UNIVERSIDADE ESTADUAL DE SANTA CRUZ – UESC
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS - DCET
COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Pré-Projeto

Lana: Assistente pessoal para informações no âmbito da UESC

Ilhéus - Bahia
2018

Gabriel Rodrigues dos Santos

Lana: Assistente pessoal para informações no âmbito da UESC

Pré-projeto para o Relatório de Estágio, apresentado à Universidade Estadual de Santa Cruz - UESC, Colegiado de Computação, como parte das exigências para aprovação na disciplina Estágio Supervisionado.

Orientador: Prof Leard de Oliveira Fernandes

Ilhéus - Bahia
2018

Lana: Assistente pessoal para informações no âmbito da UESC

RESUMO

Com a disseminação de computadores pessoais e dispositivos móveis conectados a internet, junto ao avanço da tecnologia cognitiva na área de linguagem natural, softwares de assistência pessoal surgiram para uma variedade de propósitos. A interação entre usuário e assistente pessoal, normalmente, se dá através da troca de mensagens, onde o usuário faz uma pergunta ou uma requisição de serviço, e o assistente a responde, ou executa o serviço requisitado. Este projeto propõe a implementação de um *software* de assistência pessoal, nomeada como Lana, assim como o sistema de extração de dados, para realização dos serviços disponibilizados. Os serviços disponíveis pelo software de assistente pessoal serão escolhidos dentro do contexto da UESC, nos serviços de busca de informações sobre a instituição disponíveis no site da própria universidade e informações retiradas do portal Sagres, do aluno e do professor.

Palavras-chave: Assistente Pessoal, IBM Watson, Sistemas Distribuídos, Web Scraping.

SUMÁRIO

1 INTRODUÇÃO

Com a disseminação de computadores pessoais e dispositivos móveis conectados a internet, junto ao avanço da tecnologia cognitiva na área de linguagem natural, *softwares* de assistência pessoal surgiram para uma variedade de propósitos, tais como: gerenciamento de trabalho, organização e recuperação de informações e agendamento de atividades (MITCHELL *et al.*, 1994).

A interação entre usuário e assistente pessoal, normalmente, se dá através da troca de mensagens, onde o usuário faz uma pergunta ou uma requisição de serviço, e o assistente a responde, ou executa o serviço requisitado. Desta maneira, para construir um *software* de assistência pessoal, faz-se necessário a implementação de um meio de comunicação entre usuário e assistente, além da criação de serviços que possam ser requisitados pelo usuário.

Dado a necessidade de disponibilizar serviços pela internet, se faz necessária a criação de serviços de rede (*web services*), servidores conectados à internet, construídos com o propósito de suprir as necessidades de um *site* ou uma aplicação. Programas clientes utilizam APIs (*Application Programming Interfaces*) para se comunicar com estes *web services*. De modo geral, uma API disponibiliza dados e funções para facilitar a interação entre os *softwares* e permite que eles troquem informações. Uma API *web* é a interface de um serviço *web*, que recebe e responde requisições de clientes (MARK, 2013).

Nem todos os *sites* existentes fornecem APIs, e para extrair informações destes sistemas *web,* que não possuem API pública, é necessário o uso do *web scraping,* para obter estes dados de forma automatizada. *"Web scraping* é uma técnica de *software* destinada a extrair informações de *sites. Web scrapers* simulam a exploração humana na internet" (VARGIU; URRU, 2012, tradução nossa). Os *web scrapers* podem ser vistos como *bots* programados para buscar em uma página da *web* informações e/ou executar ações. Estes *bots* percorrem o *site* através do código fonte, normalmente em formato HTML, e também podem executar instruções em JavaScript na página.

Este projeto propõe a implementação de um *software* de assistência pessoal, nomeada como Lana, assim como o sistema de extração de dados, para realizar os

serviços disponibilizados, e um *bot,* que utiliza a API de um *software* de troca de mensagens, a fim de estabelecer uma comunicação entre o usuário final e a Lana. Para realizar a troca de informações entre os *bots* e o *software* de assistência pessoal, será necessário o desenvolvimento de APIs para ambos.

A Lana funcionará como um *middleware* que recebe as mensagens do usuário final, as interpreta e responde com o que for necessário, de acordo com o seu entendimento. Os serviços disponíveis pelo *software* de assistente pessoal foram escolhidos dentro do contexto da UESC, serviços de busca de informações sobre a instituição, disponíveis no *site* da própria universidade e informações retiradas do portal Sagres, do aluno e do professor, serão implementados nos extratores de dados.

