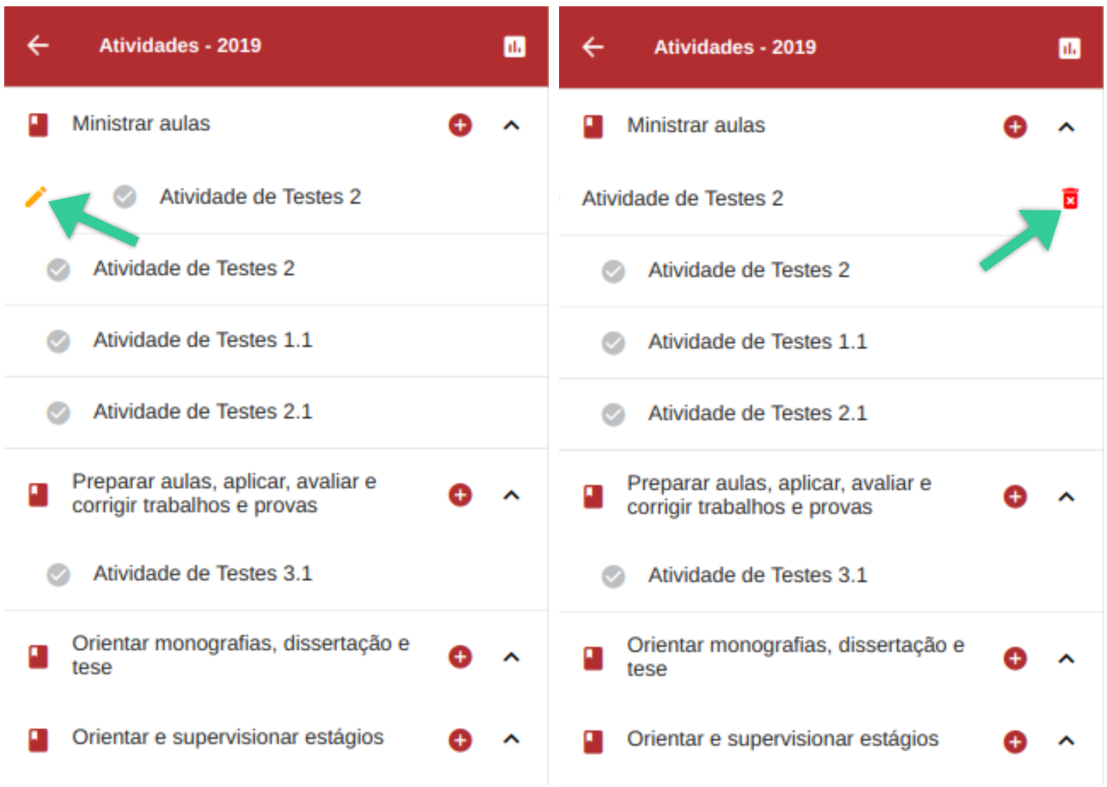
Figura 4.11: Telas das funcionalidades de: a) botão de criar atividade e b) criação de atividades.

transformação das planilhas utilizadas em coletas de RITs de anos anteriores em arquivos JSON.

Esses arquivos JSON são utilizados pela aplicação, onde, a pedido do usuário no aplicativo, todas as atividades são importadas para a base de dados.

A planilha do DCC no ano de 2019, foi registrada com os eixos pesquisa e extensão juntos. Foi feita uma adaptação no código fonte da ferramenta de importação no projeto do professor, para modelar os dados oriundos das planilhas conforme o *schema* definido no banco de dados da aplicação.

Também foi desenvolvida a coleta através da importação do arquivo XML com-



(a) Captura da tela de edição de atividades. (b) Captura da tela de exclusão de atividades.

Figura 4.12: Telas das funcionalidades de: a) edição de atividades e b) exclusão de atividades.

pactador gerado na plataforma *Lattes*[43]. O professor pode enviar o arquivo diretamente pelo aplicativo e o servidor realiza a importação para o banco de dados.

Na atual implementação, há a limitação de apenas as atividades relacionadas aos artigos publicados serem importados. Entretanto, todo o código da importação do *Lattes* foi construído modularizado, o que permite que novas atividades de outras categorias sejam implementadas em trabalhos futuros. Ambas importações necessitam do ano de interesse na importação, mas a do *Lattes* necessita do envio do arquivo conforme ilustra a Figura 4.13.

4.4.4 Estatísticas Individuais dos Professores

Baseado nas quantidades de atividades cadastradas pelo professor, foi criado um gráfico de radar de três variáveis (professor, professores do departamento e professores da instituição) conforme ilustra a Figura 4.15. O professor ilustrado em verde representa a

[43]O link oficial do site pode ser encontrado em ⟨https://lattes.cnpq.br/⟩

Figura 4.13: Captura da tela de importação das atividades DCC e *Lattes*.

quantidade de atividades do professor logado. Os professores do departamento ilustrado em laranja, representa a média de atividades do departamento. Os professores da instituição, representa a média de atividades da instituição.

Através de um gráfico de barras, é possível acompanhar a quantidades de atividades por categorias, conforme pode ser visto na Figura 4.16a. À esquerda fica o nome da categoria e à direita quantitativo de atividades para o ano escolhido.

Um segundo gráfico de barras foi criado para representar o total de atividades por eixos conforme pode ser visto na Figura 4.16b.

É possível visualizar esses dados de todos professores da instituição, bastando clicar no ícone de gráfico na frente do nome de cada professor conforme ilustra a Figura 4.17.

Foram criados dois gráficos de evolução para uma visão geral baseada em anos.

