CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Samuel Junior Germano Pimentel

CALENDÁRIO ULBRA: *SKILL* DO CALENDÁRIO ACADÊMICO DO CEULP/ULBRA PARA ALEXA

Palmas – TO

2021

Samuel Junior Germano Pimentel

CALENDÁRIO ULBRA: *SKILL* DO CALENDÁRIO ACADÊMICO DO CEULP/ULBRA PARA ALEXA

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo

Palmas – TO

2021

Samuel Junior Germano Pimentel

CALENDÁRIO ULBRA: *SKILL* DO CALENDÁRIO ACADÊMICO DO CEULP ULBRA PARA ALEXA

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Esp. Fábio Castro Araújo

Orientador

Centro Universitário Luterano de Palmas - CEULP

_____

Prof. Me. Fabiano Fagundes

Centro Universitário Luterano de Palmas - CEULP

_____

Prof. Me. Jackson Gomes de Souza

Centro Universitário Luterano de Palmas - CEULP

Palmas – TO

2021

RESUMO

PIMENTEL, Samuel Junior Germano. **CALENDÁRIO ULBRA: *skill* do calendário acadêmico do CEULP/ULBRA para Alexa**. 2021. 40 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2020.

O presente trabalho teve por objetivo desenvolver uma *skill* do calendário acadêmico do CEULP/ULBRA para Alexa, a assistente de voz virtual da Amazon. Para tanto, foram apresentados conceitos referentes a roteiros, *storyboards*, modelos de interação de voz e uma contextualização de aprendizagem de máquina na Alexa e processamento de linguagem natural na Alexa. Assistentes de voz, também conhecidos como assistentes virtuais, são programas que agem com características humanas utilizando interfaces de voz, fazendo o uso da linguagem natural para ajudar os usuários em tarefas do cotidiano. Para criação da *skill* foi necessário realizar um tratamento dos dados do calendário em PDF para arquivo em formato de Javascript Object Notation (JSON), em seguida a criação dos roteiros para simularem as conversas do usuário com a skill, e a conversão desses roteiros em storyboards, que servem para demonstrar fluxos alternativos das conversas e as viradas de conversas. Por fim, foi feito o cadastro das informações do calendário por meio da integração com o Google Calendar e o desenvolvimento da skill. Como resultado final, foi entregue uma skill do calendário acadêmico do CEULP/ULBRA para Alexa, possibilitando fazer a consulta do calendário por meio de comando de voz na Alexa.

Palavras-chave: Alexa; Node.js; *Skill*.

LISTA DE FIGURAS

LISTA DE TABELAS

Tabela 2: *roteiro* calendário Ulbra (caminho alternativo 1).

Tabela 3: *roteiro* calendário Ulbra (ajuda).

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

AWS – *Amazon Web Services*

ASK – *Alexa Skills Kit*

AVS – *Alexa Voice Service*

CEULP – Centro Universitário Luterano de Palmas

HTTP – *Hypertext Transfer Protocol*

GUI – *Graphics User Interface*

IUV – *Voice User Interface*

JSON – *Javascript Object Notation*

PDF – *Portable Document Format*

PLN – Processamento de Linguagem Natural

OCR – Reconhecimento Óptico de Caracteres

REPL - *Read-Eval-Print-Loop*

SDK – *Software Development Kit*

ULBRA – Universidade Luterana do Brasil

SUMÁRIO

1 INTRODUÇÃO

O avanço da tecnologia tem sido em um ritmo acelerado e está cada vez mais comum no dia a dia o uso de auto falantes inteligentes. Uma das áreas que vêm ganhando atenção é o uso de Assistentes Pessoais Inteligentes. Segundo Moderno (2021) cerca de 8 bilhões de assistentes de voz estarão ativas no mundo em 2024. Este número é o triplo de 2018, estima a Juniper Research (2020). O uso diário da voz para realizar tarefas e ditar comandos aumentou 47% entre os brasileiros durante a pandemia de Covid-19 (MODERNO, 2021).

O conceito de um software que fornece assistência aos usuários não é novo e já vem sendo trabalhado em diversas frentes como por exemplo em celulares e em casas automatizadas. Estes softwares objetivam auxiliar ou substituir seus usuários em certas tarefas, liberando-os para atividades mais importantes (ZAMBIASI, 2012).

Segundo Huhns e Singh (1998), um software assistente de voz virtual não pode ser visto apenas como um programa de computador personalizado. Isto porque um assistente de voz deve ser baseado em rede, ser interativo, adaptativo, de propósito geral, de execução autônoma e que possa interagir com outros assistentes pessoais.

Os assistentes de voz interagem, muitas vezes, de forma descontraída e divertida, ou ainda como um "amigo" que conhece muito bem os hábitos do seu usuário. Isso ocorre porque a Inteligência Artificial (AI), resumidamente, é a possibilidade de uma máquina, através de algoritmos, possuir capacidade cognitiva semelhante ao de um ser humano (SILVA, 2019).

A Amazon Alexa, por exemplo, é um serviço de assistente de voz inteligente na nuvem onde pode-se solicitar tarefas como realizar pesquisas, mandar executar uma lista de músicas ou questionar o horário atual (VIGLIAROLO, 2017). A Alexa permite conectar em dispositivos que tenham um *chip*, uma rede *wi-fi* e um microfone e qualquer dispositivo pode se tornar habilitado para a Alexa.

Gradualmente, a pandemia do coronavírus aproximou cada vez mais pessoas e tecnologias. Sem contar o aumento considerável da utilização de computadores e *smartphones* decorrente dos trabalhos remotos. A quarentena no Brasil registrou um crescimento de 47% do uso de serviços ou produtos com assistentes virtuais por voz. Shimabukuro (2021)

A Alexa, da Amazon, é um exemplo de como esse novo modo de se comunicar é efetivo. Com a utilização de assistentes virtuais, alguns processos comuns e frequentes no dia a dia são automatizados.

Atualmente, as informações sobre o calendário acadêmico do CEULP/ULBRA são disponibilizadas aos alunos e professores semestralmente no portal acadêmico da instituição na página “Espaço acadêmico”. As informações são disponibilizadas por meio de um *Portable*

Document Format (PDF) que é dividido pelos meses e marcado com as informações de datas comemorativas e datas importantes do semestre como atividades e provas.

Existe uma página *web* com o calendário acadêmico do CEULP/ULBRA, mas não existe um outro canal que disponha acesso a essas informações. O desenvolvimento de uma *Skill* da Alexa abrirá uma nova possibilidade de se consultar o calendário. Podendo assim aumentar a chance de disseminação da informação. Com o uso da *skill* se tornará mais prático a consulta ao calendário.

Dito isto, este trabalho originou-se do problema em desenvolver uma *skill* do calendário acadêmico do CEULP/ULBRA utilizando o SDK da Alexa para Node.js e que permita a interação do usuário com uma API das informações do calendário acadêmico. Para resolver este problema, o primeiro passo para o seu desenvolvimento compreendeu o planejamento e desenvolvimento da *skill*. Em seguida foi realizada a construção do modelo de interação de voz do usuário e a *skill*. Feito isto, foi desenvolvido a API em Node.js para fornecer os dados do calendário acadêmico. Por fim, foi feita a integração do ASK SDK da Alexa com o Node.js.

Diante disto, conclui-se que com a utilização do ASK SDK para Node.js da Alexa e o desenvolvimento do modelo de interação de voz personalizado do usuário com a *skill* possibilitou a construção da skill capaz de consultar o calendário acadêmico e retornar os eventos disponíveis para consulta referentes a cada mês.

O conteúdo do presente trabalho está estruturado da seguinte maneira: na segunda seção foi discutido sobre o Referencial Teórico, onde foi disponibilizado informações sobre os conceitos e trabalhos acadêmicos que embasam este trabalho. Na terceira seção foi abordada a metodologia aplicada no desenvolvimento do trabalho. Na quarta seção foi apresentado os resultados obtidos e discussões em torno destes resultados. Por fim, na quinta seção foram expostos as considerações finais e trabalhos futuros.

1 REFERENCIAL TEÓRICO

Nesta seção serão abordados os conceitos relacionados ao funcionamento da Alexa e de uma *skill*.

2.1 FUNCIONAMENTO DA ALEXA

A Alexa é a assistente de voz virtual da Amazon, onde é possível criar experiências de voz naturais que oferecem aos usuários uma maneira mais intuitiva de interagir com a

tecnologia utilizadas diariamente (ALEXA, 2021). Os usuários interagem com a Alexa por meio aplicativo ou *skills* que fornecem um novo canal para seu conteúdo ou serviço. As *skills* permitem que os usuários utilizem comandos de voz para realizar tarefas cotidianas, como verificar notícias, ouvir músicas ou realizar alertas, por exemplo.

O sistema Alexa permite a criação *skills* para a mesma, sendo chamado de *Alexa Skills Kit* (ASK). O ASK é uma ferramenta de desenvolvimento de software que permite a criação de conteúdos, utilizando aplicações Processamento de Linguagem Natural (PLN) já existentes nos serviços da Alexa para a realização de análises e representações da linguagem humana.