1.1 Justificativa

A mobilidade e facilidade que traz um assistente pessoal, para acessar informações é, sem dúvidas, extremamente útil, principalmente quando essas informações estão disponíveis em locais de difícil acesso. Isto, aliado à possibilidade de contatar o assistente facilmente, a partir de aplicações de troca mensagens com API pública, sem a necessidade de adquirir ou criar novos aplicativos para este propósito, são os principais motivos para o desenvolvimento deste projeto.

Além disso, a implementação da Lana será realizada de maneira que o produto gerado seja genérico, isto é, somente os extratores de dados terão funções especificas, a assistente vai tratar todas as mensagens recebidas de maneira única, desta maneira, a implementação de novos serviços, dos mais variados contextos, não acarretará na necessidade de modificar o *software* de assistente pessoal, mas sim a criação de novos extratores de dados e na integração de contexto.

Tendo em vista o projeto como um todo, percebe-se que o mesmo abrange diversas áreas e disciplinas ministradas no curso de Ciência da Computação, dentre elas: linguagens de programação, redes de computadores, sistemas distribuídos, inteligência artificial, organização e recuperação de informações, banco de dados, interface-homem-máquina, engenharia de software, dentre outros. Portanto, um projeto com tantas áreas abrangentes, justifica-se passível de trabalho de conclusão de curso.

1.2 Objetivos

1.1.1 Gerais

Implementar um *software* de assistente pessoal capaz de receber mensagens textuais, interpretá-las, ativar serviços quando necessário, e retornar ao usuário final uma resposta trivial ou uma execução de um serviço requisitado.

1.1.2 Específicos

a) Implementar *chat bot* utilizando API de um aplicativo de troca de mensagens, para ser utilizado como meio de comunicação textual entre usuário final e assistente pessoal.
b) Implementar API CRUD de banco de dados para armazenar informações dos usuários.
c) Criar e configurar intenções, contextos e diálogos, utilizando API de reconhecimento de linguagem natural.
d) Implementar API, da assistente pessoal, para receber mensagens textuais e responder mensagens, contendo textos ou links para imagens e arquivos.
e) Implementar *bot* Sagres, que será responsável por executar serviços que são resolvidos através do Portal Sagres, como, por exemplo, obter horários do semestre e listar disciplinas.
f) Implementar *bot* UESC, que será requisitado para executar os serviços dependentes do *site* da UESC, como, por exemplo, listar os cursos de graduação disponíveis e listar os últimos editais publicados.
g) Implementar *bothub* UESC, um indexador de *bots* que realizam ações no contexto da universidade e indexar os *bot* Sagres e *bot* UESC ao *bothub.*

2 REVISÃO DA LITERATURA

Segundo Enembreck e Barthes (2002), o papel principal de um assistente pessoal é isentar o usuário de realizar tarefas repetitivas ou entediantes. Os autores

também explicam que as aplicações de um assistente pessoal podem variar de pesquisas na internet até mesmo tarefas colaborativas.

Wooldridge (2002, p. XIII) apresenta sistemas multiagente como sistemas compostos por múltiplos elementos computacionais capazes de interagir, conhecidos como agentes. O autor também conceitua um agente como:

> Um sistema computacional com dois importantes recursos. Primeiro, eles são pelo menos até certo ponto capazes de ações autônomas – de decidir por si mesmos o que eles precisam fazer para satisfazer seus objetivos. Segundo, eles são capazes de interagir com outros agentes – não simplesmente trocando dados, mas com engajamento em atividades sociais que nós realizamos diariamente em nossa vida: cooperação, coordenação, negociação e coisas do gênero.

Nesse sentido, Reateguil, Ribeiro e Boff (2008) propõem um sistema multiagente para o controle de um assistente pessoal, que é explicado pelos autores como “cada agente controla uma funcionalidade especifica, e um agente mediador define qual deles deve entrar em ação a cada momento” (REATEGUI; RIBEIRO; BOFF, 2008).

O agente mediador pode ser visto como um *middleware*. Baker e Apon (2001) expõem *middleware* como uma camada de *software* localizada entre o sistema operacional e a aplicação, e que mais recentemente ressurgiu como um meio de integrar *softwares* executados em um ambiente heterogéneo.