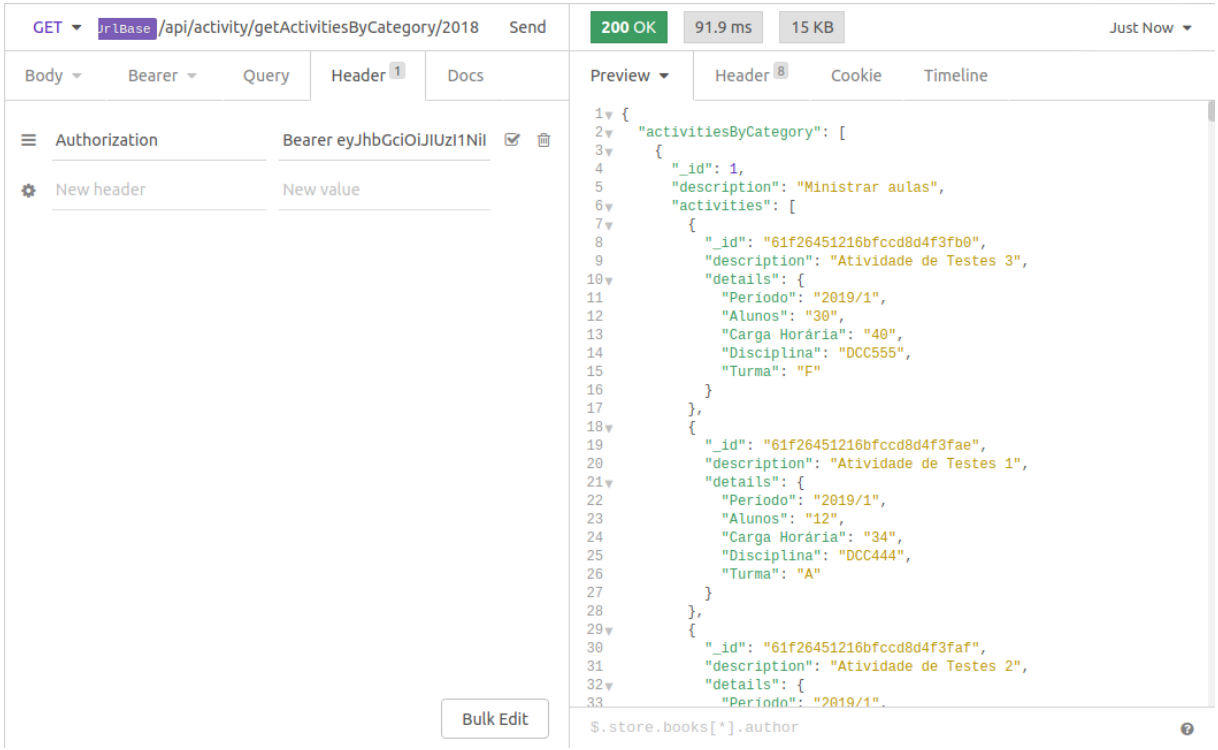


Figura 4.14: Captura da tela do Insomnia representando um exemplo de requisição com retorno das atividades de um professor em um determinado ano.

O primeiro gráfico conforme ilustra a Figura 4.18a representa a quantidade de atividades desenvolvidas durante os anos. A legenda representa os eixos, onde o eixo y representa a quantidade de atividades e o eixo x representa os anos. O segundo gráfico conforme ilustra a Figura 4.18b representa a quantidade de alunos no decorrer dos anos. O eixo y representa a quantidade de alunos e o eixo x representa os anos. No estudo de caso apresentado, os dados das planilhas do ano de 2021 não foram inseridos na base, pois o processo não foi concluído no DCC.

Como exemplo de cômputo de dados pelos detalhes das atividades, foi criado um gráfico de barras para representar a quantidade de alunos orientados ou para os quais o professor lecionou no ano. Esse tipo de análise é realizada a cada caso sobre os dados de detalhes de algumas categorias, neste caso, no número de alunos. As Figuras 4.19a e 4.19b representam a distribuição de alunos com os quais o professor interagiu.

Conforme ilustra a Figura 4.20, é possível verificar o retorno de uma requisição a API com a quantidade de atividades por ano do professor. Todos os dados e estatísticas dos professores são disponibilizados na forma de serviços.

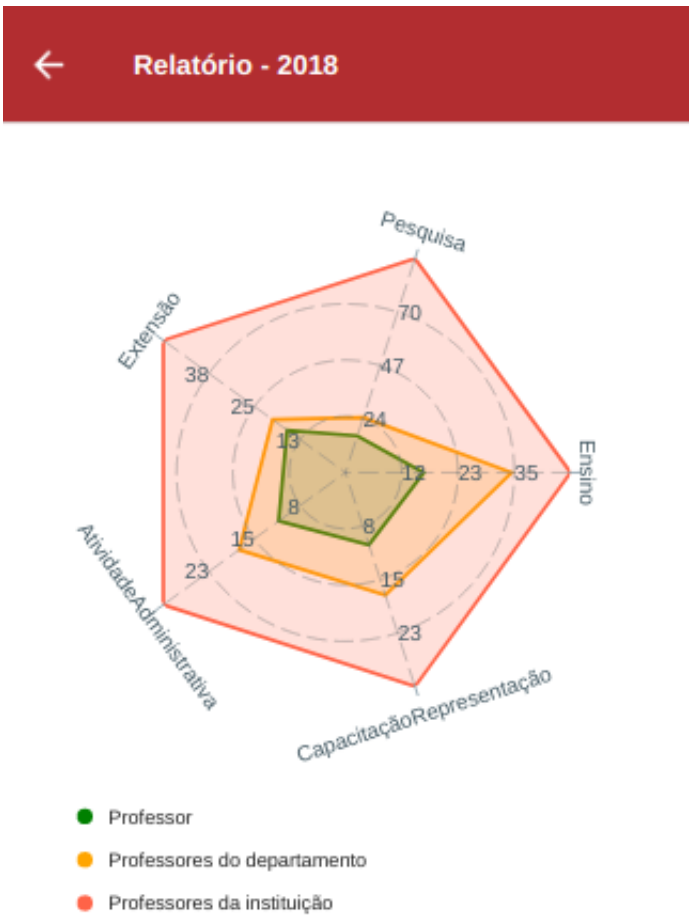


Figura 4.15: Captura da tela de relatório individual do professor.

4.4.5 Estatísticas dos Departamentos

Também foram implementadas visualizações para os departamentos, permitindo avaliar como a sua produção está evoluindo em uma visão global.

Mantendo a estrutura, a quantidade de atividades por categoria e por eixos de um departamento podem ser observadas nas Figuras 4.21a e 4.21b. Um novo gráfico de radar permite comparar a quantidade de atividades do departamento em relação à média da instituição. O departamento é representado em verde e a instituição em laranja conforme ilustra a Figura 4.22.

Conforme ilustra a Figura 4.23, é possível verificar o retorno de uma requisição a API com a quantidade de atividades por categoria de um departamento. Todos os dados e estatísticas dos departamentos são disponibilizados na forma de serviços.

(a) Captura da tela de relatório individual do professor com atividades por categorias. (b) Captura da tela de relatório individual do professor com atividades por eixo.

Figura 4.16: Telas das funcionalidades de: a) quantidade de atividades por categoria e b) quantidade de atividades por eixos.

4.4.6 Estatísticas da Instituição

Assim como nos gráficos dos departamentos, foram desenvolvidos dois gráficos em barra e um gráfico de radar para representar estatísticas da instituição.

Conforme ilustra a Figura 4.24a é possível visualizar o gráfico da quantidade de atividades por categoria, onde o eixo y representa o nome da categoria e a quantidade é representada à direita de cada barra. Assim como no gráfico da quantidade por categoria, foi desenvolvido um gráfico para representar a quantidade de atividades por eixos, conforme ilustra a Figura 4.24b, onde o eixo y representa o nome do eixo e a quantidade é representada à direita de cada da barra.