Cada *skill* tem um modelo de interação de voz que define as palavras e frases que os usuários podem dizer à Alexa, realizando a ação desejada. A Alexa suporta dois tipos de modelos de interação:

- **Modelo de interação de voz pré-construído:** a Alexa define o conjunto de expressões para cada tipo de *skill*;
- **Modelo de interação de voz personalizado:** é preciso definir as frases ou declarações que os usuários podem dizer para interagir com a *skill*.

Um modelo de interação de voz pré-construído oferece um conjunto de expressões pré-definidas. Quando se usa o modelo de interação de voz pré-construído, o *Alexa Skills Kit*, define os enunciados e solicitações, chamados de *intents*, sendo preciso apenas codificar a *skill* para responder aos *intents* pré-definidos (ALEXA, 2021).

O modelo de interação de voz personalizado oferece maior controle sobre a experiência do usuário. É necessário projetar um conjunto de palavras e frases para cada ação que a *skill* pode realizar para fornecer uma experiência de voz totalmente personalizada. *Skills* personalizadas oferecem maior flexibilidade e controle sobre o projeto e o código (ALEXA, 2021).

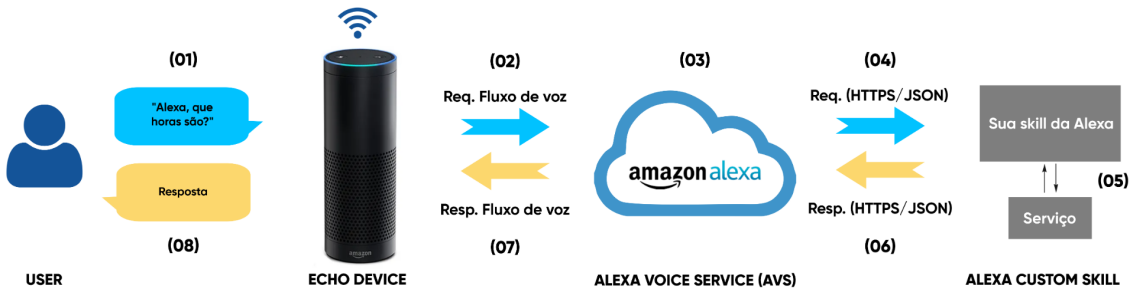
Com o modelo de interação de voz será possível desenvolver *skills* que recebem solicitações de voz, e processam a solicitação nos serviços de nuvem da Amazon. Todas as habilidades fazem o uso de linguagem natural para se adaptar às diversas maneiras como os usuários se expressam com a fala.

2.1.1. PROCESSAMENTO DE LINGUAGEM NATURAL NA ALEXA

Na Alexa, o usuário por meio de um comando de voz dando uma instrução para a mesma, a instrução é enviada para os serviços de nuvem da Alexa onde é feito o processamento e reconhecimento do comando dado para a *skill*, para realizar este processamento é utilizado o

Processamento de Linguagem Natural. Segundo Silva (2008, p. 2), o PLN constitui o núcleo das tecnologias de linguísticas e um dos novos paradigmas da Língua e da Linguística. O PLN reconhece as palavras que aparecem na frase e realiza um processo de análise que permite a extração de informações e o reconhecimento do contexto, atribuindo significado a cada palavra de acordo com o domínio. O PLN é responsável por analisar a estrutura da linguagem natural, como semântica, léxico e sintaxe, e por analisar e reconhecer o contexto das palavras no texto, resolvendo problemas de ambiguidade. A Figura 1 apresenta o fluxo do processamento de uma *skill* Alexa.

Figura 1: Fluxo de processamento .



Fonte: Alexa (2021) (tradução nossa)

Na Figura 1 pode-se observar o fluxo do usuário com a *skill*. Inicialmente o usuário realiza o comando de ativação "Alexa" ao dispositivo Echo, a fim de despertá-la, este processo é chamado de *wake word*, logo após o comando de ativação, o usuário expressa sua intenção (por exemplo: "Alexa, que horas são?") que, então, é transmitida para a *Alexa Voice Service* (AVS), onde é feito o processamento e interpretação da intenção do usuário, após isto, é realizada uma requisição *Hypertext Transfer Protocol* (HTTP) enviando os dados em formato Javascript Object Notation (JSON) para o serviço da *skill* escolhida e, por fim, retornando os dados em formato JSON para nuvem que retorna a resposta como um fluxo de voz para o dispositivo Echo.

2.1.2 APRENDIZADO DE MÁQUINA NA ALEXA

Segundo Monard (2003), o Aprendizado de Máquina (AM) é uma área da Inteligência Artificial (IA) cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como, a construção de sistemas capazes de adquirir conhecimento de forma automática. Um sistema de aprendizado é um programa de computador que toma decisões baseado em experiências acumuladas através da solução bem-sucedida de problemas anteriores.

A princípio, segundo Prati (2006), ainda não se conhece um algoritmo implementado por métodos tradicionais de programação que seja capaz de reconhecer, por exemplo, caracteres escritos à mão, quando também se adotavam métodos baseados em Reconhecimento Óptico de Caracteres (OCR, na sigla em inglês), com a utilização de AM, mas atualmente é possível utilizar outras técnicas. No entanto, utilizando técnicas de Aprendizagem de Máquina é "possível projetar um sistema computacional que aprenda a reconhecer caracteres escritos à mão através da observação de uma grande quantidade de manuscritos" COSTA (2020). De acordo com Rich (1991), entidades inteligentes destacam-se pela capacidade de adequar-se a novos ambientes e de resolver novos problemas. Um computador pode ser orientado a interpretar as informações recebidas de uma forma que melhore gradualmente seu desempenho.

Na Alexa, a aprendizagem de máquina não é diferente, a mesma foi projetada para ficar mais inteligente a cada dia, por meio da aprendizagem de máquina na nuvem. Quanto mais o cliente usa a Alexa, mais ela se adapta aos padrões de fala do usuário, vocabulário, preferências pessoais, etc. Quando a palavra de despertar (*wake word*) é dita, todas as próximas frases são processadas e armazenadas para responder à intenção do usuário, treinando o reconhecimento de fala e compreensão da linguagem natural. Assim, a Alexa está continuamente se aprimorando com cada entrada.

2.2 CONSTRUINDO UMA SKILL ALEXA

O surgimento de *Voice User Interface* (VUI, Interface de Voz de Usuário, em português), não é uma melhoria incremental da tecnologia existente; isso marca uma grande mudança na interação humano-computador. Como tal, projetar *skills* para VUIs é diferente de projetar aplicativos para *Graphics User Interface* (GUI, Interface Gráfica de Usuário, em português). Em vez de projetar VUIs como substitutos do teclado, mouse ou controles de toque, é preciso mudar toda a abordagem de design para criar interações verdadeiramente conversacionais e que priorizam a voz. Para ser eficaz, é necessário projetar VUIs para se adaptar às muitas maneiras como os usuários podem expressar significado e intenção por meio da fala (ALEXA, 2021).

Para se ter uma boa experiência e estabelecer uma confiança com o usuário é necessário projetar a interface de voz pensando sempre nos mais diversos cenários.

2.2.1 ROTEIROS

Segundo Esposito (2018), os roteiros mostram a conversa entre o usuário e Alexa, como em um filme ou peça, o roteiro determina como a conversa fluirá. Para escrever os roteiros, é necessário estar essencialmente preparando o roteiro de uma conversa, que pode ir em quase todas as direções. Pode ser difícil prever todas as direções possíveis que uma conversa pode tomar, então é necessário começar com o caminho mais fácil e em seguida adicionar variações a esse caminho, tendo assim mais possibilidades, criando uma experiência de usuário agradável. No diálogo fictício a seguir, é mostrado o roteiro da *skill* "Calendário Ulbra".

> **Usuário: –** Alexa, abra o "Calendário Ulbra".
> **Alexa: –** Olá, seja bem-vindo ao Calendário Ulbra! Posso fornecer informações das atividades registradas no calendário acadêmico, dos eventos mais próximos ou de uma data específica. Se precisar, fale "Ajuda" a qualquer momento. Gostaria de saber algo agora?
> **Usuário: –** Sim!
> **Alexa: –** Certo, me diga uma data ou evento para um dia específico.
> **Usuário: –** Me diga os eventos para o dia 21 de outubro.
> **Alexa: –** No dia 21 de outubro você tem avaliação AP2 e entrega da última Web Atividade. Algo mais que posso ajudar?