Foi apresentado na seção 1 a necessidade de criação de um meio de comunicação entre usuário e assistente pessoal. Esta comunicação é realizada a partir da troca de mensagens, logo, é necessário o entendimento de linguagem natural, e este recurso pode ser adquirido a partir de ferramentas de terceiros, como é apresentado por Yan *et al.* (2016).

Como explicado na seção 1, a implementação de serviços de um assistente pessoal pode levar a criação de *bots scrapers.* A busca de informações na *web,* através de *web scraping,* é apresentada por Vargiu e Urru (2012), onde os autores utilizam a técnica para publicidade filtrada baseada nos anúncios mais relevantes para uma página da *web*. A extração de dados também é apresentada por Soediono (1989) em seu livro “*Web* Scraping *with Python*”.

3 MATERIAIS E MÉTODOS

Para a realização deste projeto é necessária a utilização de algumas ferramentas de trabalho, e estas estão listadas na subseção materiais e em seguida, na subseção métodos, é apresentado o uso delas em cada do trabalho.

3.1 Materiais

3.1.1 JavaScript e Node.JS

JavaScript é uma linguagem de programação de alto nível, dinâmica, não tipificada, interpretada, que fornece os todos os recursos necessários para ser orientada a objeto e funcional (FLANAGAN, 2011). Originalmente ela foi criada como parte dos navegadores *web* para execução de instruções *client-side,* utilizando o interpretador de JavaScript do Google conhecido como V8.

O Node.JS é um interpretador *server-side* de JavaScript, ele foi baseado na implementação do V8, é implementado principalmente nas linguagens de programação C e C++, com ênfase em fornecer um bom desempenho e baixo consumo de memória(TILKOV; VINOSKI, 2010).

3.1.2 Python

Python é uma poderosa linguagem de programação de fácil entendimento. Ela possui estruturas de dados de alto nível e uma abordagem simples, mas efetiva, de programação orientada a objeto. Python tem uma natureza interpretada, ideal para desenvolvimento de aplicações de maneira rápida e é comumente utilizada na maioria das plataformas(SWAROOP, 2003).

3.1.3 Telegram

O Telegram é um aplicativo popular de troca de mensagens baseado em plataforma de código livre (SUTIKNO *et al.*, 2016). O Telegram é uma aplicação totalmente grátis e com uma interface simples, disponível para *smartphones* e computadores pessoais com aplicação *desktop* ou aplicação *web*.

Além disso, o Telegram também disponibiliza uma API para criação de *bots* na plataforma, desta maneira usuários podem interagir com os *bots*, enviando mensagens e comandos. O controle dos *bots* é feito através de requisições HTTPS para a API pública do Telegram.

3.1.4 IBM Watson, IBM Bluemix e Watson Assistant

"O IBM Watson é um supercomputador que combina inteligência artificial e software analítico para oferecer serviços diversos." (PITON, 2017). O Watson possui diversas APIs e serviços na área de computação cognitiva, elas possuem documentação e estão disponíveis para uso na plataforma IBM Bluemix, dentre eles está o serviço Watson Assistant. Com este serviço é possível construir soluções que entendam linguagem natural e responda de maneira similar a uma conversa entre humanos (MILLER, 2017).

É disponibilizado pelo serviço um painel de controle, e através deste painel é possível criar novas *Intents* e *Entities* dentro do *Workspace*. O *Workspace* é a área de trabalho onde será definido as opções de diálogos. *Intents* são as intenções do usuário, ou seja, as ações que o usuário pretende realizar com o serviço. Já *Entities* são as entidades que podem aparecer durante a conversação, como locais, datas, horários ou códigos específicos de algum objeto. Após a criação de diálogos utilizando as *Intents* e *Entities* é possível utilizar o Watson Assistant. Ao enviar uma mensagem de texto, o Watson vai retornar quais são as *Intents* e *Entities* na mensagem e a sua taxa de confiabilidade.

3.1.5 BaaS e Back4App

Um BaaS (*Backend as a Service*) pode ser visto como um serviço que auxilia a conexão entre o *backend* e o *frontend* de uma aplicação. O BaaS ajuda os desenvolvedores a acelerar o desenvolvimento de software e simplificar a criação de APIs. Com ele não é necessário desenvolver todo o *backend* de um aplicativo, apenas utilizar o BaaS para criar as APIs e conectar à aplicação(BATSCHINSKI, 2016).

