Desenvolvimento de Sistema Web para a Associação de Transporte Urbano de Vitória da Conquista (ATUV)

Adriano de Jesus Alves[1], Alexandro dos Santos Silva[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)
Avenida Sérgio Vieira Melo, 3.150, Zabelê – Vitória da Conquista – BA – Brazil

adrianojalves@gmail.com, alexandrossilva@ifba.edu.br

Abstract. *This paper describes a system for synchronizing the data of enrolled students between ATUV (Urban Transport Association of Vitória da Conquista) and the city's educational institutions. Using the concept of interoperability, this system will connect (or will be connected) to the various information systems of such institutions, enabling automatic data submission, as well as speeding up the re-registrantion process for student pass cards for use in public transport in the micipality of Vitória da Conquista.*

Resumo. *Este artigo descreve um sistema para a sincronização de dados de alunos matriculados entre a ATUV (Associação de Transporte Urbano de Vitória da Conquista) e as instituições de ensino da cidade. Utilizando o conceito de interoperabilidade, este sistema se conectará (ou será conectado) aos diversos sistemas de informação de tais instituições, possibilitando envio automático dos dados, bem como agilidade no processo de recadastramento de cartões de passe estudantil para uso no transporte público municipal de Vitória da Conquista.*

1. Introdução

Em Vitória da Conquista, a Associação de Empresas de Transporte Urbano (ATUV) é responsável pela geração, recarga e controle dos cartões de Bilhete Eletrônico Municipal (BEM). Os usuários do transporte coletivo utilizam estes cartões de acordo com a sua categoria: empresarial, estudantil e gratuidades (esta última aplicável a idosos, militares, carteiros e categorias especiais).

Para um aluno ter o direito ao passe estudantil, que permite a ele usufruir do desconto de 50% no valor da passagem, é necessário estar matriculado em uma instituição de ensino do município. Para tanto, a faculdade ou escola deve enviar à ATUV, a cada 3 meses, listas com os nomes de seus alunos. Estas listas são salvas em uma pasta compartilhada através do protocolo FTP (File Transfer Protocol). Entretanto, há casos em que os nomes dos alunos não constam nas listas enviadas, de modo a impedir a geração ou renovação da validade dos cartões BEM.

Neste artigo será apresentado um sistema de gestão de alunos matriculados com o qual pretende-se fornecer um protocolo de comunicação seguro entre a ATUV e as instituições de ensino, permitindo, dessa forma, maior eficiência na sincronização de dados e agilidade na atualização dos cartões dos alunos.

2. Arquitetura

Para definição da infraestrutura do sistema, considerou-se os conceitos de microsserviços e REST, além do uso de um banco de dados NoSql e da plataforma de virtualização Docker.

Microsserviços são pequenos serviços autônomos que funcionam juntos [Newman 2015]. Com base nisso, pode-se dizer que cada serviço possui uma função bem definida e única. No contexto do sistema aqui desenvolvido, foram concebidos 5 (cinco) microsserviços para: a) autenticação e autorização; b) API de recepção de dados; c) API de requisição de dados; d) consulta; e e) administração.

Em relação ao sistema gerenciador de banco de dados, optou-se, mais especificamente, pelo MongoDB, que se encontra entre os mais populares do momento. O banco de dados conta com apenas duas entidades: uma que reúne todos os dados de cada aluno matriculado e outra com os usuários que terão acesso ao sistema, dispensando, dessa forma, características típicas dos bancos de dados relacionais.

A integração com potenciais sistemas de instituições de ensino é implementada com base em API's REST, tratando-se de modelo desenvolvido por Roy Fielding para comunicação padrão utilizando-se do protocolo HTTP (Hypertext Transfer Protocol). Este modelo ganhou popularidade em função de suas características de flexibilidade, restrições, arquitetura e representação de diversos tipos de dados [Rodrigues e Nascimento 2020]. Uma API Rest possui quatro restrições de recurso: identificação, representação, resposta autoexplicativa e hipermídia.

Por fim, o Docker é utilizado para isolar os processos ou serviços do sistema operacional. Três contêineres são utilizados um para prover um ambiente de execução Java, um banco de dados MongoDB e um servidor HTTP que será utilizado pela interface front-end do sistema. Essa estrutura fornece flexibilização suficiente para utilizar as camadas da aplicação em vários ambientes e facilitar a escalabilidade. Essas características são essenciais para cumprir os requisitos que o sistema deve implementar.