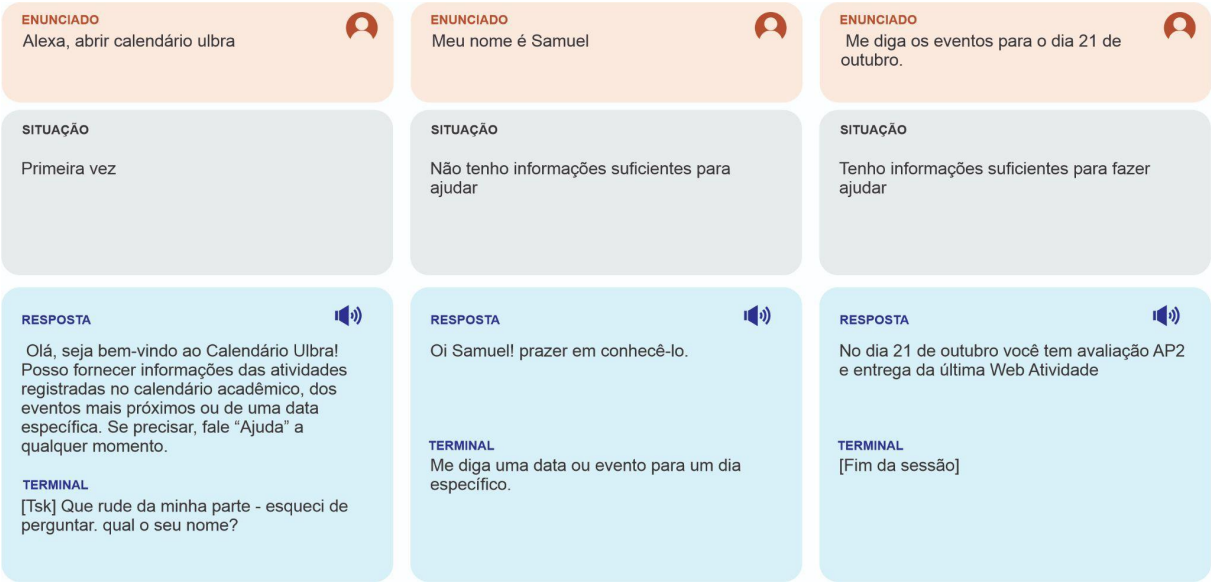
No diálogo apresentado é mostrado o roteiro da *skill* Calendário Ulbra. Os roteiros para uma *skill* Alexa são escritos no padrão de texto de uma conversação entre o utilizador e a *skill* de forma a se ter a maneira como o utilizador conversaria com a *skill.* Geralmente produzidos no formato .docx e organizados em diálogos. Após a definição, os roteiros são transformados em *storyboards*.

2.2.2 STORYBOARDS

O *design* de interfaces de voz se dá por meio de artefatos de voz. E um desses artefatos com mais relevância são as *storyboards*, que servem para planejar o progresso das conversas ao longo do tempo. Um *storyboard* é uma progressão linear ao longo do tempo. É uma boa maneira de simular as viradas em uma conversa. Os primeiros *storyboards* devem ser a maneira ideal de percorrer uma *skill* (ALEXA, 2021). As *storyboard*, segundo Preece (2013), são uma representação de sequências de ações ou eventos pelos quais o usuário e o produto devem passar para executar uma ação. Preece (2013) afirma ainda que há dois propósitos em usar-se de *storyboards*: (i) para obter *feedback* de usuários e (ii) para que a equipe de *design* considere o

cenário e a utilização do produto em maiores detalhes. Na figura 3 é apresentado o fluxo de uma *storyboard* para desenvolvimento de uma *skill* para Alexa.

Figura 2: Fluxo de uma *storyboard*.



Os *storyboards* para uma *skill* Alexa são criados na forma de *cards*, os *cards* são criados se baseando no roteiro escrito, simulando assim as viradas de conversas. A Figura 2 mostra os enunciados de situações e possíveis respostas que a Alexa pode vir a responder. Com o tempo vão se adicionando variações nesses enunciados, após criado o melhor caminho a ser seguido. Cada *storyboard* pode ajudar a encontrar novas variações no *design* construído. Quanto mais robusto for o projeto, melhor terão que ser analisadas as situações, para que o usuário tenha uma boa experiência.

3 METODOLOGIA

Nesta seção são apresentados os materiais e métodos utilizados no processo do desenvolvimento da *skill* para Alexa.

3.1 MATERIAIS

- **Node.js:** O Node.js é uma plataforma de software que é baseada no tempo de execução V8 Javaroteiro do Chrome para construir uma rede escalável sem esforço. Utiliza-se de um modelo de E/S não bloqueante e orientado por eventos o que a torna leve e eficiente, perfeita para dados intensivos em tempo real. O Node.js, inicialmente desenvolvido por Ryan Dahl, também fornece uma linguagem de programação *shell*, *Read-Eval-Print-Loop* (REPL) (Node.js, 2021). O Node.js será utilizado para o desenvolvimento da *skill* e da *Application Programming Interface* (API) que fornecerá os dados para a *skill*, juntamente com o ASK *Software Development Kit* (SDK) da Alexa.

- **AWS Lambda:** O *Amazon Web Services* (AWS) Lambda é um serviço da AWS que permite executar código sem provisionar ou gerenciar servidores. O AWS Lambda executa o código apenas quando necessário e escala automaticamente, de algumas requisições por dia a centenas por segundo. Com o AWS Lambda, pode-se executar códigos para virtualizar qualquer tipo de aplicação ou serviço back-end (AWS Lambda, 2021). Será carregado o código da *skill* para uma função Lambda e o Lambda irá executar as resposta às interações de voz da Alexa.

- **ASK SDK Alexa:** Os SDKs da ASK são ferramentas e bibliotecas de desenvolvimento de software que fornecem acesso programático aos recursos da Alexa. Os SDKs ASK estão disponíveis para Node.js, Java e Python. Os SDKs ASK podem ajudar a tornar mais fácil desenvolver habilidades Alexa, permitindo que você gaste mais tempo na implementação de recursos e menos tempo escrevendo código clichê (Alexa, 2021). O SDK servirá para utilizar os serviços já existentes da Alexa, facilitando nosso desenvolvimento.

- **Postman:** Segundo o site oficial, o Postman é uma ferramenta que dá suporte à documentação das requisições feitas pela API. Ele possui ambiente para a documentação, execução de testes de APIs e requisições em geral. Ao utilizá-lo, você

passará a trabalhar com APIs de modo mais eficiente, construindo solicitações rapidamente e, ainda, poderá guardá-las para uso posterior, além de conseguir analisar as respostas enviadas pela API.

3.2 MÉTODOS

Nesta seção são apresentados as etapas que foram executadas no desenvolvimento da *skill* do calendário acadêmico do CEULP/ULBRA para Alexa, conforme ilustrado na Figura 03.

Figura 3: Etapas para a implantação da *skill*.

A etapa de escrita do roteiro de interação de voz do usuário com a *skill*, consiste na construção do modelo de interação. O roteiro serviu de base para descobrir os inúmeros possíveis caminhos que o usuário deve seguir para interagir com a *skill* e serviu para garantir os possíveis erros ou casos a serem considerados nos enunciados a serem ditos.

Na segunda etapa de conversão dos roteiros em *storyboards*: foi feita a conversão dos roteiros que demonstram os possíveis caminhos em *storyboards*. Os primeiros *storyboards* mostram o caminho mais comum de percorrer a *skill,* em seguida é adicionado variações do mesmo enunciado no caminho e testado novamente para se encontrar falhas no design de voz.

Na terceira etapa de criação da *skill*, foi feita a integração com a API do google calendar e criado os métodos de cadastro das informações que foram retiradas do calendário disponível no portal do Aluno CEULP/ULBRA

Na etapa de definição do modelo de interação, depois de feito o diálogo e o fluxo, foi preciso conectá-los a fim de desenvolver a *skill,* os enunciados são convertidos em *intents*, que representam o que o usuário fez na *skill.* É necessário criar possíveis variações para uma mesma intenção, visto que nem todo usuário utiliza a mesma ação para executar a mesma intenção.

Após definido o modelo de interação, foi feita a integração do ASK SDK da Alexa com a linguagem Node.js que foi a utilizada na criação da *skill*. Essa integração tem por objetivo poder utilizar os serviços da Alexa, fazendo chamadas no serviços em nuvem da Alexa.

Por fim, na etapa de desenvolvimento da *skill*, foi desenvolvido a regra de negócios com base no modelo de interação construído, utilizando o ASK SDK da Alexa. Após o desenvolvimento foi realizado o deploy da *skill* e transformada em uma função lambda disponível na AWS para que seja possível realizar os testes no funcionamento.

4 RESULTADOS E DISCUSSÃO

Nesta seção, foram descritos os resultados e como se desenvolveu a execução de cada etapa deste trabalho que busca desenvolver uma *skill* do calendário acadêmico do CEULP/ULBRA para a assistente virtual Alexa.

4.1 ESCRITA DO ROTEIRO

Para o desenvolvimento da *skill* foi necessária a criação do roteiro. Para essa finalidade, foram definidas como seriam feitas as interações do usuário com a *skill,* no intuito de facilitar o desenvolvimento e servir como base para criação do modelo de interação.

Tabela 1: *roteiro* calendário Ulbra.