Outro gráfico em radar também foi desenvolvido para a instituição, porém com apenas uma variável, representando o total de atividades em todas as categorias conforme

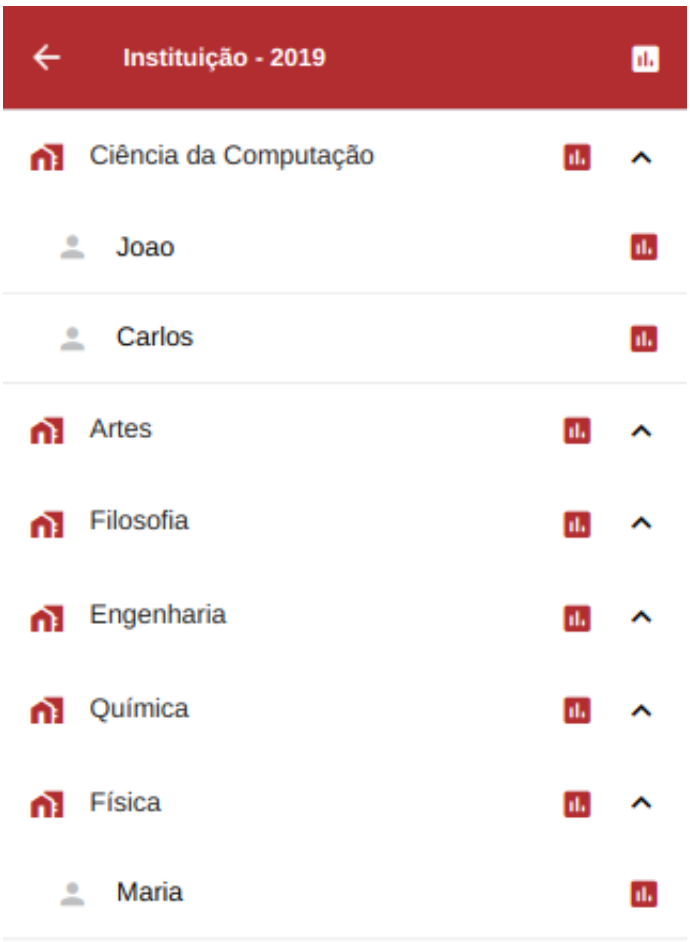
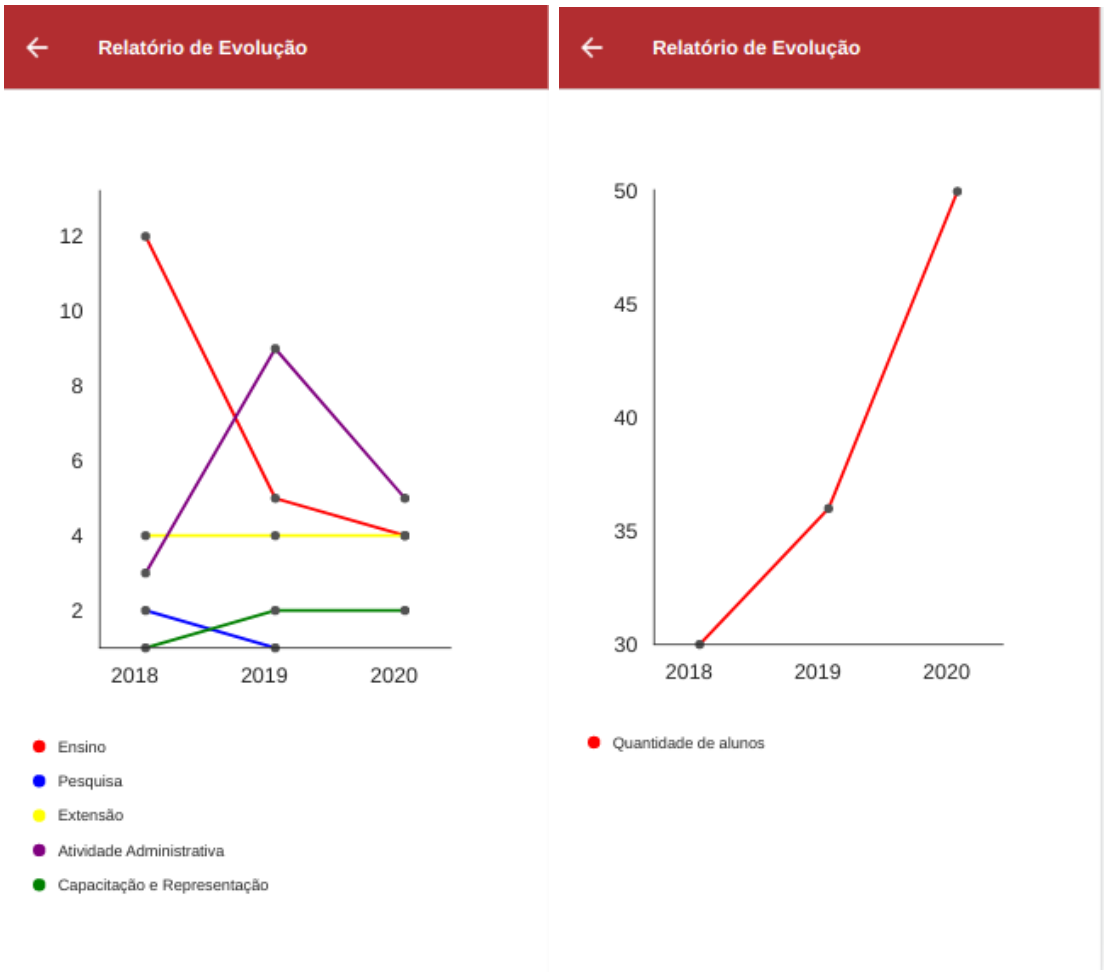


Figura 4.17: Captura da tela de professores por departamento.

ilustra a Figura 4.25.

Conforme ilustra a Figura 4.26, é possível verificar o retorno de uma requisição a API com a quantidade de atividades por eixos da instituição. Todos os dados e estatísticas da instituição são disponibilizados na forma de serviços.