3. Trabalhos Correlatos

Apesar do sistema aqui discutido possuir funcionalidades bastante específicas, foi importante buscar trabalhos que se relacionassem com a arquitetura, frameworks e/ou ferramentas usadas.

Suryotrisongko, Jayanto e Tjahyanto [Suryotrisongko et. al 2017] desenvolveram um aplicativo de reclamações públicas. Utilizado por governos e/ou repartições públicas, este aplicativo foi desenvolvido com base em uma arquitetura web, de modo a aplicar os conceitos de microsserviços para dividir as funcionalidades do aplicativo em diversas partes para atender a lógica de negócio elencada. Destaca-se também o uso do Spring Boot no backend e do Angular no front-end da aplicação.

Por sua vez, Gos e Zabierowski [Gos and Zabierowski 2020] fazem um estudo de comparação entre os modelos monolíticos e de microsserviços, de uma aplicação web desenvolvida em Java. No trabalho, os autores também utilizaram o Spring Boot e o Docker para a criação dos microsserviços e chegam à conclusão de que, nos casos em

que um sistema vier a receber muitas requisições, a arquitetura de microsserviços se mostra mais adequada, além de que facilitará a escalabilidade e a manutenção do sistema.

Por fim, Modugu e Farhat [Modugu and Farhat 2020] desenvolveram um aplicativo aplicando o conceito de Internet das Coisas (IoT – Internet of Things) utilizando Spring Boot e Angular na implementação. O sistema implementado considerou um caso de uso de gerenciamento de estoque com IoT industrial, apresentando uma solução de arquitetura inovadora.

4. Desenvolvimento

Para desenvolvimento do front-end do sistema, utilizou-se o Angular, um framework JavaScript open-source que implementa um padrão de arquitetura MVC (Model View Controller). O Angular também é caracterizado pelo uso de diretivas que permitem acrescentar propriedades ou elementos ao HTML, facilitando, dessa forma, o reuso de componentes nas aplicações. Outro recurso do framework consiste em ligações bidirecionais de dados (**two way data binding**), que permite a atualização automática da marcação HTML se houver mudanças de dados, assim como atualização de dados se houver mudança na marcação HTML [Zabot 2020].

A implementação do back-end, por sua vez, baseou-se no Spring. O Spring é um framework desenvolvido em Java, criado por Rod Johnson por volta de 2002 e que tem o intuito de facilitar a construção de aplicações que antes só eram possíveis com aquela linguagem através de EJB (Enterprise JavaBeans) [Gentil 2012]. O Spring engloba uma série de módulos e projetos que simplificam e flexibilizam a criação de aplicações. Entre eles, cita-se aqui o Spring Boot, que é usado neste trabalho, tendo sido desenvolvido a partir da necessidade do Spring suportar servidores embutidos. O Spring Boot também facilita a criação de aplicações Spring e sua posterior publicação por não necessitar de um servidor de aplicação. Com isso, o Spring Boot gerencia o servidor web controlando suas regras de negócio [Boaglio 2017]. Com essas facilidades, o conceito de microsserviços pode ser aplicado ao projeto, garantindo a flexibilização e o crescimento em larga escala, caso haja necessidade.

Com os recursos mencionados anteriormente, foi possível desenvolver API's que serão utilizadas para consulta e sincronismo de dados. Cada microsserviço oferece uma gama de recursos de acordo com a sua funcionalidade, sendo elas descritas nas subseções que se seguem abaixo.

4.1. Microsserviço de Autenticação e Autorização

O serviço de segurança, que é responsável pela autenticação e autorização de usuários, possui dois perfis, um para autenticação de usuários através da interface front-end fornecida pelo sistema e outro para os sistemas de terceiros (instituições de ensino) enviarem as informações dos seus respectivos alunos. Devido ao fato da autenticação ser única (ou seja, somente usuários cadastrados no banco de dados poderem ter acesso ao sistema), adotou-se o padrão de um único serviço para autenticação e autorização.