Usuário	Alexa
Alexa, abra o "Calendário Ulbra"	Olá, seja bem-vindo ao Calendário Ulbra! Posso fornecer informações das atividades registradas no calendário acadêmico do CEULP/ULBRA. Você pode fazer uma solicitação dos seus eventos para amanhã ou de uma data específica. Se precisar, fale "Ajuda" a qualquer momento.
Me diga os eventos para o dia 21 de outubro	No dia 21 de outubro você tem avaliação AP2 e entrega da última Web Atividade.
Me diga os eventos para amanhã	No dia 22 você tem a entrega final do TCC 2.

Na tabela 1 é mostrado o *roteiro* de fluxo 01, onde o usuário executa uma ação para solicitar a abertura da *skill*, logo em seguida, a *skill* retorna uma resposta de voz com a mensagem de boas vindas e algumas instruções para o usuário de como prosseguir. Após feita a solicitação é retornada uma mensagem de voz com a resposta para o usuário.

Tabela 2: *roteiro* calendário ulbra (caminho alternativo 1).

Usuário	Alexa
Alexa, abrir Calendário Ulbra	Olá, seja bem-vindo ao Calendário Ulbra! Posso fornecer informações das atividades registradas no calendário acadêmico do CEULP/ULBRA. Você pode fazer uma solicitação dos seus eventos para amanhã ou de uma data específica. Se precisar, fale "Ajuda" a

	qualquer momento.
Pular entrada	Desculpe, não entendi. Poderia repetir?
Alexa, sair	Até mais!

Diferente da tabela 1, na tabela 2 o usuário, ao invés de seguir o fluxo por completo, tenta pular a mensagem de boas vindas, mas acaba não tendo resposta, pois a *skill* não vai conseguir obter a resposta sem a frase solicitada existir no modelo de interação. Sem uma resposta adequada, o usuário pede para fechar a aplicação. Esse fluxo serviu para demonstrar uma outra maneira do usuário usar a *skill* e também analisar se o texto de boas vindas é muito extenso.

Tabela 3: *roteiro* calendário Ulbra (ajuda).

Usuário	Alexa
Alexa, abrir calendário ulbra	Olá, seja bem-vindo ao Calendário Ulbra! Posso fornecer informações das atividades registradas no calendário acadêmico do CEULP/ULBRA. Você pode fazer uma solicitação dos seus eventos para amanhã ou de uma data específica. Se precisar, fale “Ajuda” a qualquer momento.
Me diga as provas para o dia de amanhã	Desculpe, não entendi. Poderia repetir?
Ajuda	Para fazer uma solicitação dos seus eventos para amanhã, diga: "Meus eventos para amanhã" ou de uma data específica, se precisar fale “Ajuda” a qualquer momento.
Me diga os eventos para o dia 21 de outubro	No dia 21 de outubro você tem avaliação AP2 e entrega da última Web Atividade.

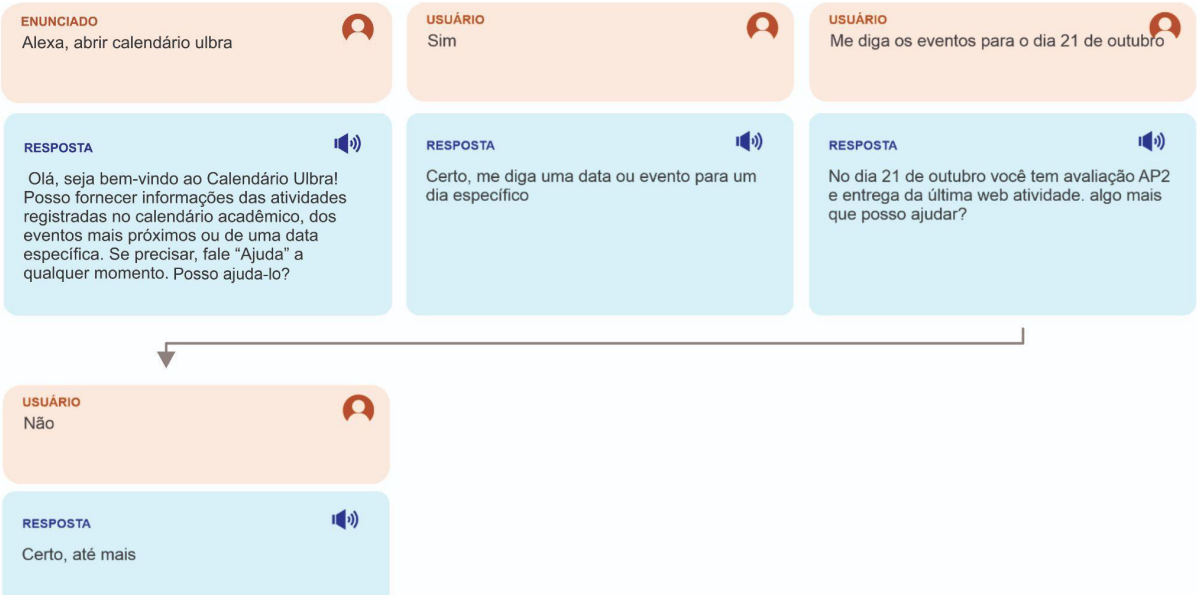
O *roteiro* de ajuda mostrado na tabela 3 serve para demonstrar o fluxo de auxílio ao usuário, caso ele não tenha entendido como utilizar a *skill* e seu funcionamento.

Os *roteiros* apresentados foram criados utilizando a ferramenta de textos word, sendo definidos três caminhos que o usuário pode seguir. Após a criação dos caminhos possíveis, foram criadas as *storyboards*.

4.2 CONVERSÃO DOS ROTEIROS EM STORYBOARDS

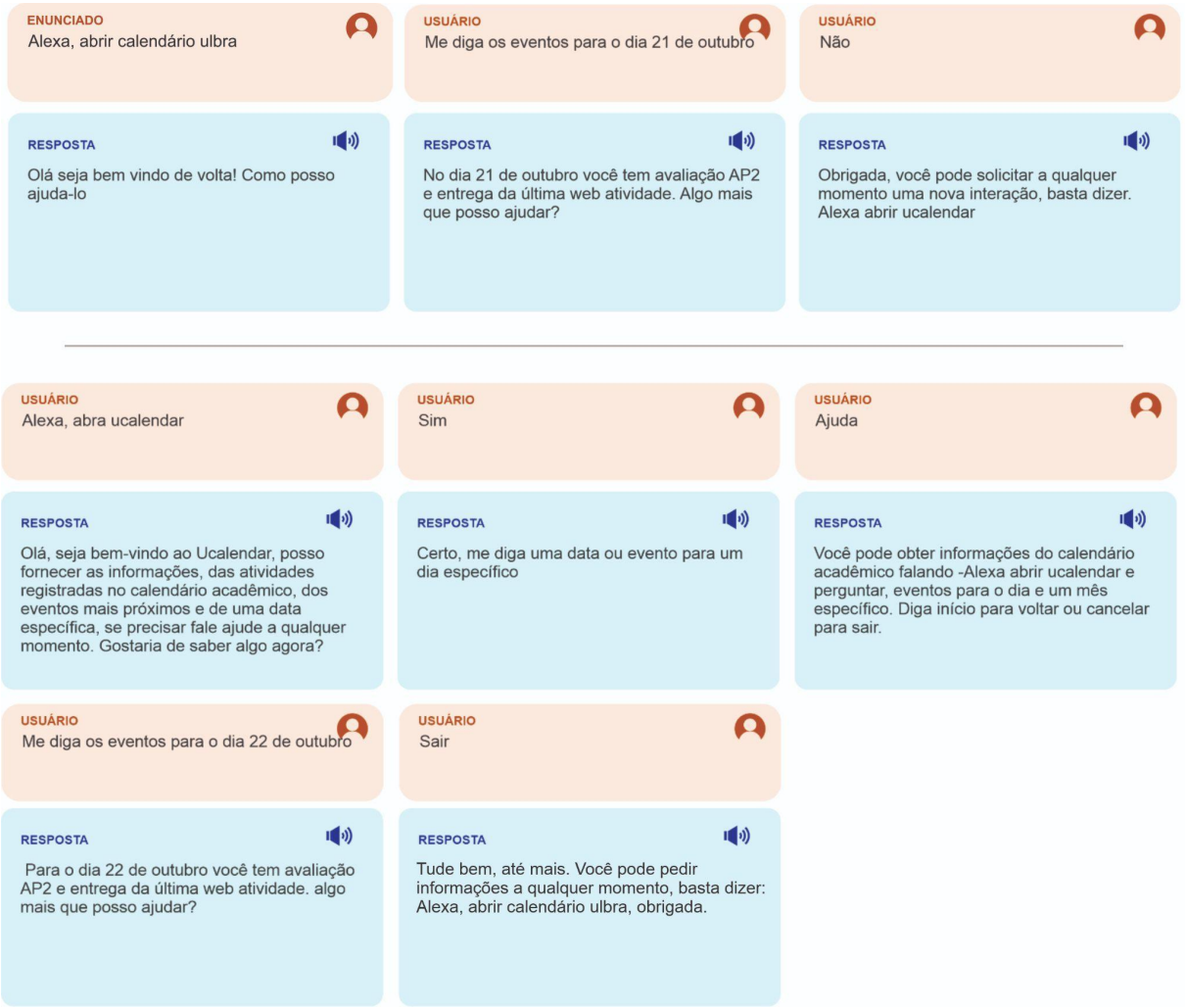
As *storyboards* foram criadas após a definição dos *roteiros*. Elas mostram a ideia dos caminhos e diferentes maneiras do usuário interagir com a *skill.* Foram criadas variações em cada cenário a fim de se obter melhores resultados e definições.

