PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

JULIO CESAR ALVES DE ARAUJO

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA ACOMPANHAMENTO DA ATIVIDADE PARLAMENTAR DOS DEPUTADOS FEDERAIS

GOIÂNIA
2021

JULIO CESAR ALVES DE ARAUJO

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA ACOMPANHAMENTO DA ATIVIDADE PARLAMENTAR DOS DEPUTADOS FEDERAIS

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Max Gontijo de Oliveira

GOIÂNIA

2021

JULIO CESAR ALVES DE ARAUJO

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA ACOMPANHAMENTO DA ATIVIDADE PARLAMENTAR DOS DEPUTADOS FEDERAIS

Este Trabalho de Conclusão de Curso julgado adequado para obtenção o título de Bacharel em Ciência da Computação, e aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, em ____/____/_____.

_____

Prof. Me. Ludmilla Reis Pinheiro dos Santos

Coordenador(a) de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Orientador: Prof. Me. Max Gontijo

_____

Prof. Me. Fernando Gonçalves Abadia

_____

Prof. Me. Lucília Ribeiro

GOIÂNIA

2021

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema, composto por um aplicativo *mobile*, aplicação *backend*, banco de dados e sistema de troca de mensagens, com a finalidade de acompanhar a atividade parlamentar dos deputados federais do Brasil. Para construí-lo, foram utilizados os dados abertos da Câmara dos Deputados como fonte de informação sobre as atividades parlamentares. Esses dados foram carregados em um banco de dados PostgreSQL que é consumido por uma aplicação implementada em Java e Spring Framework, que por sua vez, expõem os dados para o aplicativo *mobile* escrito em Dart e Flutter por meio de uma HTTP API. A aplicação resultante permite acompanhar, na perspectiva de um deputado, o uso da cota parlamentar, votações, presença em eventos, discursos, cargos em órgãos e proposições. Além disso, é possível seguir deputados específicos para receber notificações quando novas atividades forem registradas.

Palavras-chave: deputado federal, dados governamentais abertos, integração de sistemas, aplicativo mobile

ABSTRACT

This work presents the development of a system, consisting of a mobile application, backend application, database and message exchange system, in order to monitor the parliamentary activity of federal deputies in Brazil. In order to develop it, open data from the Chamber of Deputies were used as a source of information on parliamentary activities. This data was loaded into a PostgreSQL database which is consumed by a Java and Spring Framework application. This app exposes the data to the mobile Dart/Flutter app via an HTTP API. The system makes it possible to monitor a deputy's activities, such as how the parliamentary quota is used, votes, presence at events, speeches, positions in government agencies and proposals. In addition, it is possible to follow specific deputies to receive notifications about their new activities.

Keywords: federal deputy, open government data, systems integration, mobile application

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

App	Aplicativo
API	Application Programming Interface
ETL	Extract Transform Load
FCM	Firebase Cloud Messaging
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
REST	Representational State Transfer
UI	User Interface
URI	Uniform Resource Identifier
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Durante a campanha eleitoral, os candidatos aos cargos eletivos se posicionam sobre diversos temas sociais e fazem promessas ao eleitorado na tentativa de convencê-lo a escolhê-los como os seus representantes no sistema democrático. Após eleitos, é importante que a população os acompanhe e fiscalize para se certificar que eles estão cumprindo as suas promessas e agindo de acordo com o interesse público. Para que essa fiscalização possa ser feita, é preciso que haja informação por parte das instituições públicas sobre as atividades dos representantes do povo.

Nos últimos anos, com o movimento de dados abertos no governo e as leis de transparência e acesso à informação, os órgãos do Estado passaram a compartilhar as informações que são de interesse público com a sociedade. A exemplo disso, a Câmara dos Deputados possui um sítio de notícias e informações onde divulga as atividades legislativas desempenhadas pelos deputados federais. E ainda, publica essas informações em formato aberto e compreensível por máquina, possibilitando a criação de serviços e aplicações de utilidade pública.

Diante de tal possibilidade e considerando a importância das funções exercidas pelo deputado federal – ele legisla e delibera sobre temas que podem contribuir para a melhoria do país – este trabalho tem como objetivo explorar os dados abertos da Câmara dos Deputados para desenvolver uma aplicação que permita o acompanhamento das atividades parlamentares dos deputados federais.

Para tanto, foram definidos os seguintes objetivos específicos: extrair, processar e carregar os dados abertos da Câmara dos Deputados em um banco de dados relacional; desenvolver uma HTTP API para expor os dados do banco de dados; criar um aplicativo *mobile* para consumir e exibir os dados da HTTP API.

1.1 Metodologia

A metodologia utilizada para o desenvolvimento deste trabalho foi a seguinte:

- Analisar as informações disponíveis nos dados abertos da Câmara dos Deputados para definir as funcionalidades do sistema.
- Procurar por tecnologias que atenda as funcionalidades definidas.

- Arquitetar e definir a comunicação entre os sistemas que integram a solução final.
- Implementar a solução arquitetada.
- Testar e validar as funcionalidades do sistema.

1.2 Organização dos capítulos

O capítulo 2 descreve alguns trabalhos que têm objetivos similares à esta monografia.

O capítulo 3 apresenta os conceitos considerados importantes para o entendimento deste trabalho e as tecnologias utilizadas na implementação da aplicação.

O capítulo 4 descreve a construção da aplicação e apresenta os resultados obtidos.

O capítulo 5 apresenta as conclusões e propostas de trabalhos futuros.

2 TRABALHOS RELACIONADOS

Alguns softwares já foram desenvolvidos com o propósito de acompanhar a atuação parlamentar dos deputados. Neste capítulo são apresentados alguns aplicativos, identificados em pesquisa no Google Scholar e na Play Store, que têm propósitos similares a solução apresentada neste trabalho.

2.1 Infoleg

Infoleg (CÂMARA DOS DEPUTADOS, 2021) é um aplicativo que disponibiliza informações sobre as atividades legislativas da Câmara dos Deputados. O aplicativo exibe informações sobre deputados, como por exemplo, canais para contato, titularidade em comissões e agenda legislativa; lideranças partidárias; bancadas; sessões e reuniões do plenário; comissões; proposições; normas legislativas. O aplicativo também permite configurar o recebimento de notificações sobre as atividades das sessões do plenário e comissões.

2.2 Monitora, Brasil!

Monitora, Brasil! (GAMFIG CORP, 2021) é um aplicativo que permite acompanhar as atividades parlamentares dos deputados federais e senadores. Não foi possível testar as funcionalidades do aplicativo pois ele não carrega nenhuma informação ao acessá-lo. Comentários recentes no Google Play Store também relatam o mesmo problema. De acordo com a descrição no Google Play Store, o aplicativo apresenta informações sobre assiduidade, projetos propostos, Twitter e uso da cota parlamentar. O aplicativo também permite acompanhar o andamento de projetos por meio de notificações.

2.3 Meu Deputado

Meu Deputado (MENEGUZ, 2021) é um aplicativo que disponibiliza informações sobre uso da a cota parlamentar, presenças em sessões e votações dos deputados. Além disso, possui um *ranking* de deputados baseado no tipo de gasto

com a cota parlamentar. Este aplicativo parece estar com a base de dados desatualizada pois não foi possível visualizar as informações de vários deputados em exercício na legislatura atual.

2.4 Câmara Popular

Câmara Popular (TELES, 2021) é um aplicativo que permite ao usuário enviar ideias legislativas e votar em projetos de lei através do portal e-Cidadania do Senado Federal. Além disso, o *app* apresenta informações sobre os deputados federais e proposições. Sobre os deputados, é possível visualizar as presenças em eventos, participações como membro em órgãos e uso da a cota parlamentar.

2.5 Laddres

Laddres é uma aplicação desenvolvida no trabalho “Transformando dados em conhecimento: LADDRES, uma aplicação prática”, (PASSOS, 2018). Nele, o autor extrai dados de várias fontes para compor um banco de dados com informações sobre mandatos, processos judiciais, atuação parlamentar e posicionamento sobre temas sociais dos candidatos para os cargos eletivos de 2018. Também foi criado uma Web API para expor os dados processados e um aplicativo *mobile* que permite consultar as informações sobre os candidatos. Nesse trabalho, os dados abertos da Câmara dos Deputados foram utilizados para extrair informações sobre deputados, mandatos e proposições.

2.6 Comparativo entre os aplicativos

Nesta subseção é realizado um comparativo entre os aplicativos que têm como um de seus propósitos o acompanhamento da atividade parlamentar dos deputados federais. O Laddres não foi incluso na comparação pois o seu principal objetivo é fornecer informações sobre os candidatos aos cargos eletivos de 2018. A comparação foi realizada considerando as informações relacionadas a atuação parlamentar disponibilizada pelos aplicativos, conforme o Quadro 1. O presente trabalho é referenciado como “Monitor Legislativo”.

Quadro 1 - Comparativo entre os aplicativos

Informações	Infoleg	Monitora, Brasil!	Meu Deputado	Câmara Popular	Monitor Legislativo
Uso da cota parlamentar		X	X	X	X
Proposições	X	X		X	X
Participações em órgãos da Câmara	X			X	X
Presença em eventos da Câmara	X	X	X	X	X
Votos em proposições			X		X
Liderança partidária	X				
Discursos na Câmara					X

Fonte: próprio autor

Note que apenas o Monitor Legislativo disponibiliza informações sobre os discursos na Câmara. Um aspecto não comparado na tabela 1 é a existência de um sistema de notificação. Apenas o Infoleg, Monitora, Brasil! e o Monitor Legislativo dispõem de um. No caso do Monitor Legislativo, o sistema de notificação alerta sobre novas atividades de um deputado seguido pelo usuário. Já no Infoleg, as notificações são sobre as atividades das sessões do Plenário e comissões da Câmara dos Deputados. E no Monitora, Brasil!, as notificações são sobre o andamento dos projetos.

Outro recurso presente no Monitor Legislativo, mas não nos demais aplicativos, é a possibilidade de visualizar as proposições na forma de gráficos. Além disso, apenas o Monitor Legislativo e o Infoleg permitem assistir, dentro do aplicativo, os eventos que o deputado participou.

3 REFERENCIAL TEÓRICO

Neste capítulo são apresentados alguns conceitos necessários para o entendimento deste trabalho.

