



Pontifícia Universidade Católica do Rio de Janeiro

Assistente Virtual para Indicações em Viagens

Lucas Hardman Gomes Campos França

Primeiro relatório de Projeto Final de Graduação

Centro Técnico Científico - CTC

Departamento de Informática

Curso de Graduação em Ciência da Computação

Rio de Janeiro, Setembro de 2017



Lucas Hardman Gomes Campos França

Assistente Virtual para Indicações em Viagens

Primeiro relatório de Projeto Final, apresentado ao curso de Ciência da Computação da PUC-Rio como requisito parcial para a obtenção de te título de Bacharel em Ciência da Computação.

Orientador: Edmundo Torreão

Rio de Janeiro,

Setembro de 2017.

Índice

1.	Situação Atual.....	5
2.	Proposta e Objetivo do Trabalho	6
3.	Plano de Ação	8
4.	Cronograma.....	10
5.	Dicionário de Dados	11
6.	Especificação de Requisitos	14
7.	Viabilidade Técnica e Arquitetura de Software	16
8.	Estudos e Pesquisas	19
	Parte 1: Preparação do ambiente de desenvolvimento	19
	Parte 2: Inicialização do projeto e adição das SDK	20
	Parte 3: Integração com o Facebook.....	26
	Parte 4: Conhecendo o Watson Personality Insights	37
	Parte 4.1: Adicionando o Personality Insights ao projeto	37
	Parte 4.2: Descrição dos modelos de personalidade	40
	Parte 4.3: Entendendo o perfil retornado pelo serviço	44
	Parte 4.4: Interpretando os valores numéricos	49
	Parte 4.5: Descrição dos modelos de preferência de consumo.....	50
	Parte 5: Criação de exemplo de atrações turísticas.....	50
	Parte 6: Comparação entre o resultado do Personality Insights e as características das atrações turísticas (Match).....	50
	Parte 7: Conhecendo o Watson Conversation	50
9.	Definição do Escopo.....	51
10.	Modelagem	51
11.	Mockups.....	51
12.	Casos de Uso (depende do quanto for feito até então)	51
13.	Referências Bibliográficas	54

Introdução

Inicialmente criado pela IBM em 2011 para participar do programa televisivo americano Jeopardy, o Watson é um sistema de computação cognitiva. Desta forma, ele utiliza técnicas de “machine learning” para aprender com experiências anteriores e aplica estes conhecimentos em futuras tomadas de decisão.

O Watson veio para tentar evoluir na solução de um dos grandes problemas da computação: entender e interpretar a linguagem natural. Ele é capaz de entender dados não-estruturados, que representam 80% dos dados encontrados virtualmente em 2017. Estes dados são normalmente informações produzidas por “seres humanos” para “seres humanos”, incluindo textos como artigos, pesquisas e postagens. Ou seja, temos um cenário de um programa computacional tentando interpretar informações com regras como gramática, contexto e cultura, e podem ser ambíguas, implícitas, complexas ^[1].

A IBM já oferece diversos serviços do Watson que podem ser utilizados por qualquer pessoa com conhecimentos em computação e programação. Dentre eles temos o Conversation e o Personality Insights. O primeiro é um serviço que torna capaz uma conversa por chat entre um usuário “falante” de linguagem natural e um robô que utiliza a inteligência do Watson. O robô deve entender a linguagem natural e responder da forma “mais humana possível”. Já o segundo serviço lê e interpreta os dados retirados de redes sociais como Facebook e Twitter para identificar características de personalidade, necessidades, gostos, valores, preferências e hábitos.

Já podemos contar com aplicações do Watson em diversas áreas, como por exemplo na medicina, onde ele é capaz de recomendar terapias contra o câncer a partir do cruzamento da literatura científica com dados clínicos e genéticos dos pacientes ^[2]. Além dessa área, educação, finanças e “internet das coisas” também são focos da IBM para o Watson.

Dessa forma, o Watson é capaz de ser aplicado em diversos setores, entre eles o turismo. Os negócios desta indústria são normalmente diretos e objetivos, como montagem de pacotes de viagens, reservas de hotéis, aluguéis de carros, compras de passagens aéreas e compra de ingressos. O uso da computação cognitiva pode trazer para estes negócios um entendimento sobre as características e necessidades do cliente e com isso oferecer um serviço mais personalizado para cada cliente.