Figura 4: *storyboard* fluxo 1.



A Figura 04 mostra o caminho mais comum, por ser o mais simples de se chegar seguindo as falas da *skill*. O usuário faz a solicitação para que seja ativada a *skill,* após a solicitação a Alexa retorna com uma mensagem de boas vindas, em sequência, ela pergunta como pode ajudar o usuário. A resposta do usuário é a solicitação de um evento registrado no dia especificado.

Figura 5: *storyboard* fluxo 2.



Na figura 05 é realizada uma segunda interação onde mostra o caminho mais curto, após o usuário já ter utilizado a *skill* por inúmeras vezes e, já estando mais habituado, ele foi direto para o que ele queria, sem a necessidade de tantas perguntas e obtendo o mesmo resultado do primeiro ciclo de interação.

4.4 CRIAÇÃO DA API PARA GERENCIAR AS INFORMAÇÕES DO CALENDÁRIO

Após a definição dos *roteiros*, a criação das *storyboards* e definição do modelo de interação, foi desenvolvida uma API para gerenciar as informações a serem transmitidas para os usuários da *skill.* As ferramentas utilizadas para o desenvolvimento foram o Node.js, framework express.js e API do Google Calendar.

Foi necessário realizar uma integração com o Google Calendar para se armazenar as informações referentes ao calendário, visto que não é possível armazenar um arquivo na *skill.* Essa integração vai permitir com que o usuário interaja com as informações da *skill.* Para se

fazer a integração com a API do Google Calendar foi necessária a criação das chaves no Google Cloud e, após isso, foi feita a integração.

Figura 6: Integração com a API do Google Calendar.

```
1  const { OAuth2 } = google.auth
2  const oAuth2Client = new OAuth2('290613365444-vsn1qes9pest7fsutjco234iu5fg
   7qua.apps.googleusercontent.com', 'cCvNu8lMqABNVQqWky9Pbj7s')
3  oAuth2Client.setCredentials({ refresh_token: '1//04RlByIHKlO2cCgYIARAAGAQS
   NwF-L9IrZK7EVLWJB-961E6lYexUCgSiDJHUEWjvCQpkkb4MJCrYfwjvponV2oBQ4PnDVgMIje
   w' })
4  const calendar = google.calendar({ version: 'v3', auth: oAuth2Client })
5
```

Para integração com o Google Calendar foi utilizado o OAuth2. O OAuth2 é um protocolo de autorização que permite que uma aplicação se autentique em outra. No trecho de código, o primeiro parâmetro da linha 2 da Figura 6 para autenticação é o `id_do_cliente` e o segundo é a `chave_secreta`, ambos foram gerados a partir do Google Cloud. Em seguida, foi necessária a geração de um `refresh_token` que autoriza a API a escrever na agenda do google. O `refresh_token` é gerado no OAuth 2.0 *playground*. Após a configuração necessária para a integração foi criada a funcionalidade `sendCalendar(event)` para cadastro dos eventos.

Figura 7: Função de cadastro de evento no google agenda.

```
1  app.post("/calendar", function(request, response){
2
3    const {summary, location, description, eventStartTime, eventEndTime} =
   request.body
4
5    const event = {
6      summary,
7      location,
8      description,
9      colorId: 1,
10     start: {
11       dateTime: new Date(eventStartTime),
12       timeZone: 'America/Sao_Paulo',
13     },
14     end: {
15       dateTime: new Date(eventEndTime),
16       timeZone: 'America/Sao_Paulo',
17     },
18   }
19
20   calendar.freebusy.query(
21     {
22       resource: {
23         timeMin: new Date(eventStartTime),
24         timeMax: new Date(eventEndTime),
25         timeZone: 'America/Sao_Paulo',
26         items: [{ id: 'primary' }],
27       },
28     },
29     (err, res) => {
30       if (err) return console.error('Erro de consulta, livre ou ocupado:
   ', err)
31       const eventArr = res.data.calendars.primary.busy
32       if (eventArr.length === 0)
33         return calendar.events.insert(
34           { calendarId: 'primary', resource: event },
35           err => {
36             if (err) return console.error('Error ao criar um evento no ca
   lendário', err)
37             return console.log('Evento criado com sucesso.')
38           }
39         )
40       return console.log(`Desculpe, ocupado ... `)
41     }
42   )
43
```

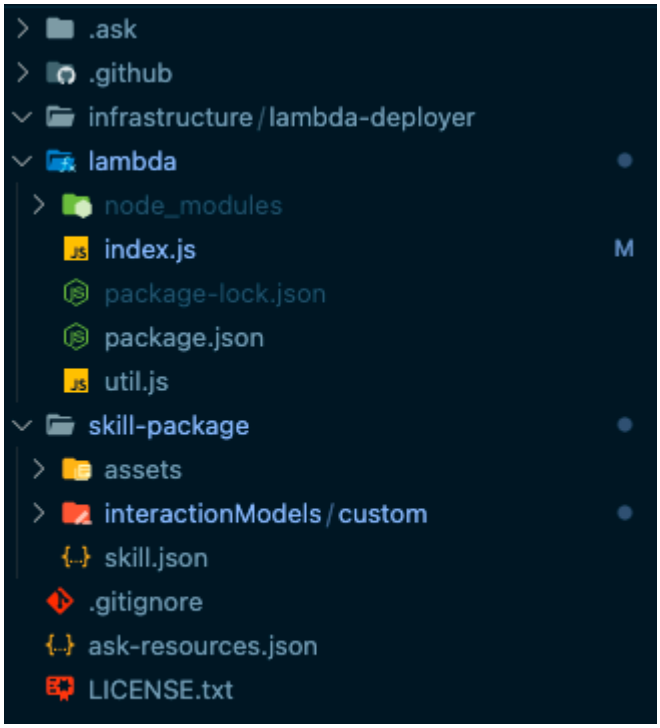
A figura 7 mostra a funcionalidade de cadastro de eventos. Para ser feito o cadastro é necessário passar os parâmetros `summary` que é título do evento, `location` que é a localização do evento e o parâmetro `description` contendo a descrição.Após passado os parâmetros é realizado o cadastro do evento. Caso tenha um evento cadastrado na mesma data, é retornada para o usuário uma mensagem de erro, que já tem um evento cadastrado nesta mesma data. Após a criação da funcionalidade, o cadastro dos eventos no ambiente de desenvolvimento foi

feito utilizando a ferramenta POSTMAN, que serve para realizar requisições HTTP sem a necessidade de criar um front-end (um site ou um portal). Depois de cadastrados iniciou-se a integração com o ASK SDK e o desenvolvimento da *skill* para Alexa

4.5 INTEGRAÇÃO DO ASK SDK DA ALEXA

Após realizada a criação da API, foi realizada a integração com SDK para ser possível a implementação da *skill.* Para dar início ao desenvolvimento da *skill* foi necessária a instalação do ASK CLI que auxilia na criação da estrutura do projeto. Para instalação do ASK CLI é executado no terminal o comando "`$npm install -g ask-cli`". Ao executar o comando é instalado globalmente o CLI que serve para inicializar o projeto. Após a instalação foi executado o comando "`$ask new`" que gera uma estrutura de pastas para iniciar a implementação com o node.js e funções lambda na AWS.

Figura 8: Estrutura de pastas da *skill*.



A estrutura de pastas a seguir é gerada após a execução do comando. A pasta "`./lambda/index.js`" contém a lógica de negócios da *skill*. A pasta "`./skill-package`" contém o modelo de interação e o arquivo skill.json onde se encontra as configurações referente a publicação da *skill.* Após iniciado o projeto este já é instalado nas dependências o ask-sdk.

4.3 DEFINIÇÃO DO MODELO DE INTERAÇÃO

Após feito a integração é dado início a criação do modelo de interação. O modelo de interação é a base com que o usuário irá interagir com a *skill* e também onde são definidas as *intents* que servem como funcionalidades dentro da *skill*. O modelo de interação é desenvolvido com base nos *roteiros* e *storyboards* e segue no formato JSON. Para se fazer um modelo de interação é seguido alguns padrões de intents, que são as intents integradas, elas ajudam na construção e evitam a repetição e criação de funcionalidades já existentes como por exemplo; cancelar, parar, pedir ajuda e etc.