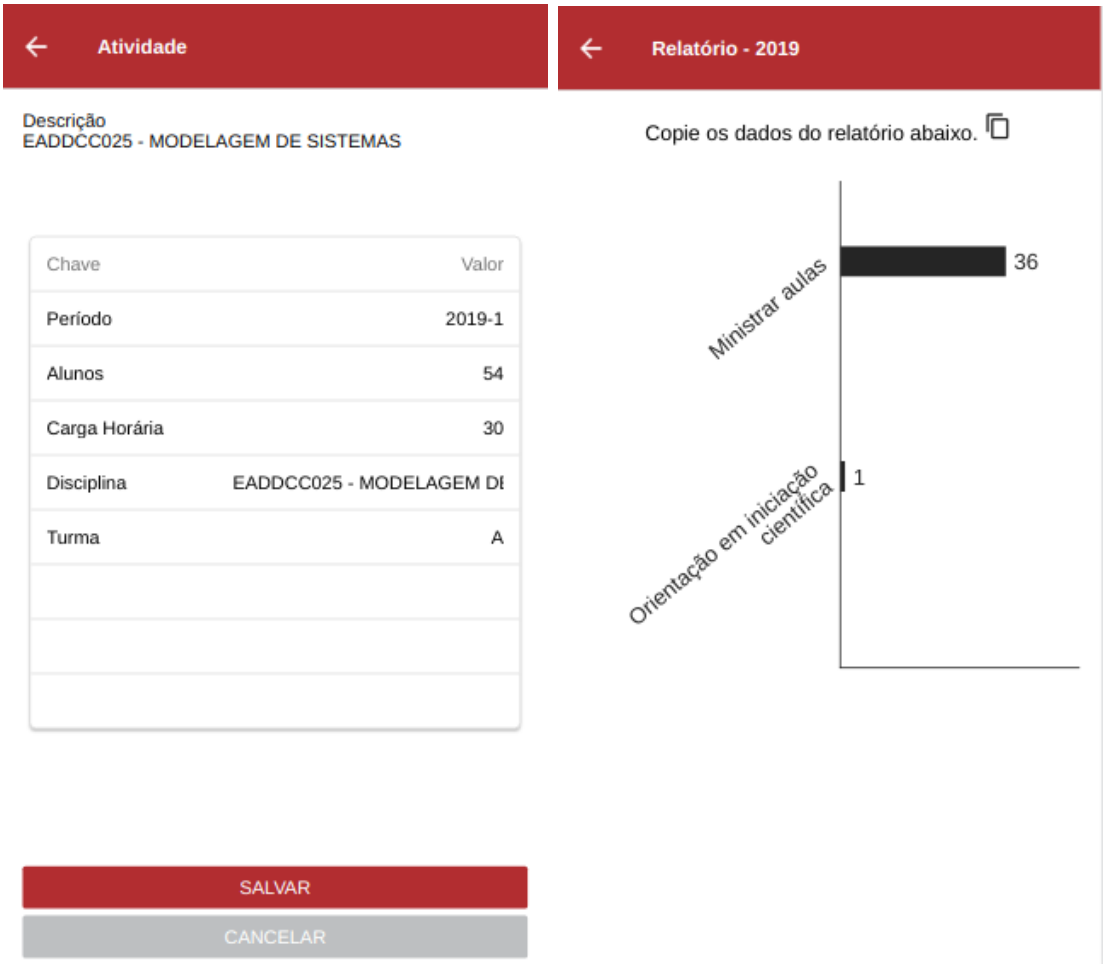
Este capítulo apresentou o planejamento, as ferramentas utilizadas e o estudo de caso do desenvolvimento de um novo aplicativo móvel de coleta e avaliação de RIT docente da UFJF.



(a) Captura da tela de relatório baseada em eixos.

(b) Captura da tela de relatório baseada no somatório de alunos em cada atividade.

Figura 4.18: Telas das funcionalidades de: a) evolução entre a quantidade de atividades nos eixos e b) evolução na quantidade de alunos em atividades exercidas.



(a) Detalhes das atividades. (b) Relatório do professor.

Figura 4.19: Captura das telas de: a) detalhes de atividades e b) distribuição dos alunos.