Após a autenticação, é retornado um token JWT, tratando-se este, pois, de um padrão aberto definido pelo RFC 7519, representando uma forma compacta e segura

para a troca de dados entre duas partes utilizando um objeto JSON. No serviço implementado, o token de autorização possui a seguinte composição: dados do usuário, relação de permissões e *refresh token* enviado na forma de cookie (este último para atualização do token quando o mesmo expira após 24h). O refresh token possui um timeout de 7 (sete) dias, após o qual o usuário deverá se autenticar novamente.

4.2. Microsserviço de API de Recepção de Dados

Trata-se de serviço que recepciona as informações dos alunos matriculados, cabendo à cada instituição de ensino criar e/ou incorporar, em caso de já possuir sistema legado, recurso de envio das informações de seus alunos. Para tal, a ATUV cadastrará, para cada instituição, um usuário cujo login e senha serão utilizados por ela para se autenticar, quando do envio de dados. Após isso, de posse do token de autorização e usando o método POST, o sistema externo poderá enviar, através do endpoint **/alunos**, uma lista de alunos contendo as seguintes informações de cada um deles: nome, número do documento de identidade, CPF, data de início, data de término, curso, turno, semestre/ano e status (matriculado ou cancelado).

4.3. Microsserviço da API de Requisição de Dados

Acionado através de interface front-end do próprio sistema, este serviço dispõe de recurso para consulta e importação dos alunos em sistemas externos mantidos por instituições de ensino. Para este caso, também será necessário que a instituição implemente e disponibilize link de requisição, login e senha para que o sistema desenvolvido neste trabalho possa se conectar e buscar as informações dos alunos matriculados naquela instituição. Inicialmente, será identificada a quantidade total de alunos que serão importados, seguindo-se a isso a requisição de dados propriamente dita; cada requisição retornará informações de até 10.000 alunos, de tal modo que será necessário que o sistema externo implemente mecanismo de paginação das requisições. Essa paginação é necessária pelo fato de que servidores HTTP possuem um limite de dados que podem ser enviados no corpo (body) de cada requisição; realizando-se um cálculo da quantidade média de bytes de cada bloco de informações contendo dados de um aluno, verificou-se que o total de 10.000 alunos por página não ultrapassaria 50% do limite do corpo das requisições que é admitido pelos principais servidores utilizados atualmente. Após a captura da resposta, os dados capturados são persistidos no banco de dados usado pelo sistema desenvolvido (MongoDB).

4.4. Microsserviço de Consulta

Este serviço é acessado através da interface front-end do próprio sistema desenvolvido usando o método GET e o *endpoint* **/alunos**, sendo incluídos na requisição, a título de parâmetros, nome ou parte do nome do aluno que se deseja consultar e o código da instituição de ensino (consultado previamente). O retorno da requisição consiste em uma lista com os dados dos alunos (em formato JSON), para posterior renderização pelo navegador.

4.5. Microsserviço de Administração

O microsserviço administrativo, também acessado exclusivamente pela interface front-

end do sistema, será responsável pela manutenção dos cadastros que permitirão o acesso aos demais módulos e microsserviços. Os usuários dos sistemas estão divididos em três perfis: a) administração, com acesso completo ao sistema; b) instituição, cujo acesso é restrito ao perfil pessoal; e c) operador, que além de acessar o próprio perfil permite consultar alunos ativos.

Os *endpoints* deste microsserviço foram implementados para auxiliar os administradores da ATUV na gestão da base de dados de alunos beneficiados com o Bilhete Eletrônico Municipal (BEM), de modo a permitir que se tenha conhecimento das instituições que estão sem informações de alunos matriculados e aquelas que estão há mais tempo sem enviar informações atualizadas de seus alunos.

5. Resultados

É possível prever melhorias significativas entre o quadro atual de gestão do Bilhete Eletrônico Municipal (BEM) vivenciado pela ATUV e a solução proposta aqui, quando ela vier a ser implantada (em andamento). Basicamente, conforme já mencionado anteriormente, há uma pasta compartilhada na qual são mantidos arquivos com listas de alunos enviados, via FTP, pelas instituições de ensino. Acessado determinado arquivo, realiza-se a pesquisa pelo nome do aluno procurado. Entre as dificuldades encontradas, podem ser citadas: a) dificuldade na localização dos arquivos pelo nome da instituição, dado que não há padronização de nomenclatura dos mesmos; b) existência de arquivos PDF em que não há possibilidade de consulta por nome de aluno, dado que as listas são incorporadas aos arquivos em formato de imagem; c) eventual indisponibilidade de acesso à pasta compartilhada; e d) eventuais transtornos aos alunos que, mesmo estando matriculados em suas instituições, ficam impossibilitados de serem recadastrados ou terem cartão emitido em função dos arquivos enviados estarem desatualizados ou não estarem disponíveis.