3.1 Dados Governamentais Abertos

O governo produz dados dos mais variados para executar as suas atividades na administração pública. Por exemplo: dados sobre a saúde, educação, finanças, habitação, contratos e licitações, previdência social e entre outros. Essas informações, quando disponibilizadas à sociedade de tal forma que qualquer pessoa consiga consultar diretamente e reutilizar, para quaisquer fins, como por exemplo, na construção de novos sistemas ou *sites*, são consideradas dados governamentais abertos (MANUAL, 2011).

Uma definição mais precisa de dados governamentais abertos foi dada em 2007, na Califórnia, por um grupo composto de 30 defensores do governo aberto. Eles definiram um conjunto de princípios que, quando adotados na publicação dos dados governamentais, tornam os governos mais transparentes, eficazes e relevantes para a sociedade (OPEN GOVERNMENT DATA, 2021). Segundo esses princípios, os dados precisam ser:

- **Completos**: todos os dados públicos são disponibilizados. Dados públicos são aqueles que não estão sujeitos a limitações válidas de privacidade, segurança ou controle de acesso.
- **Primários**: os dados são coletados da fonte com alto nível de granularidade e sem modificações ou agregações.
- **Atuais**: os dados são publicados tão rapidamente quanto necessário para que não percam o seu valor.
- **Acessíveis**: os dados são disponibilizados para alcançar o maior número de usuários possíveis e para os mais diversos propósitos.
- **Processáveis por máquina**. os dados são razoavelmente estruturados de tal forma que o processamento automatizado por máquina seja possível.

- **Não discriminatório**: os dados são acessíveis diretamente por qualquer pessoa e sem a necessidade de cadastros.
- **Não proprietários**: os dados estão disponíveis em um formato sobre o qual nenhuma entidade tenha controle exclusivo.
- **Livres de licença**: Os dados não estão sujeitos a regulamentação de direitos autorais, patentes, marcas ou segredo industrial. No entanto, admite-se restrições razoáveis de privacidade, segurança e controle de acesso.

No Brasil, o movimento de dados abertos no governo foi impulsionado por dois eventos, segundo (RIBEIRO e ALMEIDA, 2011). O primeiro, consiste na publicação do decreto nº 6.932 de 11 de agosto de 2009 que, dentre outras coisas, determina que o Poder Executivo nas suas relações com o cidadão compartilhe informações e crie mecanismos que facilitem o seu acesso. E segundo, a publicação do documento *Memorandum on Transparency and Open Government*, em 2009, pelo então presidente Barack Obama, que define algumas diretrizes para a gestão, acesso e publicação dos dados do governo dos Estados Unidos da América. Algumas iniciativas desse movimento podem ser encontradas no Portal Brasileiro De Dados Abertos (BRASIL, 2021).

Vale ressaltar que o termo “dados governamentais abertos” também é comumente utilizado para se remeter a quaisquer informações disponibilizadas pelas instituições públicas dos três poderes, não se reservando apenas aos dados do poder Executivo (CÂMARA DOS DEPUTADOS, 2021).

3.2 Principais atribuições do deputado federal

No Brasil, a cada quatros anos, os eleitores vão as urnas para eleger os seus representantes na Câmara dos Deputados. No total, 513 deputados federais são eleitos por legislatura sendo que, cada unidade federativa elege de 8 a 70 deputados federais, dependendo do tamanho da população da região. A Câmara dos Deputados, juntamente com o Senado Federal, compõe o Congresso Nacional e formam o Poder Legislativo da União (BRASIL, 1988).

O Poder Legislativo é responsável pela criação, alteração e revogação das leis. O deputado federal, por fazer parte desse poder, tem a mesma função e só pode fazê-

la sobre temas que são de competência da União, respeitando as competências exclusivas do poder Executivo e Judiciário. Tais temas estão dispostos no artigo 22 da Constituição Federal e são exemplos: direito civil, comercial, penal, processual, eleitoral, agrário, marítimo, aeronáutico, espacial e do trabalho (SOARES, 2011).

O processo legislativo se dá pela elaboração de emendas à Constituição, leis complementares, leis ordinárias, leis delegadas, medidas provisórias, decretos legislativos e resoluções. Dessas, apenas as leis delegadas e medidas provisórias não podem ser de iniciativa dos deputados federais (SOARES, 2011). Quando uma matéria legislativa é apresentada por um deputado federal, ela precisa ser discutida e votada em comissões e no Plenário da Câmara, dependendo da sua tramitação, antes de seguir para o Senado Federal ou sanção presidencial (CÂMARA DOS DEPUTADOS, 2021).

Outra competência do deputado federal é a fiscalização do Poder Executivo da União e ele pode fazer isso, de acordo com a (CÂMARA DOS DEPUTADOS, 2021), através de convocação de ministro de estado para prestar informações presencialmente; de requerimento de informação, por escrito, a ministros de estado; da apresentação de Proposta de Fiscalização e Controle para apurar irregularidades no âmbito da administração pública; da atuação na Comissão de Fiscalização Financeira e Controle.

Diariamente, o deputado federal participa de deliberações e discursões no Plenário e em comissões da Câmara. As comissões são órgãos constituídos por deputados e podem ser permanentes ou temporárias. As comissões permanentes são temáticas e tratam de proposições relacionadas a seus temas, como por exemplo, educação, segurança pública e agricultura. Já as comissões temporárias têm prazo determinado de funcionamento e podem, por exemplo, serem instauradas para investigar assuntos de interesse público e de relevância nacional, como é o caso da Comissão Parlamentar de Inquérito (CPI) (CÂMARA DOS DEPUTADOS, 2021).

Nas comissões, o deputado federal pode desempenhar a função de relator. Neste caso, ele deve apresentar um parecer sobre a matéria recebida pela comissão, defendendo a sua aprovação ou rejeição. Posteriormente, o parecer é discutido e votado pelos demais deputados integrantes da comissão (CÂMARA DOS DEPUTADOS, 2021).

E por fim, segundo (BRASIL, 1988), compete exclusivamente aos deputados federais:

- autorizar a instauração de processos contra o Presidente e o Vice-Presidente da República e os Ministros de Estado;
- realizar a tomada de contas do Presidente da República;
- elaborar o regimento interno da Câmara dos Deputados e definir normas sobre o seu funcionamento e organização;
- eleger membros do Conselho da República que, por exemplo, podem deliberar sobre Intervenção Federal, Estado de Defesa e Estado de Sítio.

Neste capítulo foram apresentadas algumas atribuições do deputado federal. De forma resumida, ele é responsável por legislar e fiscalizar o poder Executivo.

3.3 Integração de sistemas

A integração de sistemas é uma tarefa comum no desenvolvimento de software. Normalmente, uma aplicação precisa consumir dados e funcionalidades de diversas outras para entregar uma experiência ao usuário final. Por exemplo, considere uma pessoa realizando uma compra em um e-commerce. Após escolher os produtos e finalizar a compra, algum sistema irá processar o pedido e, muito provavelmente, se comunicará com outros sistemas para confirmar o pagamento, emitir nota fiscal e enviar os produtos. Esses sistemas podem estar escritos em tecnologias e plataformas diferentes, e as vezes, nem foram projetados para se comunicar ou são difíceis de serem alterados para isso, como é caso de sistemas legados.

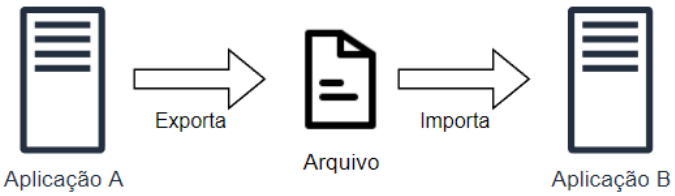
Devido a essas características, existem várias técnicas que podem ser empregadas na integração de sistemas e neste capítulo são apresentadas as utilizadas na construção deste trabalho.

3.3.1 File Transfer

De acordo com (HOHPE e BOBBY, 2004), uma forma de possibilitar a troca de dados entre sistemas é utilizando um arquivo como interface entre eles. Isso é exemplificado na Figura 1. Nela, a aplicação A periodicamente gera um arquivo em

um formato processável por máquina, por exemplo em *Extensible Markup Language* (XML) ou *JavaScript Object Notation* (JSON), e em um local acessível pela aplicação B, que por sua vez, lê o arquivo e faz o processamento de acordo com as suas necessidades.

Figura 1 - Integração utilizando arquivo



Fonte: próprio autor, baseando-se em (HOHPE e BOBBY, 2004)

Essa estratégia permite que duas aplicações troquem apenas dados, não sendo possível o compartilhamento de funcionalidades. E ainda, ao utilizá-la, deve-se levar em conta que podem ocorrer problemas de obsolescência de dados devido a periodicidade em que eles são exportados e importados pelas aplicações (HOHPE e BOBBY, 2004).

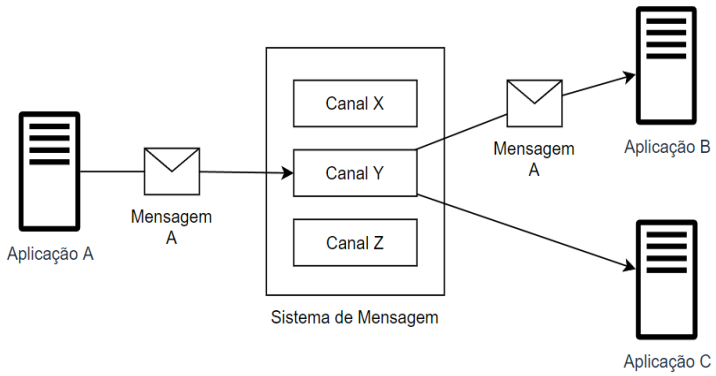
3.3.2 Messaging

Segundo (HOHPE e BOBBY, 2004), na integração por *Messaging,* ou mensageria*,* as aplicações se comunicam por meio de um sistema de mensagem. Para que a comunicação aconteça, as aplicações precisam estar conectadas em um canal, conforme as Figuras 2 e 3, e a troca de dados ou funcionalidades é feita por meio do envio e recebimento de mensagens assíncronas.