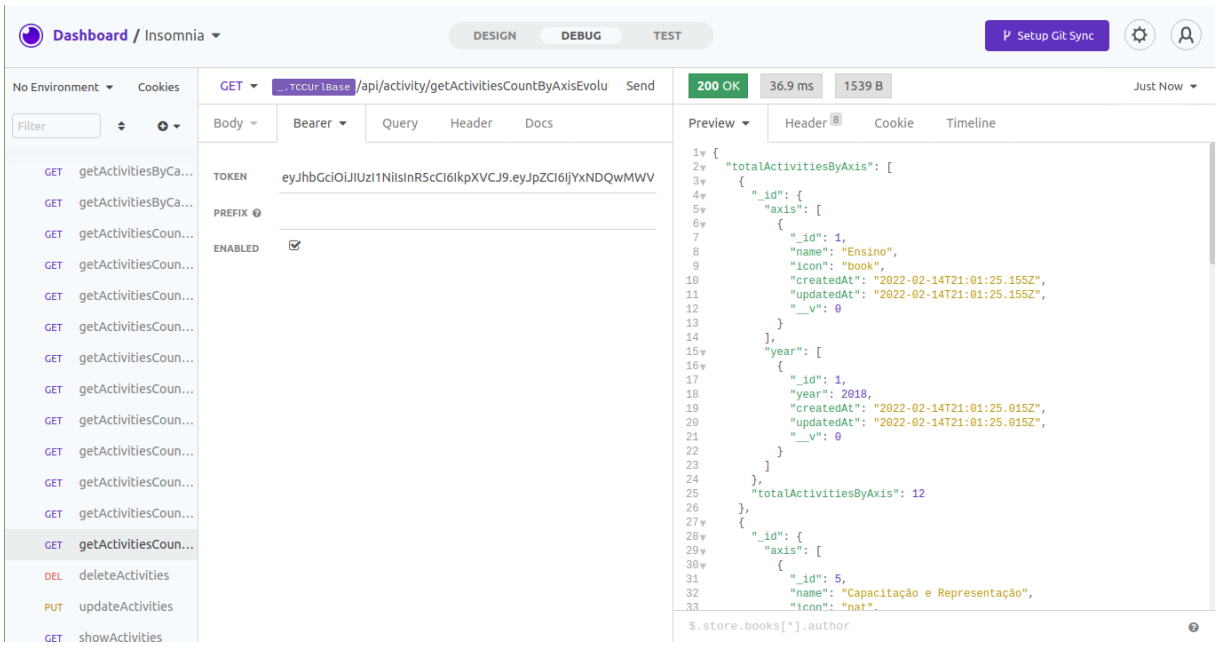
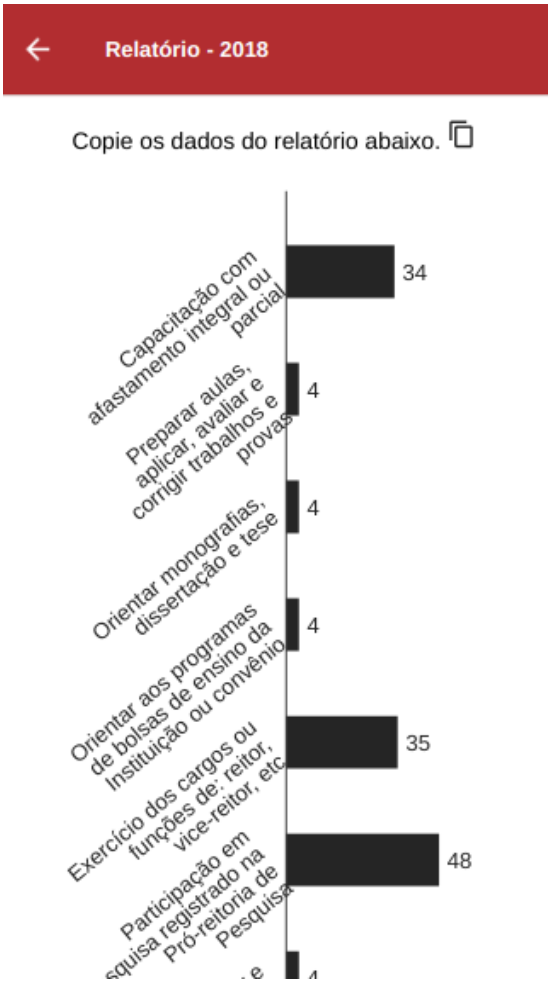
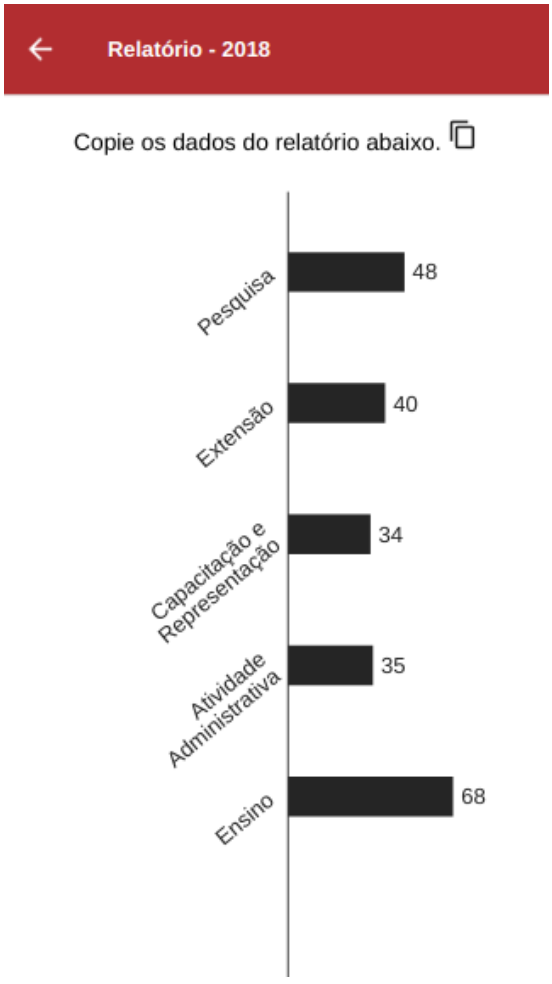


Figura 4.20: Captura da tela do Insomnia representando um exemplo de requisição a API com retorno da quantidade de atividades por ano do professor.



(a) Captura da tela de relatório do departamento com quantidade de atividades por categoria.



(b) Captura da tela de relatório do departamento com quantidade de atividades por eixo.

Figura 4.21: Telas dos gráficos do departamento: a) quantidade de atividades por categoria e b) quantidade de atividades por eixos.

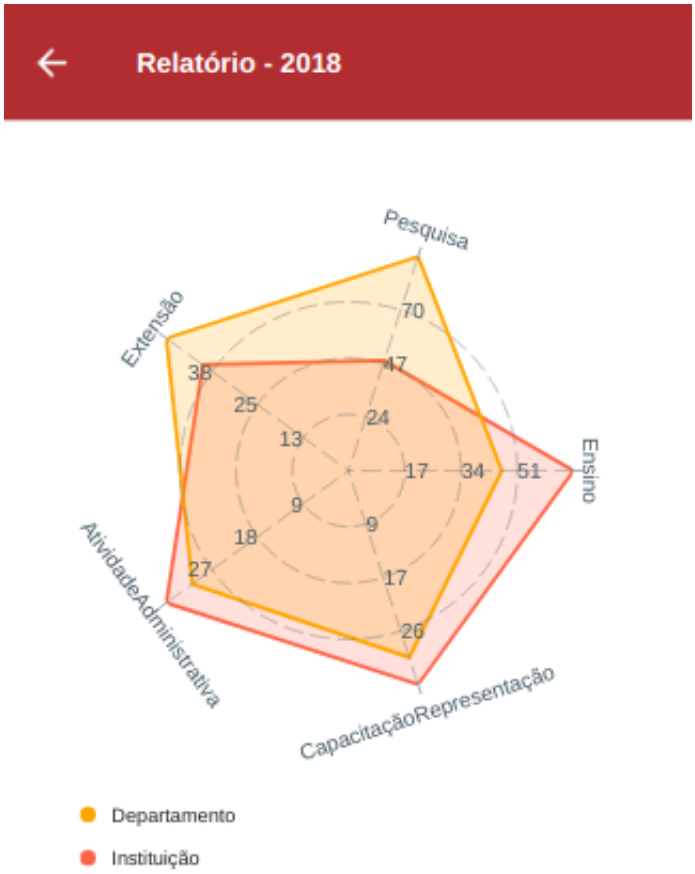


Figura 4.22: Captura da tela de relatório do departamento.