Figura 9: Modelo de interação.

```
1  {
2      "interactionModel": {
3          "languageModel": {
4              "invocationName": "calendário ulbra",
5              "intents": [
6                  {
7                      "name": "AMAZON.CancelIntent",
8                      "samples": [
9                          "Sair",
10                         "Cancelar",
11                         "Cala a boca",
12                         "Adeus"
13                     ]
14                 },
15                 {
16                     "name": "AMAZON.HelpIntent",
17                     "samples": [
18                         "Ajuda",
19                         "Help",
20                         "Socorro"
21                     ]
22                 },
23                 {
24                     "name": "AMAZON.StopIntent",
25                     "samples": [
26                         "Parar",
27                         "Stop",
28                         "Calada"
29                     ]
30                 },
31                 {
32                     "name": "GetDateEventsIntent",
33                     "slots": [
34                         {
35                             "name": "date",
36                             "type": "AMAZON.DATE"
37                         }
38                     ],
39                     "samples": [
40                         "para hoje",
41                         "de hoje",
42                         "do dia {date}",
43                         "para o dia {date}",
44                         "diga-me meus eventos para {date}",
45                         "buscar meus eventos para {date}",
46                         "Me diga os eventos para {date}"
47                     ]
48                 },
49                 {
50                     "name": "AMAZON.NavigateHomeIntent",
51                     "samples": [
52                         "início",
53                         "home",
54                         "começo"
55                     ]
56                 }
57             ],
58             "types": []
59         }
60     }
61 }
```

A figura 9 mostra o arquivo do nosso modelo de interação. a *intents* `AMAZON.CancelIntent`, `AMAZON.HelpIntent`, `AMAZON.StopIntent` e `AMAZON.NavigateHomeIntent` são intent integradas que já vem por padrão na criação de nosso projeto. Na linha 4 do código foi definido o nome de invocação da *skill,* que serve para o usuário acessar a *skill* e realizar uma *intent.* Da linha 5 até a linha 57 do código é a definição das *intents*. Cada *intent* está definido com um nome e exemplo de como pode ser chamada. A intent criada foi a `GetDateEventsIntent` foram definidos exemplos com base no modelo de interação para auxiliar na chamada dela

4.6 DESENVOLVIMENTO DA SKILL

Após definido o modelo de interação foi feito o desenvolvimento da lógica de negócios da *skill.*

Figura 10: LaunchRequesthandler.

```
1  const Alexa = require('ask-sdk-core');
2  const Axios = require("axios");
3
4  const SKILL_NAME = "calendario ulbra";
5  const EVENTS_URL = "https://content.googleapis.com/calendar/v3/calendar
   s/primary/events";
6  const MONTHS = ['Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junh
   o', 'Julho', 'Agosto',
7      'Setembro', 'Outubro', 'Novembro', 'Dezembro'];
8
9  const LaunchRequestHandler = {
10     canHandle(handlerInput) {
11         return handlerInput.requestEnvelope.request.type === "LaunchReq
   uest";
12     },
13     handle(handlerInput) {
14         const speechText = "Olá, seja bem-vindo ao calendário ulbra, po
   sso fornecer as informações, das atividades registradas no calendário a
   cadêmico do CEULP/ULBRA. Você pode fazer uma solicitação dos seus event
   os para amanhã ou de uma data específica, se precisar fale ajuda a qual
   quer momento.";
15
16         return handlerInput.responseBuilder
17             .speak(speechText)
18             .reprompt(speechText)
19             .withSimpleCard(SKILL_NAME, speechText)
20             .getResponse();
21     }
22 };
```

Para dar início a programação é importado o ask-sdk-core da Alexa, que serve para utilizar as funcionalidades do Alexa *skills kit*. Após, é feita a importação do Axios para realizar as requisições HTTP no node.js. A variável `SKILL_NAME` define o nome da skill, e a variável `EVENTS_URL` é a url da API do Google Calendar onde serão buscadas as informações referentes aos eventos registrados no calendário. Na figura 10 também é apresentado a *intent* `LaunchResquestHandler` que retorna o texto de boas vindas ao usuário após a chamada da *skill* com a utilização da *wake word.*

Figura 11: GetDateEventsIntentHandler.

```
1  const GetDateEventsIntentHandler = {
2      canHandle(handlerInput) {
3          return (
4              handlerInput.requestEnvelope.request.type === "IntentRequest" &&
5              handlerInput.requestEnvelope.request.intent.name === "GetDateEventsIn
   tent"
6          );
7      },
8      async handle(handlerInput) {
9          const accessToken = handlerInput.requestEnvelope.context.System.user.acce
   ssToken;
10
11         let timeMin = new Date();
12         let timeMax = new Date();
13         let preamble = "Seus eventos para";
14
15         if (handlerInput.requestEnvelope.request.intent.slots.date &&
16             handlerInput.requestEnvelope.request.intent.slots.date.value) {
17             const date = handlerInput.requestEnvelope.request.intent.slots.date.v
   alue;
18             console.log("meu date———————————x——————————", date)
19             timeMin = new Date(Date.parse(date + 'T01:00'));
20             timeMax = new Date(Date.parse(date + 'T01:00'));
21
22             console.log("timeMax——————————————————————", timeMax)
23             preamble += MONTHS[timeMin.getMonth()] + " " + timeMin.getDate() + ":
   ";
24         } else {
25             preamble += "hoje: "
26         }
27
28         timeMax.setDate(timeMax.getDate() + 1);
29         timeMin.setHours(0, 0, 0, 0);
30         timeMax.setHours(23, 59, 59, 999);
31
32         console.log("timeMin: " + timeMin.toISOString());
33         console.log("timeMax: " + timeMax.toISOString());
34
35         const eventsSpeechText = await getEventsSpeech(accessToken, timeMin.toISO
   String(), timeMax.toISOString());
36
37         return handlerInput.responseBuilder
38             .speak(preamble + eventsSpeechText)
39             .withSimpleCard(SKILL_NAME, preamble + eventsSpeechText)
40             .getResponse();
41     }
42 };
```

Na figura 11 a `GetDateEventsIntentHandler` pega o *accessToken* do usuário e em seguida são definidas as variáveis que recebem os valores `timeMin` o tempo mínimo que é a data atual e o `timeMax` que é data que o usuário informa a *skill*. Em seguida é iniciada a função `getEventsSpeech` que recebe o *token* do usuário, o `timeMin` e o `timeMax` e realiza uma requisição na API do Google Calendar.

Figura 12: getEventsSpeech.

```
1  const getEventsSpeech = async (accessToken, timeMin, timeMax)
   => {
2      const noEvents = 'Não há eventos';
3      const url = EVENTS_URL + "?orderBy=updated&timeMin=" + tim
   eMin + "&timeMax=" + timeMax;
4      console.log("URL: " + url);
5
6      const eventsResponse = await getEvents(url, accessToken);
7
8      if (eventsResponse && eventsResponse.data) {
9          console.log("Get events OK: " + JSON.stringify(eventsR
   esponse.data));
10
11         const events = eventsResponse.data.items;
12
13         if (events.length > 0) {
14             let eventsSpeechText = "";
15             for (let i = 0; i < events.length; i++) {
16                 const event = events[i];
17                 eventsSpeechText += "Event " + (i + 1) + ": "
   + event.summary + ' - ' + event.description + '. ';
18                 eventsSpeechText += "Location: " + event.locat
   ion + ". From " + getHour(event.start.dateTime) +
19                     " to " + getHour(event.end.dateTime) + ".
   ";
20             }
21             return eventsSpeechText;
22         }
23     }
24     return noEvents;
25 };
```

Na figura 12 é mostrada a *intent* `getEventsSpeech`. Ela recebe como parâmetros o `accessToken` do usuário, o `timeMin` e o `timeMax` que é recebido como resultado do evento de voz da Alexa ao pedir uma dia específico ao usuário. É realizada uma requisição na API, se não tiver eventos na data especificada é retornada a mensagem de voz "não há eventos neste dia especificado", caso contrário é percorrida a lista de eventos retornando para o usuário a mensagem de voz com o eventos especificado, o nome do evento, o local e o horário.

Figura 13: HelpIntentHandler.

```
1   const HelpIntentHandler = {
2       canHandle(handlerInput) {
3           return Alexa.getRequestType(handlerInput.requestEnvel
    ope) === 'IntentRequest'
4               && Alexa.getIntentName(handlerInput.requestEnvelo
    pe) === 'AMAZON.HelpIntent';
5       },
6       handle(handlerInput) {
7           const speechText = "Para fazer uma solicitação dos se
    us eventos para amanhã, diga: 'Meus eventos para amanhã' ou d
    e uma data específica, se precisar fale ajuda a qualquer mome
    nto.";
8
9           return handlerInput.responseBuilder
10              .speak(speechText)
11              .reprompt(speechText)
12              .withSimpleCard(SKILL_NAME, speechText)
13              .getResponse();
14      }
15  };
```

Caso o usuário não consiga uma resposta ou não consiga utilizar a *skill* e solicitar ajuda é chamada a `HelpIntentHandler`. Na figura 13 é apresentado o código dessa *intent.* Ela retorna uma mensagem de voz para o usuário de como ele deve usar a *skill* e a opção para o usuário de continuar utilizando a *skill* ou sair.