Back4App é uma plataforma BaaS, onde é possível criar e hospedar APIs para aplicações *web* e móveis de maneira mais rápida. O Back4App cria toda a

estruturação de banco de dados e API CRUD, além de outros recursos mais avançados para facilitar o gerenciamento de sistemas e acelerar o desenvolvimento.

3.1.6 Heroku

"Heroku é uma plataforma de nuvem baseada em um sistema de contêiner gerenciado, com serviços e dados integrados e um poderoso ecossistema, para implementar e executar aplicativos modernos."(HEROKU, 2018). Com o Heroku não é necessário desenvolver toda a estruturação de um servidor *web*, somente a integração da aplicação nele.

3.1.7 VPS *Hosting* e DigitalOcean Droplet

VPS (*Virtual Private Server*) *Hosting* é a hospedagem de máquinas virtuais, vendido por empresas como um serviço. A empresa DigitalOcean fornece um serviço de VPS *Hosting* chamado DigitalOcean Droplet.

Um DigitalOcean Droplet é um VPS com recursos adicionais de armazenamento, segurança e monitoramento para executar facilmente os aplicativos de produção(DIGITALOCEAN, 2018).

3.1.8 GitHub

GitHub é uma plataforma para hospedagem de código-fonte. A plataforma oferece a hospedagem gratuita de projetos que utilizam controle de versão usando o Git e oferece ferramentas que ajudam aos desenvolvedores colaborarem entre si. O GitHub também fornece um quadro virtual para Kanban, uma metodologia ágil de desenvolvimento de *software.*

3.2 Métodos

A implementação da assistente pessoal será modularizada, cada módulo com uma funcionalidade especifica. Portanto, serão implementados módulos de interface,

banco de dados, entendimento de linguagem natural, fachada, assistente pessoal, indexador de *bots* e os *bots scrapers.*

O desenvolvimento será realizado com base na metodologia ágil Kanban, e para melhor interatividade com a metodologia, será utilizado a ferramenta de quadro virtual disponível no GitHub, durante o desenvolvimento o quadro irá conter todas as tarefas em andamento, finalizadas e que ainda precisam ser realizadas. Além disso, todos os códigos fontes dos módulos implementados terão controle de versão Git e serão armazenados no GitHub.

O primeiro módulo a ser implementado será o de interface, este ficará responsável por lidar com a troca de mensagens de texto com o usuário final. As mensagens recebidas não serão interpretadas neste módulo, apenas serão encaminhadas para o módulo de fachada, em seguida a reposta obtida da fachada será encaminhada para o usuário. Além da implementação da troca simples de mensagem, também será preciso a criação de um outro *endpoint* para acesso direto da Lana, sem a necessidade de passar por uma requisição na fachada. A implementação deste módulo será realizada utilizando a linguagem de programação JavaScript e o interpretador Node.JS. As trocas de mensagens ocorrerão através de um *bot* do aplicativo de mensagens Telegram, para a criação desse *bot* será utilizada a API de *bots* disponibilizada pelo próprio Telegram. Faz-se necessário que o *bot* esteja sempre disponível para conversação, portanto este módulo será hospedado na plataforma Heroku.

Em seguida será realizada a implementação do módulo de banco de dados, será utilizado a plataforma BaaS Back4App devido a sua facilidade para criar e gerenciar o banco de dados na nuvem, além disso, o Back4App também disponibiliza uma SDK (*Software Development Kit*) para JavaScript, fazendo com que seja preciso apenas a criação de novas funções especificas para o armazenamento das informações necessárias. Este módulo pode ser visto como uma fachada para o acesso ao Back4App. Todas as funções especificas criadas serão feitas em JavaScript e por fim hospedadas na própria plataforma do Back4App, estas são chamadas de *Cloud Functions.*

Ao finalizar o desenvolvimento do módulo de banco de dados será iniciada a implementação do módulo de entendimento de linguagem natural, assim como o módulo de banco de dados, este funcionará como uma fachada. Para implementar este módulo será utilizado JavaScript, Node.JS e o serviço da IBM, o Watson

Assistant. Entretanto, antes de implementar a fachada para o Watson é preciso criar o *workspace* do projeto no serviço na plataforma da IBM Bluemix, em seguida dentro deste *workspace* é necessário criar todas as *Intents* e *Entities* que serão utilizadas pela assistente pessoal, em seguida será realizado o desenvolvimento dos diálogos adicionando atributos específicos às respostas do Watson quando em uma mensagem recebida for quando reconhecido alguma *Intent* ou *Entity*. Por fim, com o *workspace* devidamente criado e configurado, será realizada a implementação da fachada para facilitar a comunicação entre a Lana e o Watson.

