# 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** (SAP) surgiram para uma variedade de propósitos, tais como: gerenciamento de trabalho, organização e recuperação de informações e agendamento de atividades (MITCHELL *et al.*, 1994).

O objetivo destes *softwares* é auxiliar ou substituir seus usuários em determinadas tarefas, deixando-os livres para realizar atividades mais importantes. Convergindo aos conceitos de SAP, expostos por ZAMBIASI e RABELO (2012), Enembreck e Barthes (2002) explicam que o papel principal de um **assistente pessoal** (PA) é isentar o usuário de realizar tarefas repetitivas ou entediantes. Além disso, os autores deixam claro que as aplicações de um PA podem variar de pesquisas na internet até mesmo tarefas colaborativas.

A interação entre usuário e PA, 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, a construção de um SAP cria a necessidade de implementação de um meio de comunicação entre usuário e assistente, além da criação de serviços que possam ser requisitados pelo usuário. Visto que esta comunicação é realizada através da troca de mensagens, conclui-se que é necessário o entendimento de linguagem natural por parte do assistente, objetivando a compreensão das requisições realizadas pelo usuário.

Tendo em vista os objetivos de um PA e as suas principais necessidades, pode-se concluir que para a implementação deste *software* é indispensável o desenvolvimento de um meio de comunicação, armazenamento de dados, entendimento de linguagem natural e por fim a execução dos serviços necessários. Desta maneira, um SAP pode ser visto como uma grande aplicação monolítica, e a construção de um *software* desta maneira gera acoplamento e dificuldade de manutenção. Entretanto, visando evadir-se destes problemas, divide-se as necessidades do SAP em agentes capazes de executar tarefas especificas, criando assim um sistema multiagente.

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, ReateguiI, Ribeiro e Boff (2008) propõem um sistema multiagente para o controle de um PA, que é explicado pelos autores: “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.

Desta maneira, pode-se obter soluções adquiridas a partir de ferramentas de terceiros, por exemplo, o recurso de entendimento de linguagem natural pode ser obtido com o IBM Watson, como é apresentado por YAN *et al.* (2016), PITON (2017) e MOSTAÇO *et al.* (2018). Não restrito ao entendimento de linguagem, outros recursos como a interface de comunicação, i.e. o meio de troca de mensagens pode ser modularizado e obtido a partir de outras ferramentas, como é mostrado por MOSTAÇO *et al.* (2018) onde é utilizado a API do aplicativo de mensagens Telegram para a comunicação com o usuário.

A comunicação entre os agentes, ou módulos, de um SAP geralmente é realizada através da internet, pois quase sempre os módulos estão em sendo executados em computadores e locais distintos. Sendo assim, um SAP implementado de maneira modularizada pode ser considerado um sistema distribuído. TANENBAUM e VAN STEEN (2013) conceituam um sistema distribuído como uma coleção de computadores independentes que cooperam para resolver uma tarefa, entretanto o usuário final os enxerga como um só. COULOURIS, DOLLIMORE e KINDBERG (2012) acrescentam ao conceito de sistema distribuído como sendo um sistema no qual *softwares* ou *hardwares* conectados à rede comunicam-se e coordenam as suas ações por meio de troca de mensagens.

Dada 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 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).

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. Nesse sentido, um SAP pode utilizar uma API disponibilizada por outro sistema para a obtenção de dados ou para a realização de alguma ação.

Entretanto, nem todos os *sites* existentes fornecem API, 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 implementa um SAP, nomeada como Lana, assim como o sistema de extração de dados*,* para realizar os serviços disponibilizados, e dois *bots,* que utilizam API de *softwares* 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 PA foi implementado APIs para ambos.

A Lana funciona 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 SAP 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, foram implementados nos extratores de dados.

Além disso, a implementação da Lana foi realizada de maneira que gerou um produto genérico, isto é, somente os extratores de dados possuem funções especificas, a assistente trata todas as mensagens recebidas de maneira única, desta forma, a implementação de novos serviços, dos mais variados contextos, não acarretará na necessidade de modificar o SAP, somente à criação de novos extratores de dados e na integração de contexto.

Para a implementação do SAP foi utilizado o serviço Watson Assistant da plataforma IBM Bluemix, a solução BaaS Back4App, o *framework* Selenium para a técnica de *web scraping*, os serviços de hospedagem do Heroku e DigitalOcean e as linguagens de programação JavaScript e Python.

Este trabalho de conclusão de curso foi dividido em 4 tópicos gerais, apresentando-se no primeiro um aprofundamento teórico a respeito das plataformas e serviços utilizados. No segundo tópico é abordado a implementação de cada módulo do projeto, como e quais ferramentas foram utilizadas para o desenvolvimento de cada parte do projeto. O terceiro tópico caracteriza a exposição dos resultados obtidos e uma discussão sobre eles, além de exibir possíveis melhorias e os impedimentos sofridos durante o projeto. Por fim, no ultimo tópico é apresentado as conclusões acerca dos resultados obtidos e proposto implementações futuras.

# EMBASAMENTO TEÓRICO

## IBM Bluemix

### IBM Watson

#### Watson Assistant

## Aplicativos de Troca de Mensagens

### Telegram

### Messenger

## Back4App

## VPS Hosting

### Heroku

### DigitalOcean

## Selenium

## IDL

# MATERIAIS E MÉTODOS

## Interface

### Telegram

### Messenger

## Banco de Dados

## Entendimento de Linguagem Natural

## Fachada

## Assistente Pessoal

## Indexador de Bots

## Bots Scrapers

### Bot Sagres

### Bot UESC

## Correções e Otimizações

# RESULTADOS E DISCUSSÕES

# CONCLUSÃO