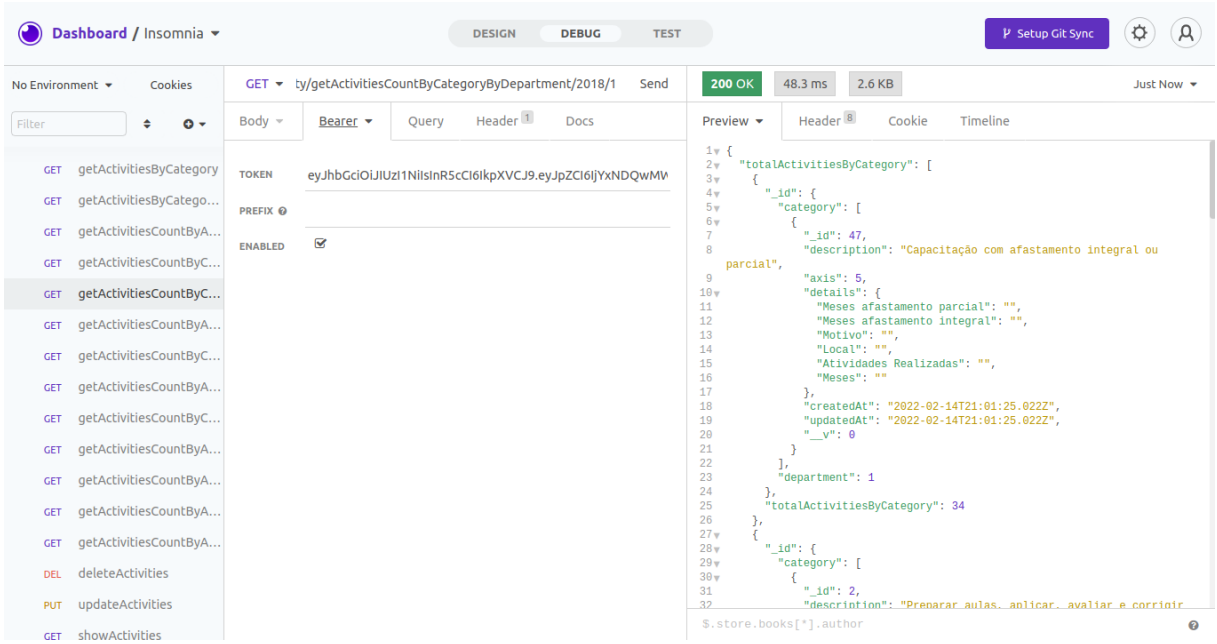
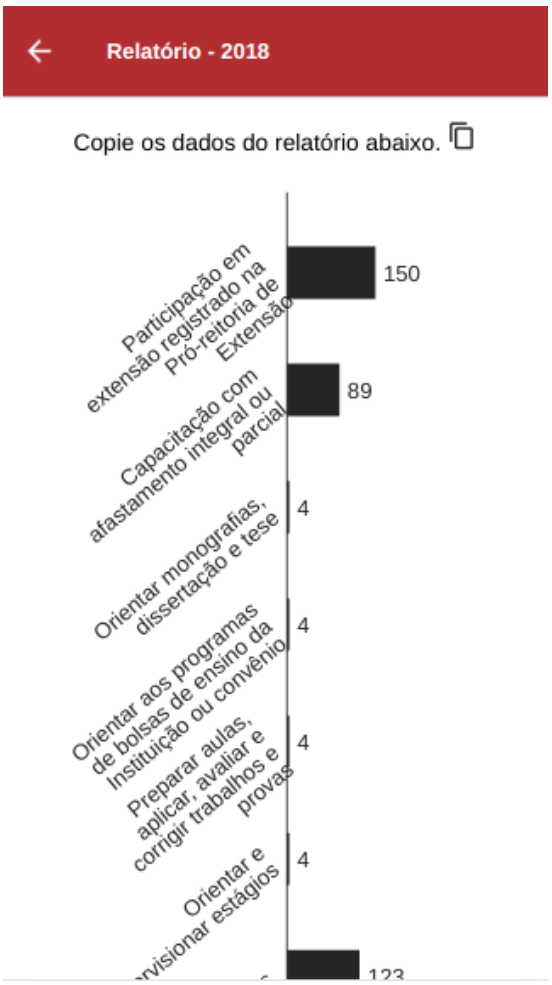
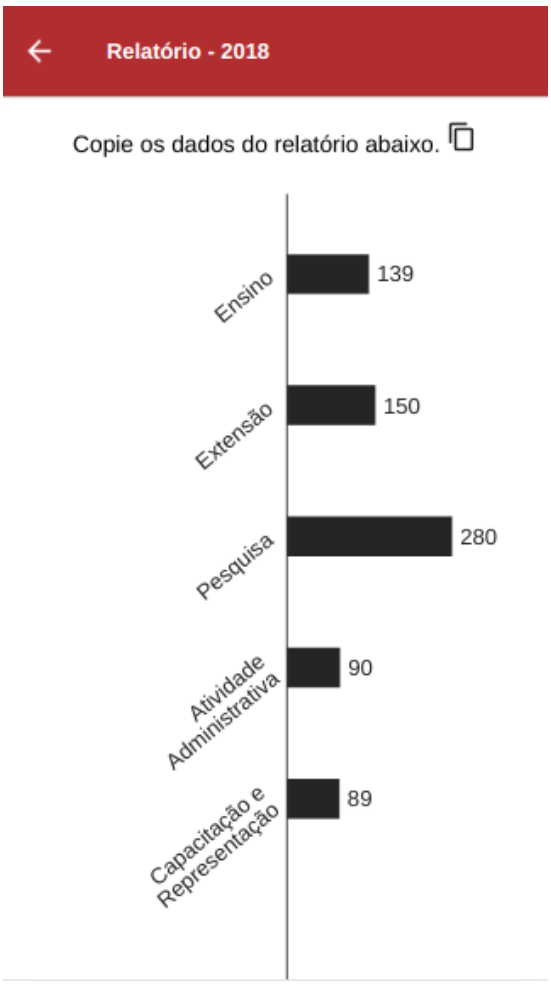


Figura 4.23: Captura da tela do Insomnia representando um exemplo de requisição a API o com retorno da quantidade de atividades por categoria de um departamento.



(a) Captura da tela de relatório da instituição com quantidade de atividades por categoria.



(b) Captura da tela de relatório da instituição com quantidade de atividades por eixo.

Figura 4.24: Telas dos gráficos da instituição: a) quantidade de atividades por categoria; e b) quantidade de atividades por eixos.

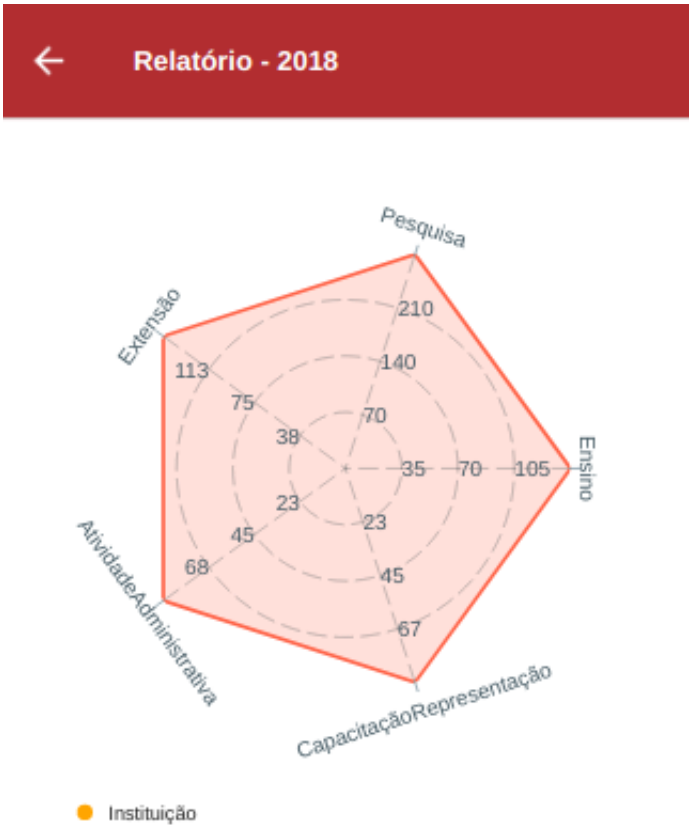


Figura 4.25: Captura da tela de relatório da instituição.