O funcionamento do módulo de fachada é relativamente simples, este será a API da assistente pessoal e ficará responsável por receber as requisições dos módulos de interface e encaminhar diretamente para a Lana. Este módulo ficará hospedado na plataforma Heroku e será utilizada a linguagem de programação JavaScript para a sua criação.

O módulo da assistente pessoal será a implementação da Lana em si, e ele será responsável pela comunicação entre todos os outros módulos. Nele será realizada a decisão de criar novos usuários e inserir novas informações no banco de dados, e também será responsável por enviar a mensagem de texto do usuário ao Watson e realizar as ações necessárias para suprir as necessidades daquela mensagem, como por exemplo, iniciar algum serviço de um *bot* que está disponível no módulo indexador de *bots,* receber a resposta deste serviço e envia-la ao usuário final. Este módulo será implementado em JavaScript e Node.JS e ficará hospedado na plataforma Heroku. O fluxo de comunicação entre todos os módulos pode ser visto na figura 1.

A fim de reduzir o acoplamento entre módulos, será implementado um módulo indexador de *bots*, batizado de BotHub. Este irá indexar todos os *bots* que possuam o mesmo contexto de serviço, por exemplo, no BotHub UESC será indexado todos os *bots* que realizam serviços no contexto da universidade. Desta maneira, a assistente pessoal não precisa saber qual *bot* é responsável por um serviço especifico, mas somente em qual contexto esse serviço se encaixa e requisitar o serviço ao BotHub responsável, como pode ser observado na figura 1. A implementação deste modulo será realizada com a linguagem de programação Python e a hospedagem na plataforma Heroku.

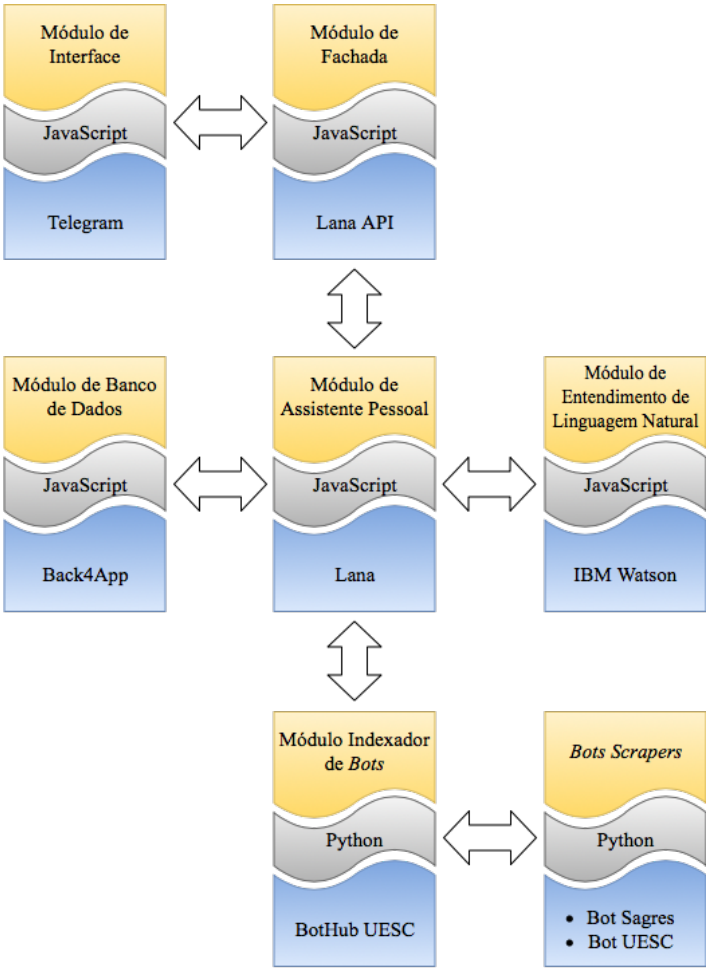

Figura 1 - Fluxo de comunicação entre módulos

Por fim será realizada a implementação dos *bots scrapers*, estes serão os responsáveis pela execução dos serviços em si. Serão implementados, em Python, os *bots* Sagres e UESC. O primeiro será responsável por obter as informações dos usuários do portal Sagres diretamente do *site*, para a execução dos serviços no portal são necessárias as credenciais de autenticação do usuário. Já o *bot* UESC ficará com a tarefa de obter as informações diretamente do *site* da UESC, para a execução dos serviços deste *bot* não haverá necessidade de passar informações extras.