Outro problema desse modelo de gestão deve-se ao fato de que as listas devem ser enviadas a cada 3 (três) meses. Em função disto, o administrador da ATUV normalmente ordena os arquivos por data de envio e, após isso, remove aquelas que estão datadas há mais de três meses.

A solução aqui apresentada apresenta melhorias significativas. A interface front-end disponibiliza dashboard administrativo, no qual é permitido ao administrador verificar quais instituições estão com suas listas de alunos defasadas há mais tempo e aquelas que não possuem nenhum aluno cadastrado no sistema, conforme demonstrado na Figura 1.

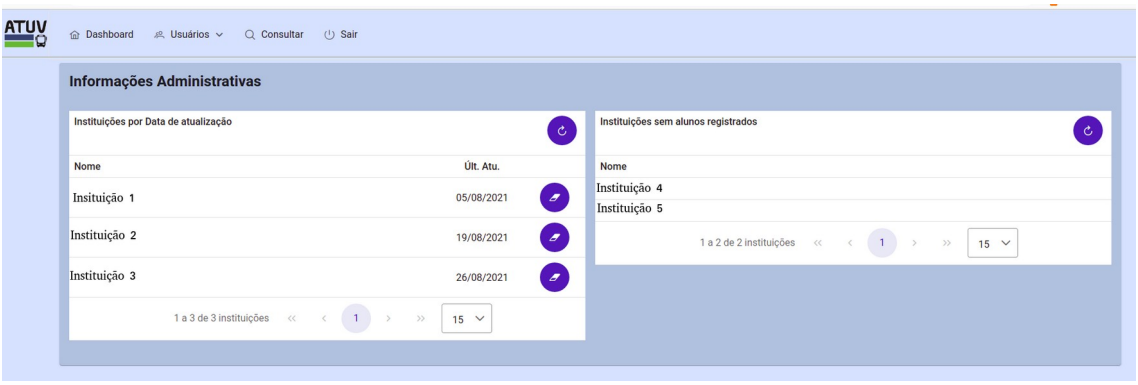

Figura 1. Dashboard administrativo

Na Figura 2, é apresentada a lista de instituições de ensino cadastradas acompanhada da possibilidade de inclusão, edição, exclusão e listagem de alunos. Para aquelas instituições que optarem pela utilização de API de acesso externo, a configuração do link de requisição dos dados dos alunos é realizada também neste painel. Por fim, também é possível executar a importação de planilhas eletrônicas em formato XLS/XLSX para as instituições que adotarem este método de troca de informações.