O canal é uma estrutura lógica que recebe e armazena as mensagens. O sistema de mensagem pode ter vários canais, criados pelos desenvolvedores, de acordo com as necessidades de integração das aplicações. Através do canal uma mensagem pode ser entregue para um único aplicativo, mesmo existindo vários conectados a ele, ou para vários. Em algumas implementações de sistema de mensagens, como por exemplo, o ActiveMQ, o canal que entrega cada mensagem

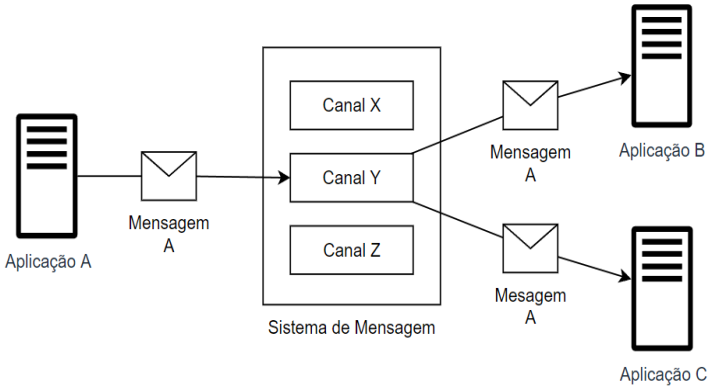
para um único aplicativo é chamado de *queue* (fila), e o que entrega para muitos, de *topic* (tópico).

Figura 2 - Comunicação um-para-um



Fonte: próprio autor, baseando-se em (HOHPE e BOBBY, 2004)

Figura 3 - Comunicação um-para-muitos



Fonte: próprio autor, baseando-se em (HOHPE e BOBBY, 2004)

Um aspecto relevante nesse tipo de integração é que a aplicação que deseja consumir uma mensagem em um canal não precisa estar disponível, ou seja, em execução ou pronta para processar a mensagem quando ela é enviada. Isso possibilitou, no desenvolvimento deste trabalho, que uma aplicação *back-end* entregasse dados para um dispositivo *mobile*, mesmo sem saber se ele estava conectado na *internet* ou ligado.

3.3.3 RESTful Web Service

Um *RESTful Web service* é um *software* que fornece uma interface de integração por meio da rede e que segue as restrições definidas pelo estilo arquitetural *Representational State Transfer* (REST).

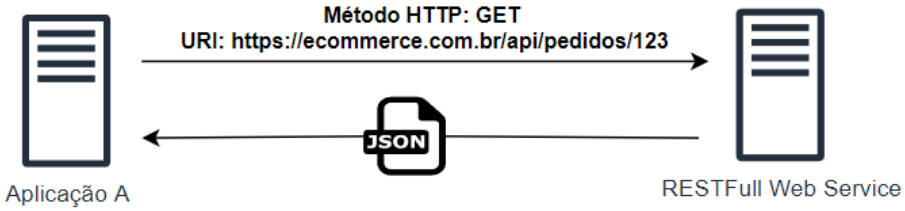
REST foi proposto por Roy Thomas Fielding no ano 2000 em sua tese de doutorado intitulada por *"Architectural Styles and the Design of Network-based Software Architectures"*. Nela, mais especificamente no capítulo 5, Fielding descreve um conjunto de restrições que define o estilo arquitetural. Essas restrições, de acordo com (FIELDING, 2000), são as seguintes:

- ***Client-Server***: ter uma arquitetura cliente-servidor. Nessa arquitetura um servidor expõe um conjunto de serviços que podem ser requisitados pelo cliente. O servidor, ao receber uma requisição do cliente, pode rejeitar ou processar a requisição e devolver uma resposta.
- ***Stateless*****:** a comunicação entre o cliente e o servidor não deve conter estado. Isso implica que uma requisição feita pelo cliente deve conter todas as informações necessárias para que o servidor consiga fazer o processamento.
- ***Cache*****:** a resposta do servidor deve incluir indicações de que os dados podem ou não ser armazenados pelo cliente para reuso posterior em requisições equivalentes.
- ***Uniform Interface*****:** ter uma *interface* uniforme entre o cliente e servidor para que a comunicação seja feita de forma padronizada. Para isso, os recursos providos pelo servidor precisam ter um identificador; os recursos podem ser manipulados por meio das representações; as mensagens devem ser autodescritivas; e ter a hipermídia como o motor de estado do aplicativo.
- ***Layered System*****:** ter um sistema organizado em camadas hierárquicas, onde uma camada inferior fornecer serviços para a camada superior que, por sua vez, só consegue acessar os serviços da camada inferior adjacente.
- ***Code-On-Demand*****:** permitir a extensão das funcionalidades do cliente por meio de código executável enviado pelo servidor. A implementação dessa restrição é opcional.

Fielding, em sua tese, não define tecnologias, sintaxe de protocolo e detalhes de implementação para construção de sistemas com arquitetura REST. No entanto, de acordo com (SILVEIRA, 2013), “REST pode ser alcançado através de, por exemplo, HTTP e representações com suporte a hipermídia, mas poderia ser utilizada outra pilha de tecnologias com as mesmas características.”.

Um exemplo de comunicação com um *RESTful Web Service* utilizando o protocolo HTTP é exemplificado na figura 4. Nela, a aplicação A requisita ao *Web Service* uma ação sobre o recurso identificado pela *Uniform Resource Identifier* (URI) https://ecommerce.com.br/api/pedidos/123. O uso do método HTTP GET indica que a aplicação A deseja consultar a representação do recurso. Além disso, pela estrutura da URI é possível deduzir que o recurso solicitado é um pedido. Ainda na figura 4, o *Web Service* responde a requisição enviando a representação do pedido no formato *JavaScript Object Notation* (JSON). O conteúdo da resposta do *Web Service* poderia ser algo como mostrado na figura 5. A representação retornada contém um *link* que permite que a aplicação A acesse os produtos relacionados ao pedido. A inclusão de *links* para outros recursos nas representações implementa o suporte a hipermídia (RICHARDSON e RUBY, 2007).

Figura 4 - Comunicação com um RESTful Web Service



Fonte: próprio autor

Figura 5 - Representação em JSON

Fonte: próprio autor

No restante do texto o termo REST API é utilizado como sinônimo de *RESTful Web Service.* Também é utilizado o termo *endpoint* para referenciar um ponto de acesso de recurso na REST API. Por exemplo: na URI https://ecommerce.com.br/api/pedidos/123, o caminho /api/pedidos/123 é um *endpoint* do serviço ecommerce.com.br. E por fim, HTTP API é utilizado para referenciar um *Web Service* que permite que outros sistemas se comuniquem com ele por meio do protocolo HTTP, mas não segue todas as restrições do estilo arquitetural REST.

3.4 Tecnologias utilizadas

Neste capítulo são descritas as linguagens de programação, *frameworks,* bibliotecas, banco de dados e serviços que foram utilizados na construção deste trabalho.

3.4.1 Java

Java é uma linguagem de programação de uso geral, concorrente, orientada a objetos e fortemente tipada (ORACLE, 2021). O *design* da linguagem foi influenciado por, entre outras linguagens, C e C++. No entanto, diferente dessas duas, Java conta com gerenciamento automático de memória por meio de *Garbage Collector* (GC).

A compilação da linguagem é feita, normalmente, para um código intermediário chamado de *bytecode*, que é interpretado por uma *Java Virtual Machine* (JVM). Isso permite que as aplicações escritas em Java sejam portáveis entre plataformas.

3.4.2 Spring

Spring (SPRING, 2021) é uma plataforma composta por um conjunto de soluções para desenvolvimento de aplicações Java. Ela tem como foco tornar a programação em Java mais simples e produtiva. Das soluções disponíveis, as utilizadas neste trabalho foram:

- **Spring Boot:** reduz a quantidade de configurações necessárias para o desenvolvimento e execução de aplicações em produção.
- **Spring Framework:** provê um modelo de programação e configuração para aplicativos corporativos construídos em Java (SPRING, 2021). Conta com recursos, como, injeção de dependência, eventos, *data binding, Aspect-Oriented Programming* (AOP), transação, suporte a *Data Access Object (DAO)* e *Model View Controller (MVC).*
- **Spring Data:** facilita o uso de tecnologia de acesso a dados, banco de dados relacionais e não relacionais, *frameworks* de *map-reduce e* serviços de dados baseados em *cloud* (SPRING, 2021).

3.4.3 Dart

Dart é uma linguagem de programação desenvolvida pela Google cujo objetivo é ser a mais produtiva para o desenvolvimento de aplicativos multiplataforma (DART, 2021). Dart é fortemente tipada e oferece proteção contra referência nula de variáveis. O código escrito em Dart pode ser compilado em código de máquina, no caso de aplicativos direcionados a dispositivos moveis e *desktop*, ou em Java Script, para aplicativos voltados a web.

3.4.4 Flutter

Flutter é um conjunto de ferramentas desenvolvidas pela Google para construção de *User Interface* (UI) multiplataforma a partir de uma única base de código (FLUTTER, 2021). Flutter utiliza Dart como linguagem base e pode ser compilado nativamente para dispositivos móveis, web, desktop e embarcados.

A construção de interfaces gráficas no Flutter é feita por meio da composição de vários w*idgets.* O *widget* é uma classe que representa elementos gráficos ou de layout, como por exemplo, textos, botões, imagens, linhas e colunas. Os *widgets* também são construindo a partir da composição de outros.

O Flutter disponibiliza uma coleção de widgets prontos para uso e permite a criação de novos. Isso normalmente é feito através da criação de subclasses de StatelessWidget ou StatefulWidget, dependendo se há a necessidade de gerenciar algum estado.

3.4.5 Firebase Cloud Messaging

O Firebase Cloud Messaging (FCM) é um serviço que permite a troca de mensagens entre plataformas de forma confiável e gratuita (FIREBASE, 2021). Normalmente, o FCM é utilizado para enviar mensagens a partir de uma aplicação *back-end* para um aplicativo *mobile* ou Web.

Existem dois tipos de mensagem no FCM, sendo elas:

- **Mensagem de notificação**: essas mensagens são processadas pelo *Software Development Kit* (SDK) do FCM na aplicação cliente e exibidas automaticamente para o usuário.
- **Mensagem de dados**: são mensagens tratadas pelo código da aplicação cliente.

As mensagens podem ser distribuídas ao aplicativo cliente de três formas: para um único dispositivo, para um grupo de dispositivos ou para dispositivos inscritos em um tópico.