Figura 14: CancelAndStopIntentHandler.

```
1  const CancelAndStopIntentHandler = {
2      canHandle(handlerInput) {
3          return (
4              handlerInput.requestEnvelope.request.type === "In
   tentRequest" &&
5              (handlerInput.requestEnvelope.request.intent.name
   === "AMAZON.CancelIntent" ||
6                  handlerInput.requestEnvelope.request.intent.n
   ame === "AMAZON.StopIntent")
7          );
8      },
9      handle(handlerInput) {
10         const speechText = "Até mais, xoxo!";
11
12         return handlerInput.responseBuilder
13             .speak(speechText)
14             .withSimpleCard(SKILL_NAME, speechText)
15             .getResponse();
16     }
17 };
```

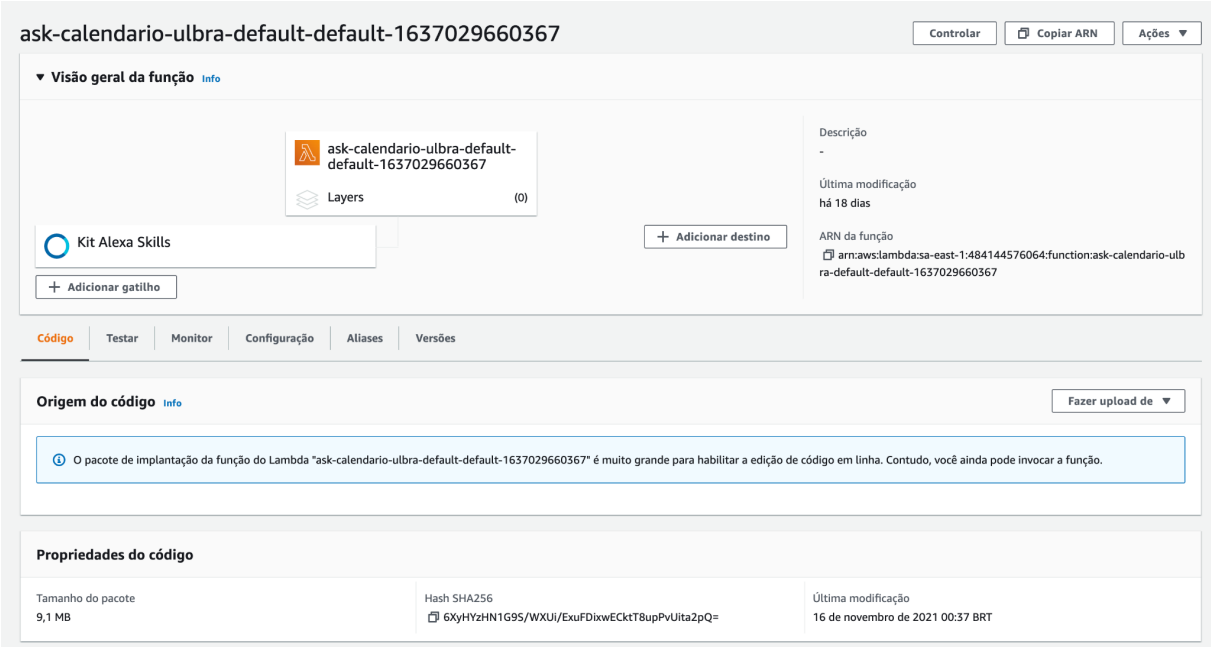
A figura 14 demonstra a intent `CancelAndStopIntentHandler`. Ao ser chamada são cancelados os eventos que foram solicitados e o usuário recebe a resposta de voz "Até mais!" e a sessão com a *skill* é encerrada. Ela pode ser chamada no meio de uma conversa e também após uma conversa no meio da sessão.

Figura 15: skillBuilder

```
1  const skillBuilder = Alexa.SkillBuilders.custom();
2
3  exports.handler = skillBuilder
4      .addRequestHandlers(
5          LaunchRequestHandler,
6          GetDateEventsIntentHandler,
7          HelpIntentHandler,
8          CancelAndStopIntentHandler,
9          SessionEndedRequestHandler,
10         IntentReflectorHandler
11     )
12     .addErrorHandlers(ErrorHandler)
13     .lambda();
14
```

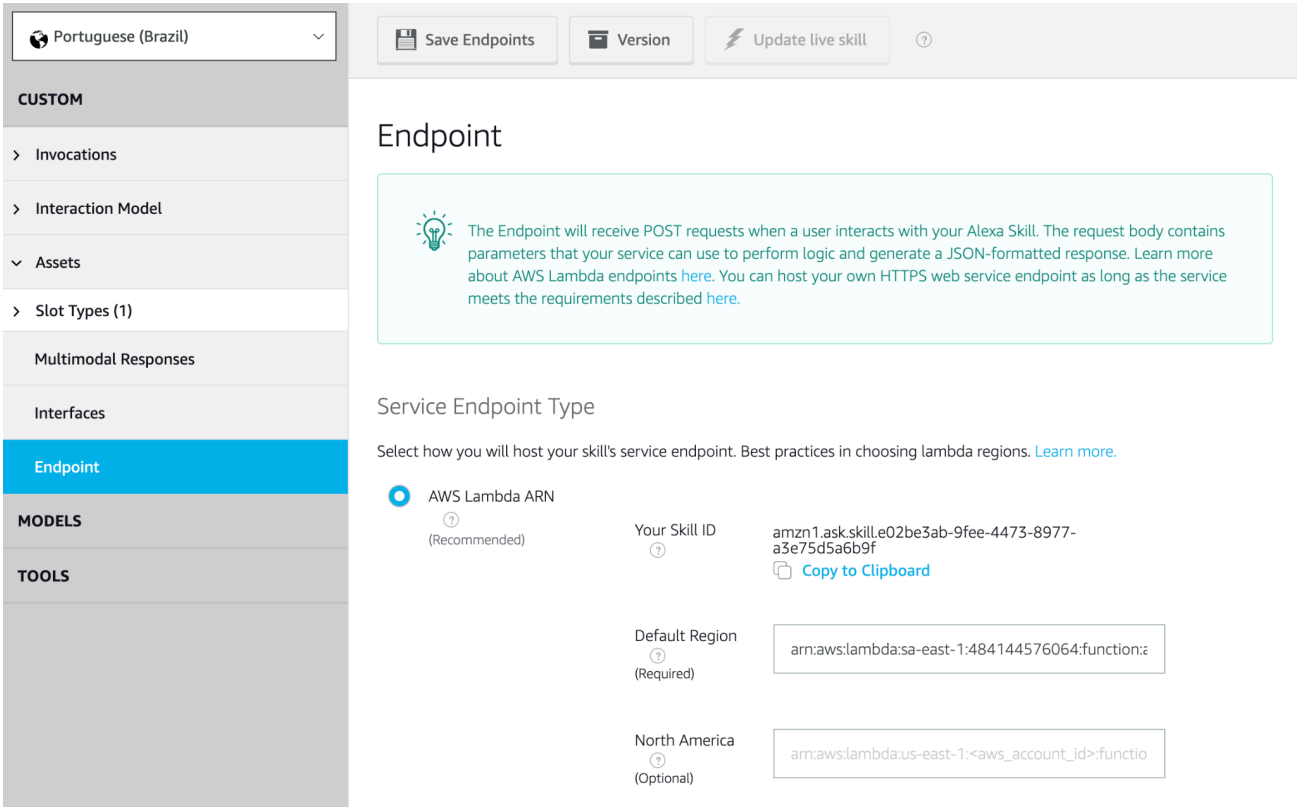
Após o desenvolvimento da *skill* é necessário fazer a exportação das funções para que seja possível a chamada fazer o *deploy* que consiste em hospedar o código no servidor de AWS para que seja possível executá-lo. Para realizar o *deploy* e hospedar a função na AWS como função Lambda é executado o comando `"$ask deploy"` ao executar o comando a é hospedado os arquivos referentes as *intents* na AWS e uma nova *skill* é hospedada no console da Amazon Alexa com o modelos de interação.

Figura 16: AWS Função Lambda.