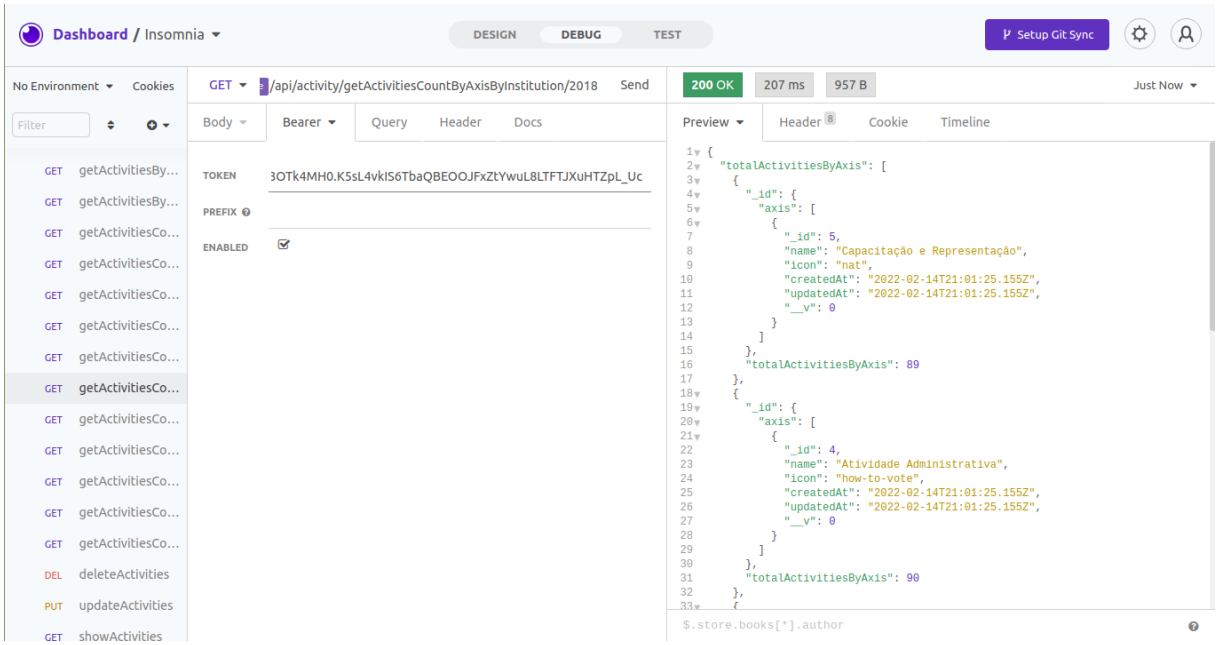


Figura 4.26: Captura da tela do Insomnia representando um exemplo de requisição a API com o retorno da quantidade de atividades por eixos da instituição.

5 Considerações Finais

Os professores do magistério superior apresentam anualmente um Relatório Individual de Trabalho (RIT) com suas atividades realizadas. Um problema enfrentado pelas instituições na elaboração do RIT, é a falta de uma plataforma unificada para envio e processamento dos relatórios dos professores ao longo dos anos.

Logo, se todas as informações forem inter-relacionadas, elas podem permitir tomar decisões mais assertivas, facilitando o trabalho dos professores, assembleia docente e chefia ao avaliar o desempenho anual de cada docente, conhecer o seu corpo docente e identificar problemas e propor algum plano estratégico de melhorias.

Este trabalho apresentou o desenvolvimento de um sistema onde são feitas coletas das atividades do ano corrente e coletas através de importações de planilhas de anos anteriores e dados do *Lattes*. Os dados são compilados em gráficos, a fim de deixar compreensível a sua descrição e facilitar a avaliação do trabalho desenvolvido pelos professores ao longo da carreira.

Com posse de todas essas informações, é possível ter uma visão geral de quais foram as atividades mais desenvolvidas, no ano corrente ou ao longo dos anos, quais eixos ou categorias possuem maior esforço, e como o professor se posiciona dentro do departamento e dentro da instituição.

A solução proposta permitiu importar os dados das planilhas de anos anteriores do DCC da UFJF. Foi feita uma modelagem dos dados e os mesmos foram organizados por categorias, eixos e anos, respectivamente. Dessa forma a evolução das atividades produzidas no departamento puderam ser detalhadas ao longo dos anos nos quais a coleta e importação foram realizados.

Adicionalmente à implementação de ações para criação, edição e exclusão de informações, a importação dos dados de publicações foi implementada para receber o arquivo gerado do da plataforma *Lattes*.

Os gráficos desenvolvidos no sistema, envolvendo dados de diversos RITs, fornecem base para o departamento tomar decisões pautadas na evolução dos dados, e não

só no RIT anual como é feito atualmente. Logo, isso nos leva a crer que o objetivo deste trabalho foi atingido e em trabalhos futuros pode ser melhorado, como, por exemplo, dar continuidade nas implementações de outras atividades oriundas do *Lattes*.

A aplicação desenvolvida fornece uma API onde qualquer outra aplicação cliente possa consumir e criar outras aplicações com base nos dados coletados. Logo, o objetivo secundário foi atingido, pois, fornece uma interface para que outros aplicativos possam ser criados e possam consumir os dados e relatórios disponíveis na API.

Com os dados oriundos das planilhas do DCC e dos dados públicos disponibilizados pela plataforma *Lattes* foi possível realizar o estudo de caso sobre os dados de um departamento real, onde conjecturamos que o objetivo foi atingido.

5.1 Limitações e Trabalhos Futuros

A importação dos dados oriundos do *Lattes*, por exemplo, se restringiram aos artigos no ano de interesse. Entretanto, o currículo *Lattes* é a principal fonte de produção no Brasil, e muito mais informações podem ser extraídas, como projetos, orientações e participações em bancas. Como o módulo de importação permite ser expandido, adicionar novas categorias conforme o interesse do departamento e instituição é um processo que pode ser feito em trabalhos futuros.