3.4.6 PostgreSQL

PostgreSQL (POSTGRESQL, 2021) é um Sistema de Gerenciamento de Banco de Dados Relacional (DBMS) gratuito e de código aberto que utiliza a linguagem *Structured Query Language* (SQL) para manipular e consultar os dados.

De acordo com (HEUSER, 1998), um banco de dados relacional organiza os dados em uma coleção de tabelas. Essas são compostas por um conjunto de linhas que, por sua vez, são constituídas por uma coleção de colunas que podem ser de vários tipos, como por exemplo, um texto ou um valor numérico. Uma linha pode conter uma chave primária que é representada por uma coluna ou um conjunto de colunas que a identifica exclusivamente dentro da tabela. E ainda, uma linha pode conter uma coluna ou um conjunto de colunas, chamada de chave estrangeira, que referência uma chave primária de uma linha. A chave estrangeira é utilizada para implementar o conceito de relacionamento no banco de dados relacional.

3.4.7 Quartz

Quartz é uma biblioteca de código de aberto para agendamento de trabalhos que pode ser integrada em qualquer aplicação Java (QUARTZ, 2021). Com ela é possível programar a execução de milhares de tarefas com base no tempo. As tarefas são componentes escritos em Java que declaram o que deve ser executado. Um exemplo de caso de uso do Quartz pode ser o seguinte: diariamente, às 09:00hs, enviar um e-mail aos clientes de um *e-commerce* sobre as promoções disponíveis.

4 DESENVOLVIMENTO DO PROJETO

Neste capítulo são descritas as etapas realizadas para construir a aplicação proposta nesta monografia.

4.1 Descrição das funcionalidades

A Câmara dos Deputados possui um portal de dados abertos que disponibiliza informações acerca das suas atividades parlamentares. As informações disponíveis são sobre:

- Legislaturas: período de 4 anos que compreende o mandato parlamentar.
- Deputados: representante do povo cuja principais funções são legislar e fiscalizar.
- Cota parlamentar: cota mensal utilizada para custear os gastos do deputado com despesas relacionadas ao exercício da atividade parlamentar. Isso inclui, entre outras, despesas com passagens aéreas, telefonia, serviços postais e manutenção de escritórios de apoio à atividade parlamentar. O valor máximo da cota vária de acordo a unidade federativa que o deputado representa. Em Goiás, por exemplo, o valor atual é de R$35.507,06 mensalmente (CÂMARA DOS DEPUTADOS, 2021).
- Discursos: pronunciamentos realizados pelos deputados em eventos da Câmara.
- Partidos Políticos: partidos oficialmente registrados no Tribunal Superior Eleitoral Brasileiro.
- Blocos Parlamentares: composição de vários partidos políticos que atuam como um só grupo e sob a mesma liderança (CÂMARA DOS DEPUTADOS, 2021).
- Órgãos Parlamentares: instituições onde os deputados realizam suas atividades parlamentares, como por exemplo, comissões, Mesa Diretora, procuradorias, conselhos e o Plenário.

- Frentes Parlamentares: junção de deputados e senadores de diferentes partidos cujo objetivo é discutir um assunto específico (CÂMARA DOS DEPUTADOS, 2021).
- Eventos: eventos organizados pela Câmara e seus órgãos, como por exemplo, seminários, reuniões e sessões deliberativas, audiências públicas e palestras.
- Proposições: qualquer matéria submetida à apreciação do Senado, da Câmara ou do Congresso Nacional, como por exemplo, PEC (Proposta de Emenda à Constituição); projetos de lei ordinária, de lei complementar, de decreto legislativo e de resolução; requerimentos; pareceres; indicações e emendas (SENADO FEDERAL, 2021).
- Votações: resultado de uma decisão tomada por um grupo de deputados sobre uma proposição específica.

Considerando essas informações, chegou-se nas seguintes funcionalidades que a aplicação deve ter:

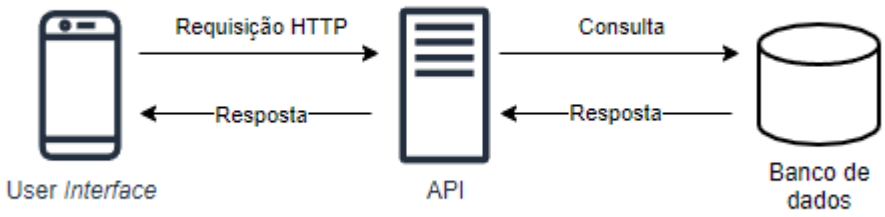
- **Visualizar deputados**: o usuário deve conseguir visualizar os deputados da legislatura atual.
- **Visualizar proposições**: o usuário deve conseguir visualizar as proposições que são de autoria de um determinado deputado.
- **Visualizar discursos**: o usuário deve conseguir visualizar os discursos realizados por um determinado deputado.
- **Visualizar participação em eventos**: o usuário deve conseguir visualizar os eventos que um determinado deputado participou.
- **Visualizar uso da cota parlamentar**: o usuário deve conseguir visualizar o uso da cota parlamentar de um determinado deputado.
- **Visualizar cargos em órgãos**: o usuário deve conseguir visualizar a participação como membro em órgãos da câmara de um determinado deputado.
- **Visualizar votações**: o usuário deve conseguir visualizar os votos de um determinado deputado.

- **Receber notificações sobre novas atividades dos deputados:** o usuário deve conseguir receber notificações sobre novas atividades de um determinado deputado.

4.2 Análise das funcionalidades

Todas as funcionalidades descritas na seção 4.1 podem ser entendidas como operações de consulta de dados. Em termos de implementação, consiste em uma *User Interface* (UI), neste caso, um aplicativo *mobile*, fazendo requisições de busca de dados para uma API, conforme a Figura 6. A Câmara dos Deputados disponibiliza uma REST API que pode ser utilizada para esse propósito. Dessa forma, por exemplo, a funcionalidade "Visualizar discursos" poderia ser implementada utilizando o *endpoint* /deputados/{id}/discursos dessa API, que retorna os discursos de um deputado identificado por {id}. No entanto, essa REST API tem algumas limitações. Por exemplo: não é possível filtrar os discursos com base no conteúdo da transcrição. Ou ainda, caso quisesse consultar o valor total do uso da cota parlamentar de um deputado, também não seria possível pois a API não retorna os dados de forma agregada.

Figura 6 - Comunicação entre UI e API



Fonte: próprio autor

A funcionalidade "Receber notificações sobre novas atividades dos deputados" exige a capacidade de detectar quando novos registros são inseridos nos dados abertos. A Câmara dos Deputados não dispõe de API's passivas (*webhooks*) que emitem eventos sobre atualização de dados. Dessa forma, é necessário implementar

algum mecanismo que consiga identificar a atualização de informações nos dados da Câmara.

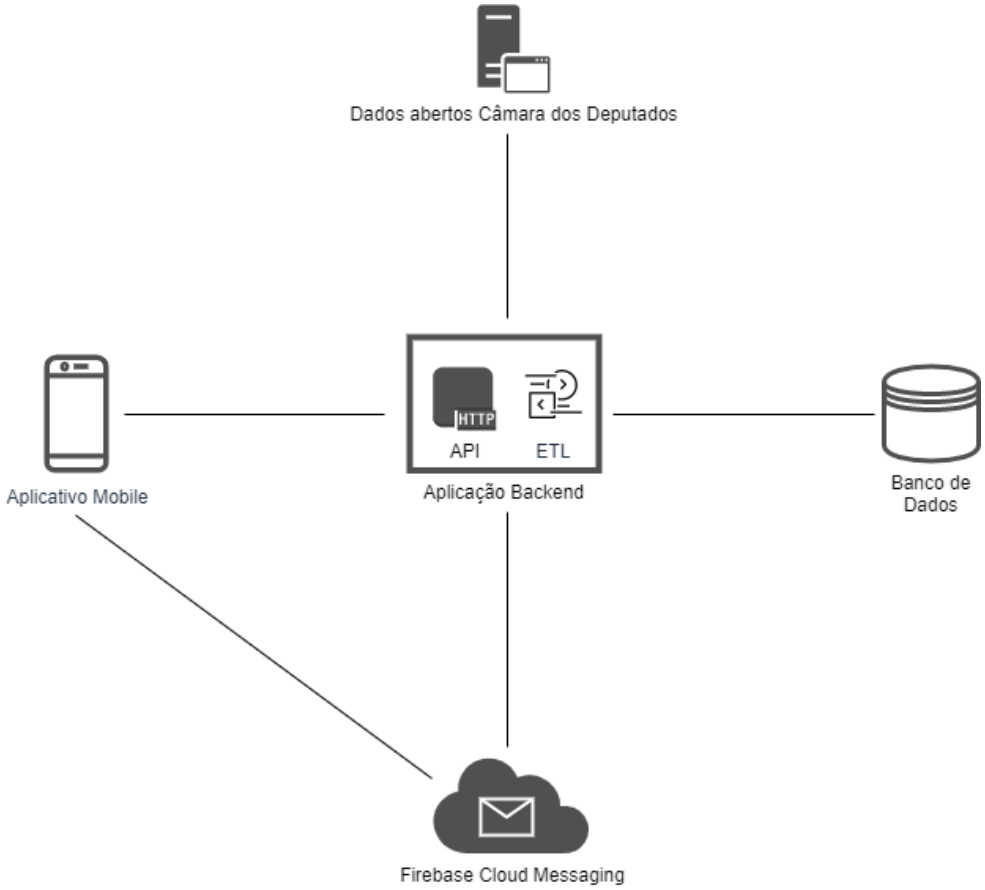
4.3 Arquitetura da aplicação

Para resolver as limitações da REST API da Câmara, optou-se em replicar os dados abertos para um banco de dados relacional. Isso permitirá a agregação de dados, criação de filtros personalizados e identificação de novos dados. Em contrapartida, aumentará a complexidade do sistema pois será necessário garantir a consistência dos dados através de sincronizações periódicas.