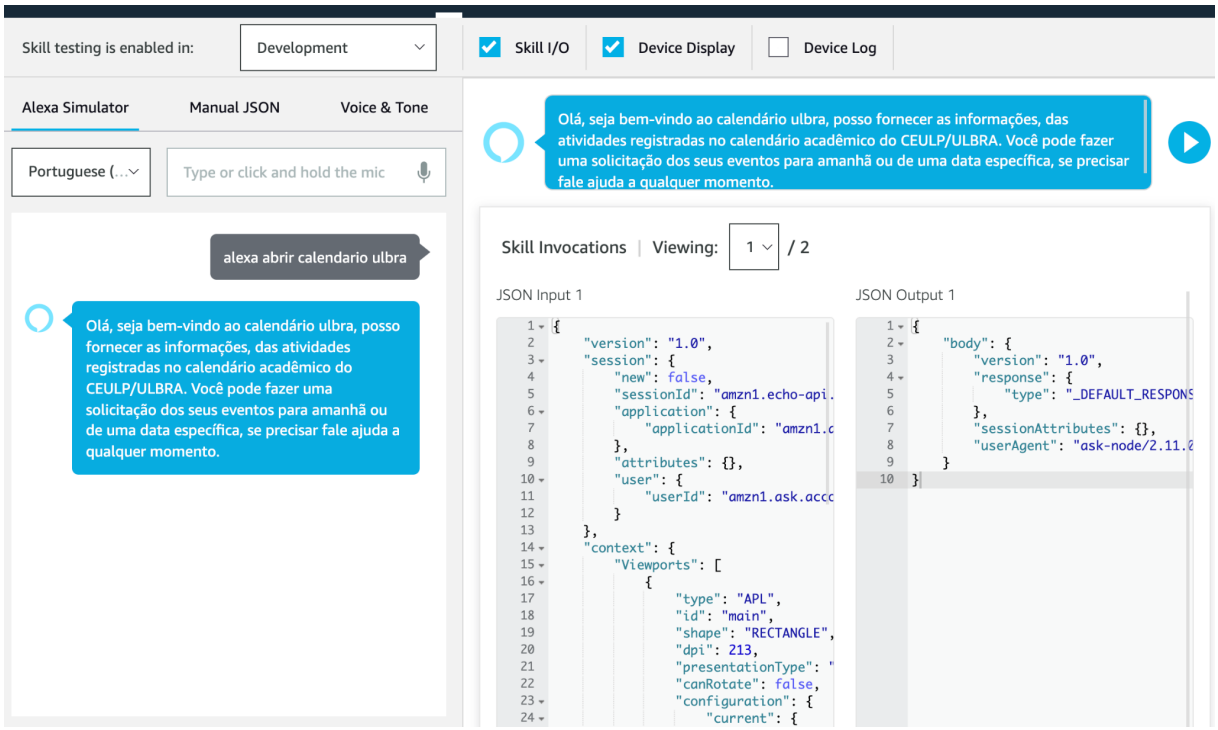
A função lambda gerada com a utilização do node versão 12.0.0 é utilizada na *skill* hospedada no console de *desenvolvimento* da Amazon Alexa, passando o parâmetro ARN da função no console para fazer referência a função. No painel da AWS é possível visualizar o código, testar o código e monitorar as chamadas que foram solicitadas pelos usuários.

Figura 17: Console desenvolvedor Alexa.



Após fazer o deploy no console de desenvolvedor em `./assets/endpoint` na variável `Default Region` é passada a referência do na Lambda ARN. Por fim, após o desenvolvimento concluído, para ver o funcionamento da *skill* sem um dispositivo ECHO foi utilizado o console da amazon.

Figura 18: Console de desenvolvedor da Alexa.



O console do desenvolvedor simula uma conversa entre o usuário e uma *skill* da mesma forma como o usuário faria se tivesse um dispositivo ECHO, na figura 18 é mostrado o resultado final da *skill* em seu pleno funcionamento.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve por objetivo desenvolver uma *skill* do calendário do CEULP/ULBRA para a Amazon Alexa. Para essa finalidade, foram realizadas etapas divididas em objetivos, desde planejar e projetar a *skill*, construir um modelo de interação de voz do usuário com a *skill*, desenvolver a api em Node.js que forneceu os dados para a *skill* do calendário acadêmico e a integração com o ASK SDK da Alexa.

Os objetivos específicos do trabalho foram atingidos, desde a etapa de planejamento a integração com o ASK SDK da Alexa para Node.js. O desenvolvimento da API e integração com o API do Google Calendar demandou um pouco mais de estudo do que o planejado, visto que foi uma etapa de mudança fora do escopo original do trabalho, essa mudança foi necessária para que se chegasse ao resultado final do trabalho.

Foi possível chegar ao objetivo geral de desenvolvimento da skill do calendário acadêmico do CEULP/ULBRA utilizando o ASK SDK da Alexa para node.js com a utilização da metodologia e os materiais abordados no trabalho, sem nenhum impedimento. O desenvolvimento deste trabalho servirá de base para trabalhos futuros relacionados ao

desenvolvimento de *skill* na Alexa, tendo em vista que esse é o primeiro trabalho acadêmico do CEULP/ULBRA relacionado a essa área de conhecimento.

Para trabalhos futuros, é possível realizar melhorias no código, fazendo a integração com a API da própria instituição e também a inserção de novas funcionalidades como por exemplo expandir não só saber informações do calendários, como também encontrar a sala, e melhorias na interface de voz com o usuário, facilitando ainda mais a utilização da *skill.*

REFERÊNCIAS

ALEXA, Detalhes do produto Amazon Alexa – **Amazon Alexa**, 2021. Accessed: 2021-04- 21. URL: https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html

COSTA, João Carlos Gonçalves. **Assistentes Virtuais para Comunicação Empresarial**. 2017.

COSTA, Elizangela Santos da et al. **Avaliando atributos de credibilidade de páginas Web utilizando Aprendizagem de Máquina**. 2020.

CHOWDHURY, Gobinda G. **Natural language processing. Annual Review of Information Science and Technology**, 37(1):51–89, jan 2005.

ESPOSITO, Jedidiah. **How to Build Your First Alexa Skill: 5 Steps to Get Started**. Amazon Alexa, [S. l.], p. 1-1, 4 set. 2018. Disponível em: https://developer.amazon.com/en-US/blogs/alexa/alexa-skills-kit/2018/09/how-to-build-your-first-alexa-skill-5-steps-to-get-started. Acesso em: 16 jul. 2021.

FERNANDES, José Thiago; ROSSI, Magali Andreia. **Interfaces de Voz com Usuário Aplicado a Sistemas para Internet**.

HENKE, Márcia et al. Aprendizagem de máquina para segurança em redes de computadores: Métodos e aplicações. **Minicursos do XI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2011)**, v. 1, p. 53-103, 2011.

JS, Node; JS, NODE. Node. js. **Tradução de: SILVA, AG Disponível em**, 2020.

LOMÔNACO, José Fernando Bitencourt et al. Desenvolvimento de conceitos: o paradigma das transformações. **Psicologia: Teoria e Pesquisa**, v. 17, n. 2, p. 161-168, 2001.
RICH, E. Knight, K. (1991). "**Artificial Intelligence**", McGrawHill. 1991.

MODERNO, Consumidor. Evolução da Alexa: a influência da assistente de voz no comportamento do consumidor. **Consumidor Moderno**, [S. l.], p. 1-1, 8 mar. 2021. Disponível em: https://www.consumidormoderno.com.br/2021/03/08/evolucao-alexa-influencia-assistente-voz-comportamento-consumidor/. Acesso em: 7 abr. 2021.

MONARD, Maria Carolina; BARANAUSKAS, José Augusto. **Conceitos sobre aprendizado de máquina. Sistemas inteligentes-Fundamentos e aplicações**, 2003, 1.1: 32.

Number of voice assistant devices in use to overtake world population by 2024, reaching 8.4bn, led by smartphones: Voice-interactive devices in use to double by 2024. 1. ed. England: Juniper research, 28 abr. 2020. Disponível em: https://www.juniperresearch.com/press/number-of-voice-assistant-devices-in-use. Acesso em: 27 jul. 2021.

PREECE, J.; Rogers, Y.; Sharp, H. (2013). **Design de Interação: além da interação humano-computador.** Tradução: Isabela Gasparini; revisão técnica: Marcelo Soares Pimenta - 3. Ed – Porto alegre: Bookman, 2013

SCHÕN, D. **El profesional reflexivo: cómo plensan los profesionales cuando actúan**, Barcelona: Paidós, 1998.

SILVA, Daniela Filipa Macedo Braga Moreira da. **Algoritmos de Processamento da Linguagem Natural para Sistemas de Conversão Texto-Fala em Português**. p. 2.. Dissertação (Doutorado) - Faculdade de Filoloxía da Universidade da Coruña, [S. l.], 2008. 187 p. Acesso em: 23 julho. 2021.

SINGH, M. P. and HUHNS, M. N. **Service-Oriented Computing: Semantics, Processes, Agents.** John Wiley & Sons, New York, NY, EUA, 1a. Edição. 2005.

SHIMABUKURO, Igor. Uso de assistentes virtuais no Brasil cresce 47% durante a pandemia. **Olhar Digita**l, [S. l.], p. 1-1, 14 out. 2021. Disponível em: https://olhardigital.com.br/2020/10/14/noticias/uso-de-assistentes-virtuais-no-brasil-cresce-47-durante-a-pandemia/. Acesso em: 26 abr. 2021.

SILVA, J. A. S.;MAIRINK, C. H. P. **Inteligência artificial: aliada ou inimiga.** LIBERTAS: Rev. Ciênci. Soc. Apl., Belo Horizonte, v. 9, n. 2, p. 64-85, ago./dez. 2019.

VIGLIAROLO, Brandon. **Amazon Alexa**: The smart person's guide. [S.I.], 2017. Disponível em: Acesso em: 24 abril. 2021.