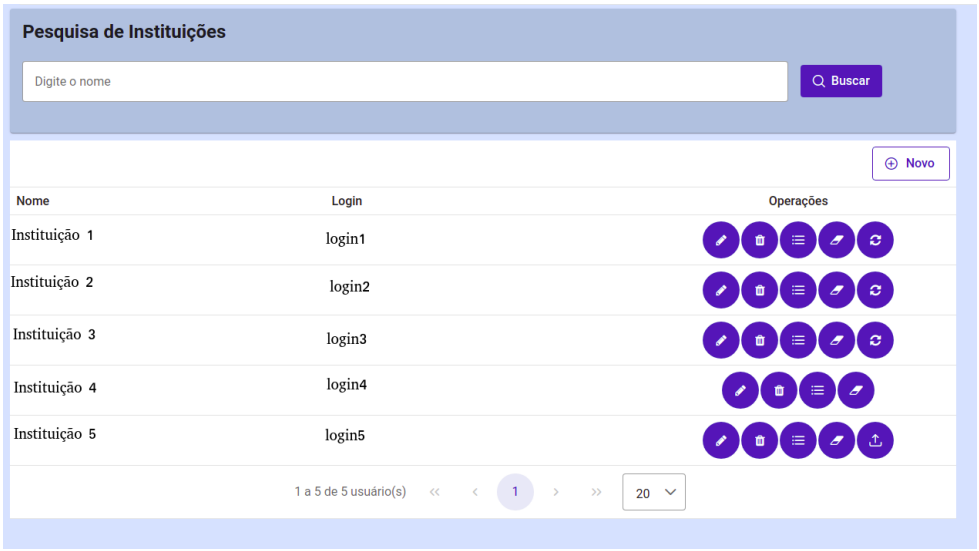

Figura 2. Consulta de lista de instituições cadastradas

Ao consultar a lista de alunos de determinada instituição (apresentada na Figura 3) o administrador passa a dispor de um novo recurso, que consiste na exclusão ou alteração do status de tais alunos (matriculado ou cancelado). Por sua vez, somente alunos com status de matriculado aparecerão nas consultas dos operadores responsáveis pela emissão de novos cartões de passe estudantil ou recadastramento daqueles que se encontram com validade expirada. No modelo de gestão atual, apesar da existência da pasta compartilhada via FTP, os nome de alunos com matrículas canceladas são enviados por e-mail; a ATUV, ao receber determinado nome, registra uma observação no sistema de recadastramento, informando da impossibilidade de realização do procedimento de recadastramento, mesmo que aquele nome permaneça presente no arquivo da respectiva instituição de ensino. Caso essa observação não seja detectada

pelo operador, o recadastramento do cartão será liberado de forma errônea. Com o sistema aqui desenvolvido, ao receber a notificação, o administrador alterará o status do aluno e o mesmo não será mais exibido nas consultas realizadas pelos operadores, bloqueando, dessa forma, recadastramentos inválidos.

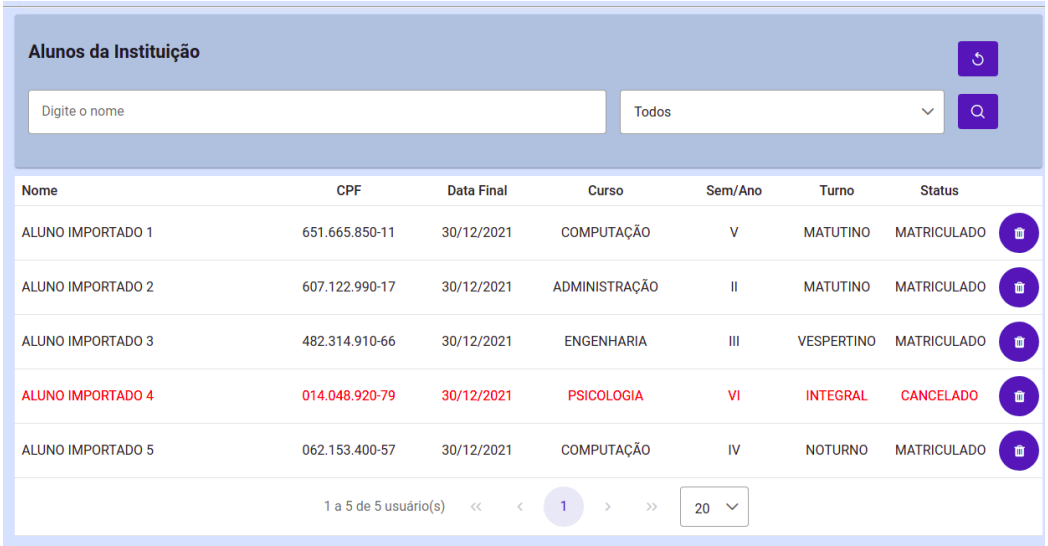

Figura 3. Consulta de lista de alunos pelo administrador

Por fim, a Figura 4 demonstra interface de consulta de alunos, sendo ele acessível tanto pelo administrador como pelos operadores. Nesta interface, selecione-se a instituição desejada e, após isso, informa-se o nome do aluno procurado (ou parte dele) para realizar a consulta. Os dados dos alunos que correspondem ao critério de busca são exibidos logo abaixo; caso não apareça o nome do aluno procurado, o precedimento de recadastramento não deverá ser realizado.