A arquitetura da aplicação, em nível de comunicação entre os sistemas, é descrita na Figura 7. Os sistemas presentes são:

- Aplicação Backend: é a aplicação responsável por fazer a Extração, Transformação e Carga (ETL) dos dados da Câmara. Ela também expõe o banco de dados através de uma HTTP API e envia mensagens para o Firebase Messaging Cloud quando novos dados são identificados durante o processo de ETL. A aplicação foi construída utilizando Java e Spring.
- Aplicativo Mobile: é a *interface* de comunicação entre o usuário e o sistema. Ele consome os dados expostos pela aplicação *backend* e recebe mensagens do Firebase Messaging Cloud. O aplicativo foi desenvolvido utilizando Dart e Flutter.
- Banco de dados: é um banco de dados PostgreSQL para armazenamento dos dados processados pela ETL.
- Firebase Cloud Messaging: permite a comunicação orientada a mensagem entre a aplicação *backend* e o aplicativo mobile.

Figura 7 - Comunicação entre os sistemas



Fonte: próprio autor

4.4 Implementação

Nesta seção é apresentado o processo de construção e integração dos sistemas descritos na seção 4.3.

4.4.1 Extração, Transformação e Carga dos dados

Os dados abertos da Câmara dos Deputados podem ser acessados através de REST API ou arquivos. Por meio de REST API, os dados são disponibilizados nos formatos de JSON e XML e têm limites de retorno de 100 itens por requisição HTTP. Além disso, a REST API tem *endpoints* específicos para cada tipo de informação que se deseja acessar. Já por meio de arquivos, os dados estão disponíveis nos formatos de CSV, ODS, XLSX, JSON e XML e estão divididos de acordo com o tipo de informação. Em alguns casos, os dados são subdivididos por ano ou legislatura, como

por exemplo, as proposições e deputados membros dos órgãos, respectivamente, e há casos, como os órgãos parlamentares, em que estão agrupados em um só arquivo.

Os dados da Câmara foram extraídos e carregados em um banco dados relacional. Por ser mais rápido, a extração desses dados foi realizada, em grande parte, por de meio de arquivos. Os quadros 2 e 3 descrevem os arquivos e *endpoints* da REST API utilizados na extração dos dados. Ambos podem ser consultados em https://dadosabertos.camara.leg.br/swagger/api.

Quadro 2 - Arquivos utilizados para extração dos dados

Arquivo	Dados extraídos
Despesas pela Cota Parlamentar	valor líquido, fornecedor, data e tipo da despesa
Proposições	identificador, tipo, número, data da apresentação, situação, ementa e URL do documento PDF que detalha a proposição
Tema das proposições	classificação temática da proposição
Autores das Proposições	deputados autores da proposição
Órgãos	identificador, descrição, sigla, nome e tipo do órgão
Deputados membros dos órgãos	deputados que participam como membro em órgãos da Câmara
Deputados	identificador, nome, partido, unidade federativa, foto e situação do deputado
Eventos	identificador, data, situação, descrição e tipo do evento
Deputados presentes em cada evento	deputados que registraram presença um determinado evento
Votações	identificador, data, descrição, resultado e órgão em que ocorreu a votação
Voto de cada parlamentar	voto (sim ou não) do deputado em uma determinada votação

Fonte: próprio autor

Quadro 3 - Endpoints da REST API utilizados para extração de dados

Endpoint	Dados extraídos
/deputados/{id}	foto do deputado
/deputados/{id}/discursos	data, sumario e transcrição do discurso
/eventos/{id}	*Uniform Resource Locator (*URL) para acesso a gravação do evento

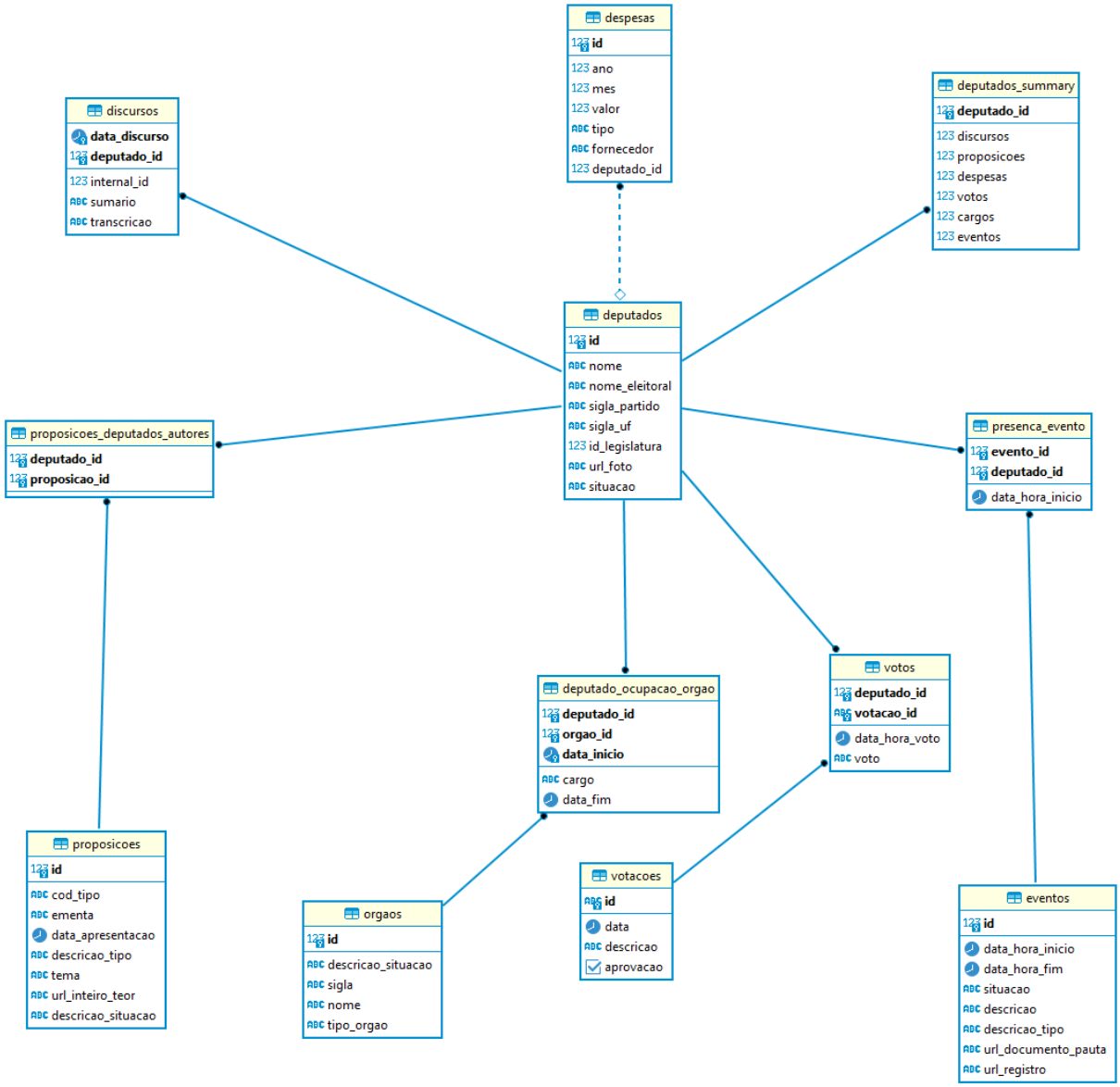
Fonte: próprio autor

O processo de ETL dos dados foi realizado de forma programática através dos componentes escritos em Java. A aplicação processa apenas os registros com data

de ocorrência na legislatura corrente. Durante a construção deste trabalho, os arquivos processados foram os seguintes: quando divididos por ano, os datados em 2019, 2020 e 2021; quando divididos por legislatura, apenas os identificados por “L56”; e os não divididos. Conforme a tabela 3, também foram utilizados alguns *endpoints* da REST API para extrair dados. Isso foi necessário porque a Câmara não disponibiliza os discursos, foto do deputado e URL de registro do evento via arquivos.

A figura 8 mostra a estrutura do banco de dados no qual os dados foram carregados.

Figura 8 - Modelo físico do banco de dados relacional



Fonte: próprio autor

4.4.2 Sincronização dos dados

Para manter o banco de dados sincronizado com os dados da Câmara foi escrito uma tarefa com o Quartz que periodicamente executa o processo de ETL descrito na seção 4.4.1. Para o contexto dos dados da Câmara, duas estratégias de sincronização são possíveis. A primeira consiste em apagar todos os dados do banco de dados, extrair os dados da Câmara e inseri-los novamente. Na segunda estratégia, os dados da Câmara são extraídos e comparados, com base no identificador do registro, com os dados existentes no banco de dados. Se o registro não existe, ele é inserido, caso contrário, é atualizado.

Na implementação da sincronização a primeira estratégia foi utilizada apenas para as “Despesas pela Cota Parlamentar”, pois a Câmara não disponibiliza um identificador para esses registros e não foi possível determinar um conjunto de valores que servisse para esse propósito. Para as demais informações foi possível implementar a segunda estratégia com algumas suposições sobre os dados da Câmara.

A primeira suposição é sobre os registros dos “Discursos”, “Autores das Proposições”, “Deputados membros dos órgãos” e “Deputados presentes em cada evento” e “Voto de cada parlamentar”. Eles não dispõem de um identificador emitido pela Câmara, mas foi possível definir um conjunto de valores nos registros que o identificasse unicamente, conforme o Quadro 4. E segundo, todos os registros de dados disponibilizados pela Câmara não são excluídos. Ou seja, só há atualização dos registros existentes ou inclusão de novos.

A motivação para utilizar a segunda estratégia é que ela possibilita a identificação de novos registros durante o processo de sincronização. Isso foi utilizado para detectar novas atividades de um deputado e enviar notificações via FCM para o aplicativo *mobile*.

Quadro 4 - Suposição de identificadores

(continua)

Informação	**Identificador**
Discursos	data do discurso e identificador do deputado
Autores das Proposições	Identificador do deputado e identificador da proposição

Quadro 4 - Suposição de identificadores

(conclusão)

Deputados membros dos órgãos	Identificador do deputado, identificador do órgão e data de início no cargo
Voto de cada parlamentar	Identificador do deputado e identificador da votação
Deputados presentes em cada evento	Identificador do evento e identificador do deputado

Fonte: próprio autor

4.4.3 HTTP API

Para expor os registros do banco de dados foram criados alguns *endpoints*, conforme a Quadro 5. Eles são utilizados pelo aplicativo mobile para exibir informações aos usuários.