As atividades relatadas devem corresponder com a estimativa no PIT, entretanto, o processo de avaliação e confronto com o PIT também não foi implementado. O módulo de coleta do PIT também é uma tarefa para trabalhos futuros.

Alguns departamentos e institutos instituem um sistema de pontuação para cada categoria de atividades com o intuito de definir uma métrica global para a produção dos professores. Esta funcionalidade não foi implementada tendo em vista que seria necessário todo um sistema de criação de métricas isolado, que por si só já é um trabalho complexo para evitar distorções entre os eixos de atividades. Entretanto, é um ponto de grande interesse que poderá ser abordado em trabalhos futuros.

Bibliografia

ALCARAZ, P. C. F. Entity framework, uma introduÇÃo ao orm utilizando o code first. 2016. Disponível em: ⟨http://repositorio.utfpr.edu.br/jspui/bitstream/1/13464/2/MD_COADS_2016_2_05.pdf⟩.

BEZERRA, F. S. R.; VIANNA, W. Desenvolvimento nativo vs ionic vs react native: uma análise comparativa do suporte à acessibilidade em android. 2021. Disponível em: ⟨https://repositorio.ufc.br/bitstream/riufc/58979/1/2021_tcc_fsrbezerra.pdf⟩.

Brasil. *Lei de Diretrizes e Bases da Educação Nacional, LDB 9394*. 2021. Atualizada em abril de 2021. Disponível em: ⟨https://www2.senado.leg.br/bdsf/bitstream/handle/id/593336/LDB_5ed.pdf⟩.

BRASIL. Lei nº 14.129, de 29 de março de 2021. Diário Oficial da União, Seção 1, p. 3-7, 2021.

CERQUEIRA, D.; BITTENCOURT, R. A. Comparação e avaliação de frameworks mobile multiplataforma. 2014.

DRUCKER, P. The spirit of performance. In: . Open University Press, UK: Milton Keynes, 1989.

EXPO. *Documentação Oficial Expo*. Expo, s.d. Online; Acesso em Dez. 2021. Disponível em: ⟨https://docs.expo.dev/⟩.

FERREIRA, T. *Quem é Tim Berners-Lee? Conheça o "pai da internet"*. 2021. Online; Acesso em Jan. 2022. Disponível em: ⟨https://olhardigital.com.br/2021/08/24/internet-e-redes-sociais/quem-e-tim-berners-lee-conheca-o-pai-da-internet/⟩.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. Tese (Doutorado), 2000. Disponível em: ⟨https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm⟩.

GANDHEWAR, N.; SHEIKH, R. *Google Android: An Emerging Software Platform For Mobile Devices*. 2009.

IBC, E. Métodos e objetivos da coleta de dados em empresas – Por que e como são feitas? In: . [S.l.: s.n.], 2017. Disponível em: ⟨https://www.ibccoaching.com.br/portal/mercado-trabalho/metodos-objetivos-coleta-de-dados-empresas-por-que-como-sao-feitas/⟩.

ICE. *Resolução Nº02/2016 Normatiza a elaboração, acompanhamento e avaliação do Plano Individual de Trabalho e do Relatório Individual de Trabalho dos docentes do Instituto de Ciências Exatas da UFJF*. 2016.

ICE-UFJF. *Normatiza a elaboração, acompanhamento e avaliação do Plano Individual de Trabalho e do relatório Individual de trabalho dos docentes do Instituto de Ciência Exatas da UFJF*. 2016. Disponível em: ⟨https://www2.ufjf.br/ice/wp-content/uploads/sites/268/2017/01/Resolu%C3%A7%C3%A3o-02-2016-PIT-final.pdf⟩.

LOUDON, K. *Desenvolvimento de Grandes Aplicações Web*. 1a edição. ed. [S.l.]: Novatec, 2010. ISBN 978-85-7522-251-5.

MARQUES, A. I. A. Desenvolvimento de api para aplicação cloud. 2018. Disponível em: ⟨https://iconline.ipleiria.pt/bitstream/10400.8/3263/1/2151668_Ana%20Marques_MEICM_Tese.pdf⟩.

MATHEUS, G. R.; LOUREIRO, A. A. F. *Introdução à computação móvel*. 1998. Disponível em: ⟨https://homepages.dcc.ufmg.br/~loureiro/cm/docs/cm_livro_2e.pdf⟩.

Ministério das Comunicações. *Brasil tem mais de 234 milhões de acessos móveis em 2020*. 2021. Atualizada em abril de 2021. Disponível em: ⟨https://www.gov.br/pt-br/noticias/transito-e-transportes/2021/04/brasil-tem-mais-de-234-milhoes-de-acessos-moveis-em-2020⟩.

MONTANHEIRO, L. S.; CARVALHO, A. M. M.; RODRIGUES, J. A. Utilização de json web token na autenticação de usuários em apis rest. 2017. Disponível em: ⟨https://www.enacomp.com.br/2017/docs/json_web_token_api_rest.pdf⟩.

MOZILLA. *A web e seus padrões*. 2022. Online; Acesso em Fev. 2022. Disponível em: ⟨https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/The_web_and_web_standards⟩.

OpenJS Foundation. *A brief history of Node.js*. s.d. Online; Acesso em Dez. 2021. Disponível em: ⟨https://nodejs.dev/learn/a-brief-history-of-nodejs⟩.

REACTJS. *Documentação Oficial ReactJS*. ReactJS, 2022. Online; Acesso em Fev. 2022. Disponível em: ⟨https://reactjs.org/⟩.

REIFSCHNEIDER, M. B. Considerações sobre avaliação de desempenho. In: . Newport, Rhode Island: Scielo, 2010.

RUSCHEL, H.; ZANOTTO, S. M.; MOTA, W. C. Computação em nuvem. 2010. Disponível em: ⟨https://www.ppgia.pucpr.br/~jamhour/RSS/TCCRSS08B/Welton%20Costa%20da%20Mota%20-%20Artigo.pdf⟩.

SOUZA, E. C. de; OLIVEIRA, M. R. de. Comparativo entre os bancos de dados mysql e mongodb: quando o mongodb é indicado para o desenvolvimento de uma aplicação. 2019. Disponível em: ⟨https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/664/411⟩.

UFJF. *Resolução Nº46/95-CEPE Dispõe sobre Regime de Trabalho, Atividades Docentes e outras Providências*. 1995.

VENTEU, K. C.; PINTO, G. S. Desenvolvimento móvel híbrido. *Revista Interface Tecnológica*, 2018. Disponível em: ⟨https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/337⟩.