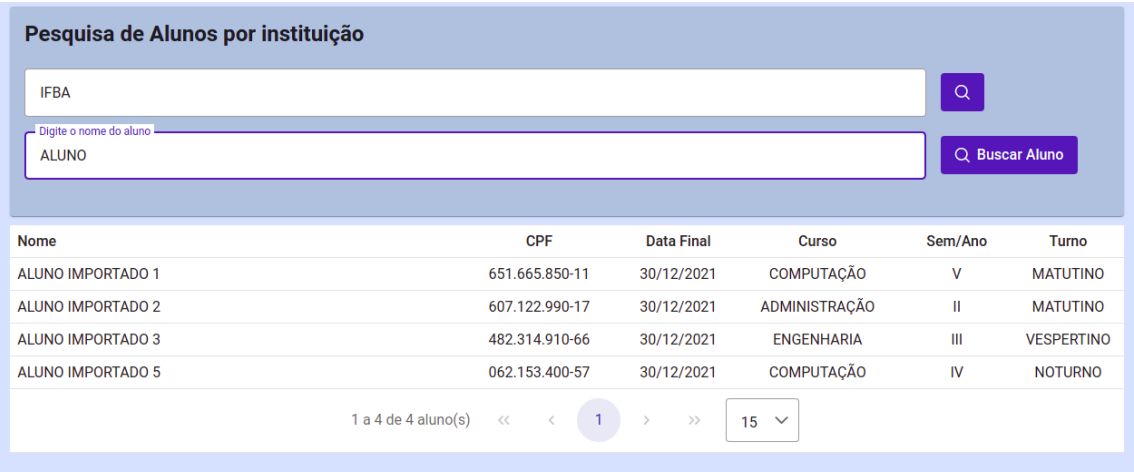

Figura 4. Consulta de alunos matriculados por instituição de ensino

6. Conclusões e Trabalhos Futuros

O objetivo deste artigo foi apresentar uma solução para a consulta de alunos matriculados nas diversas instituições de ensino da cidade de Vitória da Conquista, de modo a permitir que a ATUV passe a ter uma gestão mais eficiente dos cartões do

Bilhete Eletrônico Municipal. No momento o sistema se encontra em fase de implantação; as próximas etapas incluem a instalação do sistema em um servidor na nuvem e a apresentação de suas funcionalidades junto às instituições de ensino para que as mesmas decidam qual método de troca de informações de alunos é mais apropriado às suas necessidades e recursos.

Um possível trabalho futuro que representaria mais uma alternativa de envio dos dados dos alunos consistiria no desenvolvimento de um módulo desktop. Ao conceber este módulo, ele disporia de interface de configuração de conexão com algum banco de dados da instituição de ensino no qual são mantidos os dados dos seus respectivos alunos para posterior envio destes dados através da já implementada API de recepção de dados. O módulo teria o desafio de se conectar aos mais variados bancos existentes, proporcionando à instituição necessidade apenas de configurá-lo com usuário, senha e script de consulta SQL, por exemplo junto ao banco de dados para que seja possível obter os dados dos alunos e, após isso, enviá-los através da API de recepção de dados para a ATUV.

Referências

Boaglio, Fernando. (2017), “Spring Boot” Acelere o desenvolvimento de microsserviços, Casa do Código, Brasil.

Gentil, Efraim. (2012). “Introdução ao Spring Framework” https://www.devmedia.com.br/introducao-ao-spring-framework/26212, Novembro.

Gos, Konrad and Zabierowski, Wojciech. (2020). The Comparison of Microservice and Monolithic Architecture, Ukraine.

Modugu S.R., Farhat H. (2020) Implementation of the Internet of Things Application Based on Spring Boot Microservices and REST Architecture. In: Silhavy R., Silhavy P., Prokopova Z. (eds) Software Engineering Perspectives in Intelligent Systems. CoMeSySo 2020. Advances in Intelligent Systems and Computing, vol 1294. Springer, Cham. https://doi.org/10.1007/978-3-030-63322-6_3

Newman, Sam. (2015) Building Microservices, Edited by Mike Loukides and Brian MacDonald, O’Reilly Media, Inc., United States of America.

Rodrigues, I., Nascimento, Thiago. (2020) Integração de aplicações [recurso eletrônico], Fernando Sergio Soares Fagonde e Cassiano Ricardo Neubauer Moralles, SAGAH, Brasil.

Suryotrisongko, Hatma and Jayanto, Dedy Puji and Tjahyanto, Aris (2017). Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot, Information Systems International Conference, Indonesia.

Zabot, Diego. (2020), Aplicativos com Bootstrap e Angular: como desenvolver apps responsivos, Diego Zabot e Ecivaldo Matos, Érica, Brasil.