Quadro 5 - Endpoints da aplicação backend

(continua)

Verbo HTTP	**Endpoint**	**Descrição**
GET	/api/v1/deputados	Retorna uma lista de deputados.
GET	/api/v1/deputados/{id}	Retorna informações sobre um deputado identificador por {id}.
GET	/api/v1/deputados/{id}/resumo	Retorna uma contabilização de recursos (discursos, despesas etc.) vinculado ao deputado identificador por {id}.
GET	/api/v1/deputados/{id}/despesas	Retorna uma lista de despesas custeadas pela cota parlamentar.
GET	/api/v1/deputados/{id}/despesas/tipos/resumo	Retorna o total de gastos, por tipo de despesa, vinculado ao deputado identificador por {id}.
GET	/api/v1/deputados/{id}/despesas/meses/resumo	Retorna o total de gastos, por mês de ocorrência da despesa, vinculado ao deputado identificador por {id}.

Quadro 5 - Endpoints da aplicação backend

(conclusão)

GET	/api/v1/deputados/{id}/cargos	Retorna uma lista de cargos em órgãos da Câmara ocupados pelo deputado identificado por {id}.
GET	/api/v1/deputados/{id}/discursos	Retorna uma lista de discursos realizados pelo deputado identificado por {id}.
GET	/api/v1/deputados/{id}/eventos	Retorna uma lista de eventos com presença registrada pelo deputado identificado por {id}.
GET	/api/v1/deputados/{id}/proposicoes	Retorna uma lista de proposições cujo deputado identificador por {id} é um dos autores.
GET	/api/v1/deputados/{id}/proposicoes/temas/resumo	Retorna uma contabilização das proposições, por tema, cujo deputado identificador por {id} é um dos autores.
GET	/api/v1/deputados/{id}/proposicoes/tipos/resumo	Retorna uma contabilização das proposições, por tipo, cujo deputado identificador por {id} é um dos autores.
GET	/api/v1/deputados/{id}/votos	Retorna uma lista de votos realizados pelo deputado identificado por {id}.

Fonte: próprio autor

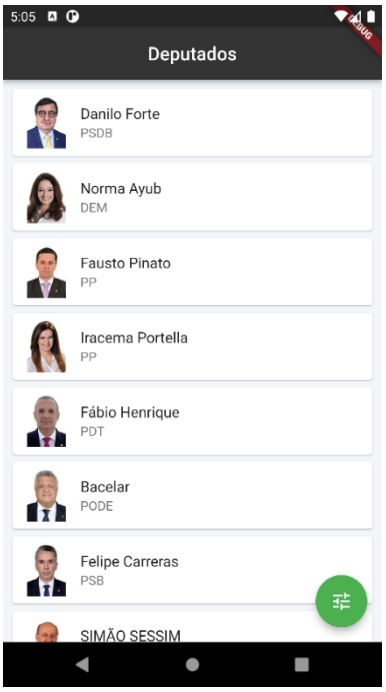
4.4.4 Aplicativo mobile

Nesta seção são apresentadas as telas desenvolvidas no aplicativo *mobile* e a navegação entre elas.

A Figura 9 mostra a tela inicial do aplicativo. Ela exibe uma lista de todos os deputados da legislatura corrente. Para cada deputado é exibido o nome, foto e

partido. O botão flutuante na parte inferior da tela abre uma caixa de diálogo, conforme a Figura 10, que permite filtrar os itens da lista. Os filtros disponíveis são: pelo nome, por unidade federativa, por partido e deputados seguidos.

Figura 9 - Tela inicial do aplicativo



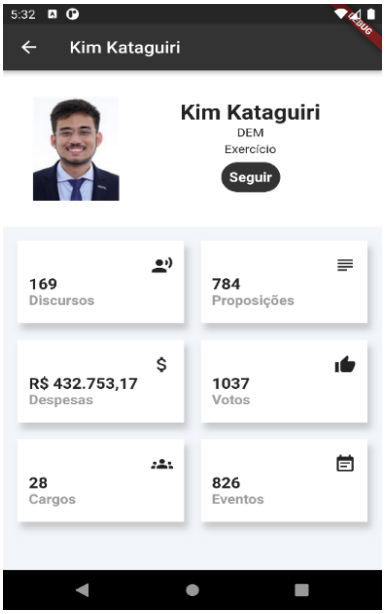
Fonte: próprio autor

Figura 10 - Filtros da lista de deputados

Fonte: próprio autor

A Figura 11 mostra a tela de resumo de atividades. Ela é aberta após clicar em um deputado na tela inicial e contém um resumo de todas as informações relacionadas a ele. O botão “Seguir” ativa o recebimento de notificações sobre novas atividades. Quando pressionado, o título do botão passa a ser “Deixar de seguir” e, se pressionado novamente, desativa o recebimento das notificações.

Figura 11 - Tela de resumo de atividades



Fonte: próprio autor

A Figura 12 mostra a tela de discursos. Ela é aberta após clicar na caixa “Discursos” da tela de resumo de atividades e contém os pronunciamentos realizados pelo deputado em eventos da Câmara. Cada item da lista exibe a data e sumário do discurso. Os itens podem ser filtrados pelo conteúdo da transcrição e data, conforme a Figura 13. Clicar no botão “Ver transcrição” abre uma caixa de diálogo contendo a transcrição do discurso, conforme a Figura 14.

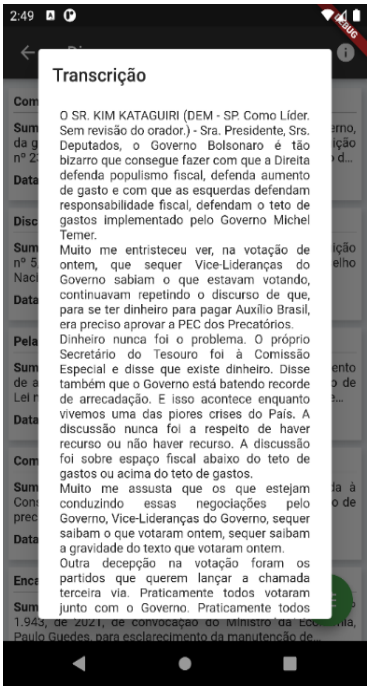
Figura 12 - Tela de discursos

Fonte: próprio autor

Figura 13 - Filtros da tela de discursos

Fonte: próprio autor

Figura 14 - Transcrição do discurso



Fonte: próprio autor

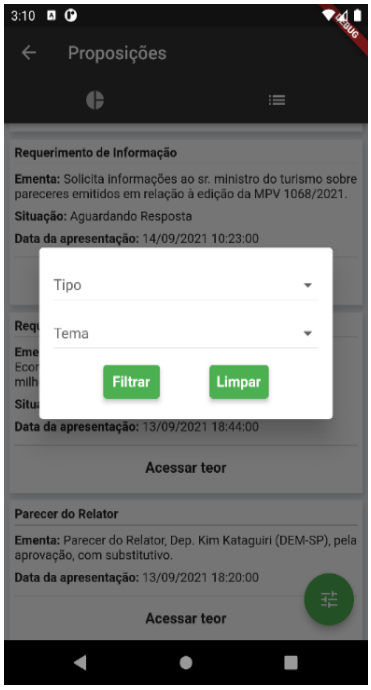
Ao clicar na caixa “Proposições” da tela de resumo de atividades é aberta a tela de proposições. Ela contém todas as proposições cujo deputado é um dos seus autores. Nessa tela é possível visualizar as proposições na forma de lista ou graficamente.

Na visualização em lista (Figura 15), para cada proposição é exibido o tipo, ementa, tema, situação e data da apresentação. Clicar no botão “Acessar teor” abre uma tela (Figura 17) que apresenta o conteúdo da proposição em PDF. Ainda na visualização em lista, os itens podem ser filtrados pelo tipo e tema, conforme a Figura 16.

Figura 15 - Tela de proposições

Fonte: próprio autor

Figura 16 - Filtros da tela de proposições



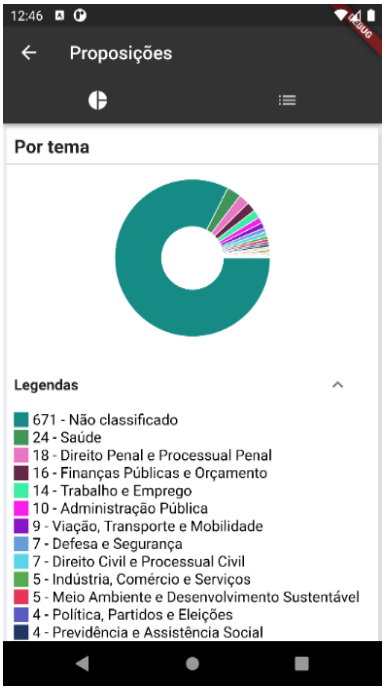
Fonte: próprio autor

Figura 17 - Tela de visualização do conteúdo da proposição

Fonte: próprio autor

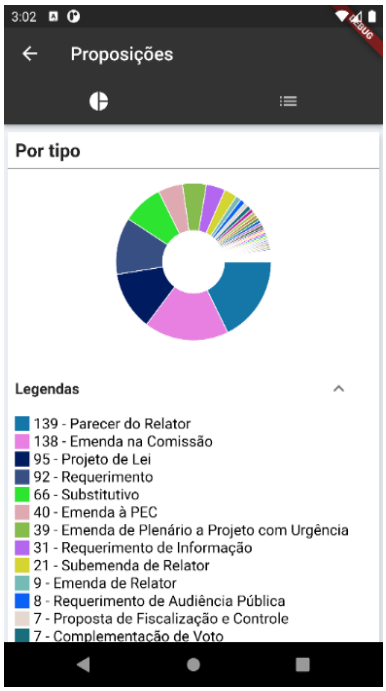
Na visualização gráfica, existem dois gráficos de setores que mostram a quantidade de proposições por tipo e por tema, conforme as Figuras 18 e 19.