Ao finalizar a implementação dos módulos, será realizado os testes necessários para certificar que eles estejam funcionando corretamente, corrigir possíveis falhas e otimizar o funcionamento geral.

4 CRONOGRAMA

Tabela 1 – Cronograma

Tarefas	Data de Início	Data de término
Levantamento de ferramentas e estudo de viabilidade	07/08/2018	21/08/2018
Pré-Projeto	21/08/2018	04/09/2018
Módulos de Interface, Banco de Dados, Entendimento de Linguagem Natural, Fachada e Assistente Pessoal	04/09/2018	18/09/2018
BotHub UESC, *bot* Sagres e *bot* UESC	18/09/2018	09/10/2018
Escrita da Monografia	09/10/2018	23/10/2018
Escrita da Monografia	23/10/2018	06/11/2018
Escrita da Monografia	06/11/2018	20/11/2018
Escrita da Monografia	20/11/2018	02/12/2018
Revisão de Conteúdo e Montagem de Apresentação Oral	02/12/2018	07/12/2018

REFERÊNCIAS

BAKER, Mark; APON, Amy. **Middleware**. *International Journal of High Performance Computing Applications*, 2001.

BATSCHINSKI, George. ***What is a Backend as a Service?*** Disponível em: <https://blog.back4app.com/2016/01/11/what-is-a-backend-as-a-service/ - more-705> Acesso em: 4 set. 2018.

DIGITALOCEAN. ***DigitalOcean Droplets***. Disponível em: <https://www.digitalocean.com/products/droplets/> Acesso em: 4 set. 2018.

ENEMBRECK, F.; BARTHES, J.-P. **Personal assistant to improve CSCW**. *The 7th International Conference on Computer Supported Cooperative Work in Design*, 2002. Disponível em: <http://ieeexplore.ieee.org/document/1047710/> Acesso em: 4 set. 2018.

FLANAGAN, David. ***JavaScript:*** The Definitive Guide 6th Edition. 2011.

HEROKU. ***The Heroku Platform***. Disponível em: <https://www.heroku.com/platform> Acesso em: 4 set. 2018.

MARK, Massé. ***REST API Design Rulebook***. 2013.

MILLER, M. ***IBM Bluemix Docs Conversation***. Disponível em: <https://console.bluemix.net/docs/services/conversation/index.html#about> Acesso em: 4 set. 2018.

MITCHELL, Tom M. et al. **Experience with a learning personal assistant**. *Communications of the ACM*, 1994.

PITON, Otávio Henrique Gotardo. **AUTOMAÇÃO RESIDENCIAL UTILIZANDO A PLATAFORMA EM NUVEM IBM BLUEMIX**. 2017.

REATEGUI, Eliseo; RIBEIRO, Alexandre; BOFF, Elisa. **Um Sistema Multiagente Para Controle De Um Assistente Pessoal Aplicado a Um Ambiente Virtual De Aprendizagem**. *Renote*, 2008.

SOEDIONO, Budi. ***Web Scraping with Python***. 1989.

SUTIKNO, Tole et al. **WhatsApp, Viber and Telegram which is Best for Instant Messaging?** *International Journal of Electrical and Computer Engineering (IJECE)*, v. 6, n. 3, 1 jun. 2016. Disponível em: <http://www.iaescore.com/journals/index.php/IJECE/article/view/443> Acesso em: 4 set. 2018.

SWAROOP, Ch. **A Byte of Python**. *A Byte of Python*, 2003.

TILKOV, Stefan; VINOSKI, Steve. **Node.js**: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, 2010.

VARGIU, Eloisa; URRU, Mirko. **Exploiting web scraping in a collaborative filtering- based approach to web advertising**. *Artificial Intelligence Research*, v. 2, n. 1, 20 nov. 2012. Disponível em: <http://www.sciedu.ca/journal/index.php/air/article/view/1390> Acesso em: 4 set. 2018.

WOOLDRIDGE, Michael. **Introduction to Multiagent Systems**. *Information Retrieval*, 2002.

YAN, Mengting et al. **Building a Chatbot with Serverless Computing**. 2016.