Figura 18 - Gráfico de setores por tema de proposição



Fonte: próprio autor

Figura 19 - Gráfico de setores por tipo de proposição



Fonte: próprio autor

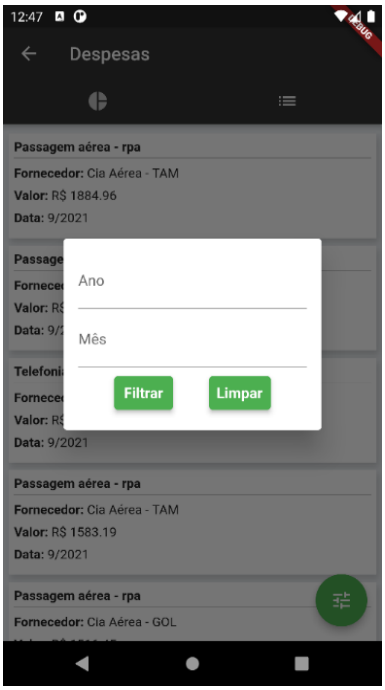
Ao clicar na caixa “Despesas” da tela de resumo de atividades é aberto a tela de uso da conta parlamentar. Ela contém as despesas custeadas pela cota parlamentar do deputado. Nessa tela é possível visualizar as despesas na forma de lista ou graficamente.

Na visualização em lista (Figura 20), para cada despesa é exibido o tipo, valor, fornecedor e data. Os itens podem ser filtrados pelo ano e mês em que a despesa foi descontada da cota, conforme a Figura 21.

Figura 20 - Tela de uso da conta parlamentar

Fonte: próprio autor

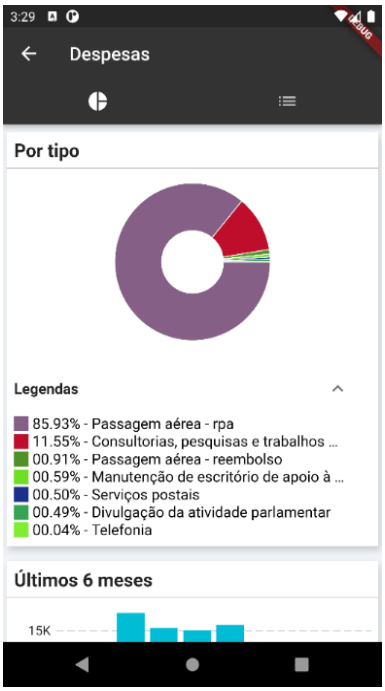
Figura 21 - Filtros da tela de uso da cota parlamentar



Fonte: próprio autor

Na visualização gráfica (Figura 22 e 23), existe um gráfico de setores que mostra o percentual gasto por tipo de despesa e um gráfico de barras que mostra o total gasto, por mês, nos últimos 6 meses.

Figura 22 - Gráfico de setores por tipo de despesa



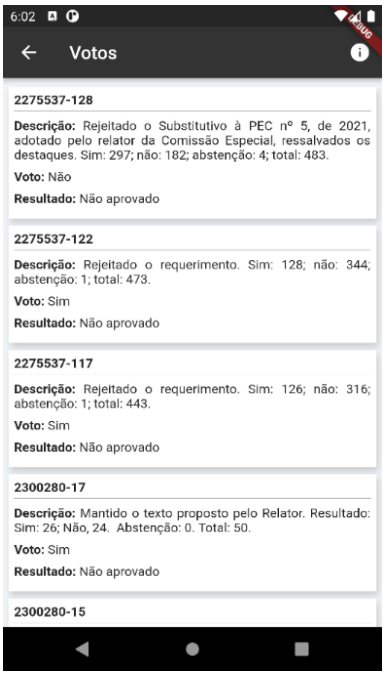
Fonte: próprio autor

Figura 23 - Gráfico de barras dos gastos dos últimos 6

Fonte: próprio autor

A Figura 24 mostra a tela de votos. Ela é aberta após clicar na caixa “Votos” da tela de resumo de atividades e contém os votos nominais, não secretos, realizados pelo deputado. Cada item da lista exibe o identificador, descrição, voto e resultado da votação.

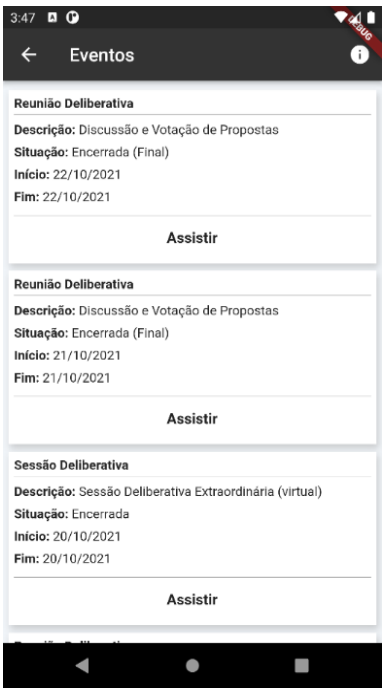
Figura 24 - Tela de votos



Fonte: próprio autor

A Figura 25 mostra a tela de participações em eventos. Ela é aberta após clicar na caixa “Eventos” da tela de resumo de atividades e contém as presenças em eventos do deputado. Cada item da lista exibe o tipo, descrição, situação, data início e fim. O botão “Assistir” abre uma tela que permite assistir a gravação do evento, conforme a Figura 26.

Figura 25 - Tela de participações em eventos



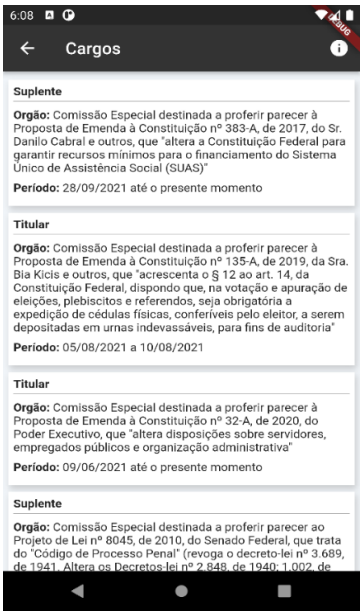
Fonte: próprio autor

Figura 26 - Tela de visualização do evento

Fonte: próprio autor

A Figura 27 mostra a tela de cargos em órgãos. Ela é aberta após clicar na caixa “Cargos” da tela de resumo de atividades e contém as participações do deputado como membro em órgãos da Câmara. Cada item da lista exibe o cargo, órgão e o período da participação.

Figura 27 - Tela de cargos em órgãos

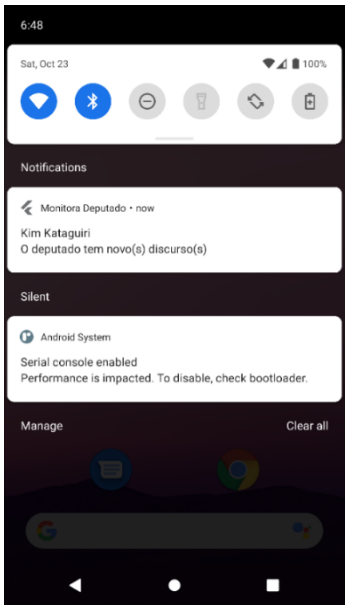


Fonte: próprio autor

4.4.5 Notificações

Durante o processo de ETL dos dados da Câmara é verificado se existem novos registros que não estão no banco de dados. Caso exista, uma notificação é enviada para o aplicativo mobile, por meio do FCM. A figura 28 mostra um exemplo de notificação. “Kim Kataguiri” e “O deputado tem novos(s) discurso(s)” é o título e conteúdo da notificação, respectivamente.

Figura 28 - Notificação



Fonte: próprio autor

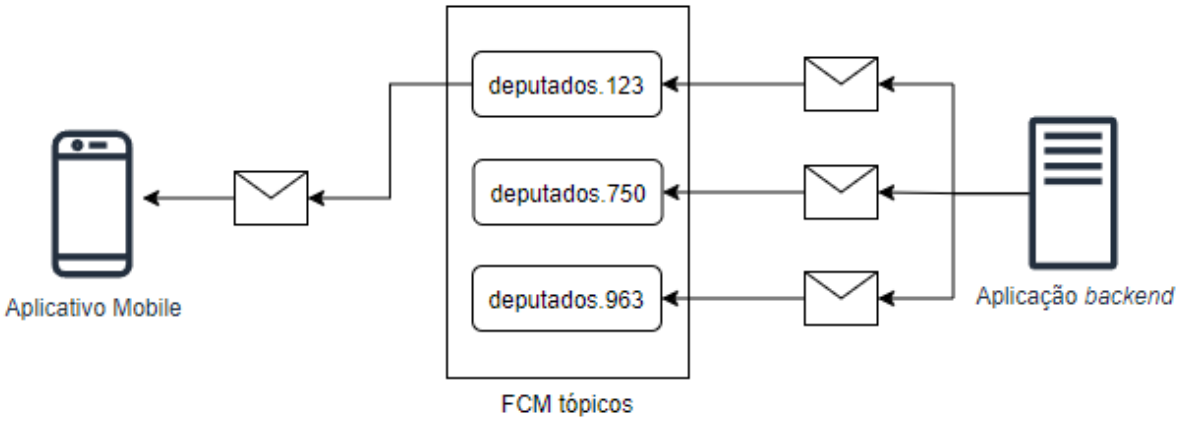
As notificações só são entregues aos usuários que seguem o deputado que está no título da notificação. Isso funciona da seguinte forma: a aplicação *back-end*, ao detectar um novo registro no processo de ETL, envia uma mensagem para um tópico do FCM nomeado como “deputados.{id}”, onde “{id}” é o identificador do parlamentar nos dados da Câmara. Por exemplo: foi identificado um novo discurso para o deputado João que têm o id igual a 123. Uma mensagem será enviada para o tópico “deputados.123” contendo uma notificação cujo título é João e o conteúdo é “O deputado tem novo(s) discurso(s)”. Se mais de um novo discurso for identificado para o deputado durante a ETL, apenas uma notificação é enviada.

Do lado do aplicativo *mobile*, quando o usuário clica no botão “Seguir” da tela de resumo de atividades do deputado João, uma inscrição no tópico “deputados.123” é realizada. A partir desse momento, o usuário passa a receber as notificações. Para deixar de seguir, e consequentemente não receber mais notificações sobre o deputado, basta clicar no botão “Deixar de seguir” na mesma tela.

As notificações só são exibidas se o aplicativo estiver em *background* ou encerrado. Além disso, não são enviadas notificações sobre novas despesas devido à dificuldade de identificar novos registros conforme discutido na seção 4.4.2.

A figura 29 ilustra o envio de mensagens da aplicação *backend*.

Figura 29 - Troca de mensagens entre a aplicação *backend* e o aplicativo *mobile*



Fonte: próprio autor

5 CONSIDERAÇÕES FINAIS

A iniciativa de dados abertos na Câmara dos Deputados permitiu, de forma simples e não burocrática, o acesso às informações sobre as suas atividades parlamentares. Com isso, foi possível desenvolver uma aplicação que permite o cidadão acompanhar o uso da cota parlamentar, votações, presença em eventos, discursos, cargos em órgãos e proposições propostas do deputado federal que ele ajudou a eleger.

Como resultado, este trabalho apresentou o desenvolvimento de um aplicativo *mobile*; a extração, transformação e carga de dados da Câmara; a implementação de uma HTTP API; a integração com um sistema de mensagem.

De forma geral, os resultados obtidos foram satisfatórios no que diz respeito as funcionalidades do aplicativo. Apenas na tela de visualização das votações não obtivemos um bom resultado pois não foi possível apresentar ao usuário a proposição que estava sendo votada. Isso se deve porque a Câmara dos Deputados não disponibiliza, em todos os casos, a proposição que de fato foi objeto da votação, mas sim uma lista das possíveis proposições que foram alvo da votação. Dessa forma, nem sempre era possível exibir com certeza a proposição que estava sendo votada. Mesmo com esse problema, a tela de votação foi mantida no aplicativo pois em alguns casos a descrição da votação dá uma indicação textual da proposição votada.

Algumas melhorias na aplicação ainda podem ser realizadas. No aplicativo *mobile*, seria interessante exibir os eventos que o deputado participou utilizando algum *widget* de calendário. Isso facilitaria o acesso aos eventos com base na data da sua ocorrência. E na aplicação *backend,* separar o processo de ETL da HTTP API e paralelizar a extração dos dados da Câmara para reduzir o tempo de execução do processo de ETL.

Na versão do sistema apresentada neste trabalho, foram exploradas funcionalidades básicas de consulta de informação e algumas visualizações gráficas de dados. No entanto, com a infraestrutura de ETL desenvolvida tornou-se possível que análises de dados mais complexas sejam realizadas.

Como trabalhos futuros, ficam as seguintes sugestões:

- Comparar o desempenho dos deputados com base no uso da cota parlamentar e matérias legislativas propostas e aprovadas.

- Ranquear os deputados que votam mais parecidos com outro deputado.
- Realizar mineração textual nos dados dos discursos para tentar identificar os assuntos ou temas mais abordados pelo deputado.

REFERÊNCIAS

BRASIL. Constituição da República Federativa do Brasil de 1988, 1988. Disponivel em: <http://www.planalto.gov.br/ccivil_03/constituicao/constituicaocompilado.htm>. Acesso em: 06 out. 2021.

BRASIL. **Portal Brasileiro de Dados Abertos**, 2021. Disponivel em: <https://dados.gov.br>. Acesso em: 06 out. 2021.

CÂMARA DOS DEPUTADOS. **Youtube**, 2021. Disponivel em: <https://youtu.be/bUQThBwN1nM>. Acesso em: 13 nov. 2021.

CÂMARA DOS DEPUTADOS. Assessoria de Imprensa - Comissões, 2021. Disponivel em: <https://www2.camara.leg.br/comunicacao/assessoria-de-imprensa/guia-para-jornalistas/comissoes>. Acesso em: 06 out. 2021.

CÂMARA DOS DEPUTADOS. Cota para o exercício da atividade parlamentar — Portal da Câmara dos Deputados, 2021. Disponivel em: <https://www2.camara.leg.br/transparencia/acesso-a-informacao/copy_of_perguntas-frequentes/cota-para-o-exercicio-da-atividade-parlamentar>. Acesso em: 21 out. 2021.

CÂMARA DOS DEPUTADOS. Dados Abertos da Câmara dos Deputados, 2021. Disponivel em: <https://dadosabertos.camara.leg.br/>. Acesso em: 21 out. 2021.

CÂMARA DOS DEPUTADOS. Deputados - Portal da Câmara dos Deputados, 2021. Disponivel em: <https://www2.camara.leg.br/transparencia/acesso-a-informacao/copy_of_perguntas-frequentes/deputados##17>. Acesso em: 21 out. 2021.

CÂMARA DOS DEPUTADOS. Entenda o Processo Legislativo, 2021. Disponivel em: <https://www.camara.leg.br/entenda-o-processo-legislativo>. Acesso em: 13 nov. 2021.

CÂMARA DOS DEPUTADOS. Guia para jornalistas - Câmara dos Deputados, 2021. Disponivel em: <https://www2.camara.leg.br/comunicacao/assessoria-de-imprensa/guia-para-jornalistas/camara-dos-deputados>. Acesso em: 06 out. 2021.

CÂMARA DOS DEPUTADOS. Infoleg. **Google Play**, 2021. Disponivel em: <https://play.google.com/store/apps/details?id=br.leg.camara.infolegmovel>. Acesso em: 01 out. 2021.

CÂMARA DOS DEPUTADOS. Pra que serve o Dados Abertos?, 2021. Disponivel em: <https://dadosabertos.camara.leg.br/howtouse/sobre-dados-abertos.html>. Acesso em: 06 out. 2021.

CÂMARA DOS DEPUTADOS. Saiba o que faz um deputado federal - Notícias - Portal da Câmara dos Deputados, 2021. Disponivel em: <https://www.camara.leg.br/noticias/545049-saiba-o-que-faz-um-deputado-federal>. Acesso em: 21 out. 2021.

DART. Dart overview. **Dart programming language**, 2021. Disponivel em: <https://dart.dev/overview>. Acesso em: 31 out. 2021.

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

FIREBASE. Firebase Documentation. **Firebase Cloud Messaging**, 30 out. 2021. Disponivel em: <https://firebase.google.com/docs/cloud-messaging>.

FLUTTER. Beautiful native apps in record time - Flutter. **Flutter**, 2021. Disponivel em: <https://flutter.dev/>. Acesso em: 30 out. 2021.

GAMFIG CORP. Monitora, Brasil! **Google Play**, 2021. Disponivel em: <https://play.google.com/store/apps/details?id=org.monitorabrasil>. Acesso em: 01 out. 2021.

HEUSER, C. A. **Projeto de Banco de Dados**. 4ª. ed. [S.l.]: Sagra Luzzatto, 1998.

HOHPE, G.; BOBBY, W. **Enterprise Integration Patterns:** Designing, Building And Deploying. [S.l.]: Pearson Education, Inc, 2004.

MANUAL. Manual dos Dados Abertos: Governo, 2011. Disponivel em: <https://www.w3c.br/pub/Materiais/PublicacoesW3C/Manual_Dados_Abertos_WEB.pdf>. Acesso em: 06 out. 2021.

MENEGUZ, G. Meu Deputado. **Google Play**, 2021. Disponivel em: <https://play.google.com/store/apps/details?id=com.meneguz.meudeputado>. Acesso em: 01 out. 2021.

OPEN GOVERNMENT DATA. **8 Principles of Open Government Data**, 2021. Disponivel em: <https://public.resource.org/8_principles.html>. Acesso em: 06 out. 2021.

ORACLE. Oracle Help Center. **The Java Language Specification**, 2021. Disponivel em: <https://docs.oracle.com/javase/specs/jls/se16/html/jls-1.html>. Acesso em: 10 out. 2021.

PASSOS, N. R. S. **Transformando dados em conhecimento : LADDRES, uma aplicação prática**. Monografia (graduação em Engenharia da Computação) - Universidade Federal de Sergipe. São Cristóvão, SE. 2018.

POSTGRESQL. PostgreSQL: The world's most advanced open source database, 2021. Disponivel em: <PostgreSQL: The world's most advanced open source database>. Acesso em: 06 out. 2021.

QUARTZ. Quartz Enterprise Job Scheduler. **Quartz**, 30 out. 2021. Disponivel em: <http://www.quartz-scheduler.org/>.

RIBEIRO, C. J. S.; ALMEIDA, R. F. **Dados Abertos Governamentais (Open Government Data):** Instrumento para Exercício de Cidadania pela Sociedade. XII ENANCIB - Encontro Nacional de Pesquisa em Ciência da Informação. [S.l.]: [s.n.]. 2011.

RICHARDSON, L.; RUBY, R. **RESTful Web Services**. 1ª. ed. [S.l.]: O'Reilly Media, 2007.

SENADO FEDERAL. Proposições - Senado Notícias, 2021. Disponivel em: <https://www12.senado.leg.br/noticias/glossario-legislativo/proposicao>. Acesso em: 21 out. 2021.

SILVEIRA, P. E. A. **Introdução à Arquitetura e Design de Software:** uma visão sobre a plataforma Java. [S.l.]: Casa do Código, 2013.

SOARES, J. R. B. O papel do deputado federal na República Federativa brasileira, 2011. Disponivel em: <https://www2.senado.leg.br/bdsf/bitstream/handle/id/242918/000926870.pdf?sequence=3&isAllowed=y>. Acesso em: 06 out. 2021.

SPRING. Home. **Spring**, 2021. Disponivel em: <https://spring.io/>. Acesso em: 06 out. 2021.

SPRING. Spring Data. **Spring**, 2021. Disponivel em: <https://spring.io/projects/spring-data>. Acesso em: 31 out. 2021.

SPRING. Spring Framework. **Spring**, 2021. Disponivel em: <https://spring.io/projects/spring-framework>. Acesso em: 30 out. 2021.

TELES, I. B., 2021. Disponivel em: <https://play.google.com/store/apps/details?id=br.leg.desafio.camarapopular>. Acesso em: 01 out. 2021.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante JULIO CESAR ALVES DE ARAUJO do Curso de CIÊNCIA DA COMPUTAÇÃO, matrícula 201620028 0094-8 telefone: 62 99405-7501 e-mail JULIO.JC_19@HOTMAIL, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado DESENVOLVIMENTO DE UMA APLICAÇÃO PARA ACOMPANHAMENTO DA ATIVIDADE PARLAMENTAR, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 14 de DEZEMBRO de 2021.

Assinatura do(s) autor(es): Julio Cesar Alves de Araujo

Nome completo do autor: Julio Cesar Alves de Araujo

Assinatura do professor-orientador: Max Gonzaga

Nome completo do professor-orientador: MAX GONZAGA DE OLIVEIRA