1. Situação Atual

A IBM possui uma gama de soluções para indústria de turismo e viagens que pode ser encontrada em [IBM.com/Travel](https://www.ibm.com/travel). A maior parte das companhias deste setor estão implementando soluções em forma de chatbots e robôs, ambas ainda “não maduras” o suficiente para impactar em estratégias significativas dessa indústria ou explorar toda a capacidade da tecnologia. Dessa forma, a maior parte das empresas nesta área estão apenas observando o avanço da computação cognitiva e as poucas que estão investindo lideram projetos pequenos e com baixos riscos^[3]. Este é o caso da empresa britânica Thomson, que trabalha no desenvolvimento de um chatbot interage com o usuário providenciando, em tempo real, respostas baseadas em pesquisas sobre destinos e perguntas sobre férias^[4]. Por exemplo, um robô conversa com usuário final com o intuito de sugerir as “melhores” férias baseando-se em um maior entendimento sobre o que o usuário está procurando. Outro trabalho similar é o WayBlazer, que implementa inteligência artificial para melhorar e personalizar a experiência do usuário em websites relacionados a turismo e viagens que contrataram o serviço^[5]. Neste caso, o serviço pode ser contratado para um website de turismo (por exemplo, Trivago), onde os usuários vão ter o seu comportamento mapeado (cliques, compras, e outros dados relevantes)^[6] e assim, com o tempo, o website vai ter um perfil de cada usuário (o que pode ser utilizado para diversos fins, como por exemplo passar a sugerir/facilitar a compra de produtos ou serviços relacionados a este perfil).

Por outro lado, a grande maioria dos serviços disponíveis online para essa indústria funcionam de forma onde o usuário informa o que quer e o sistema responde exclusivamente de acordo com o que foi informado. Ou seja, não há uma análise específica sobre o perfil do usuário que influencie nos resultados. Entre estes sistemas, temos o Hotel Urbano e o Trivago, que possuem a principal funcionalidade de encontrar melhores preços em hotéis. Outro sistema conhecido é o Decolar.com, que permite centralizar no aplicativo todas as informações sobre uma viagem, incluindo tickets de embarques, reservas em hotéis, alugueis de carros e ingressos de atrações turísticas, por exemplo. Nesta mesma linha há várias outras aplicações, bem comuns em forma de websites ou aplicativos para celular.

2. Proposta e Objetivo do Trabalho

Este projeto tem como proposta um estudo sobre computação cognitiva e sobre as APIs do Watson, utilizando um aplicativo mobile que funciona como assistente virtual para indicações em viagens como “prova de conceito” (POC - Proof of Concept). Desta forma, será utilizada uma API de um assistente virtual (“IMB Watson Conversation”) para viabilizar a interação com usuários relativa a seus interesses pessoais sobre viagens. O projeto também utilizará a API “IBM Watson Personality Insights” para obter o perfil do usuário, a partir da leitura e interpretação das suas contas nas redes sociais. Os dados obtidos do “Personality Insights” serão interpretados pelo aplicativo, a ser desenvolvido neste projeto, posicionando os usuários num determinado perfil para que lhes sejam feitas sugestões de acordo com seus gostos, interesses e estilo de vida.

O sistema será implementado em uma plataforma tecnológica ainda a ser definida, podendo ser Android ou iOS, e deverá pedir acesso à conta do Facebook ou do Twitter do usuário. Com as credenciais concedidas, o IBM Watson Personality Insights fará a leitura do perfil do usuário e exportará um arquivo contendo pontuações obtidas em diversas características humanas^[7]. Este arquivo deve ser interpretado pelo sistema a ser desenvolvido, que deve guardar informações que identificam as tendências de gostos e interesses do usuário. Estas informações guardadas devem servir como “base” para identificar locais de possível interesse.

Após efetuar o login e garantir o acesso à uma rede social ao sistema, o usuário se encontra em uma sala de chat, onde uma assistente virtual, funcionando através do IBM Watson Conversation^[8], inicia uma conversa. Neste dialogo, a assistente procura palavras-chave para identificar quais são os interesses do usuário naquele momento, e com isso começar a fazer sugestões baseadas nas informações armazenadas no sistema. A aceitação ou não do usuário pode alterar as informações persistidas no sistema, fazendo com que o sistema passe a oferecer sugestões melhores no futuro.

Assim como o sistema armazena informações sobre o usuário, ele também deve guardar informações sobre destinos, atrações turísticas, restaurantes, exposições e outros estabelecimentos ou eventos ao redor do mundo. Estas informações devem ser utilizadas pelo

sistema a ser implementado comparadas com as informações sobre o usuário para formar as sugestões da assistente.

Desta forma, podemos observar que estamos lidando com um volume muito grande de informações, e ainda não foi especificado em como obter parte delas. Tendo isto em vista, assim como o tempo limitado para a disciplina de Projeto Final, a quantidade de informações no sistema provavelmente será limitada. A princípio a assistente deverá conseguir conversar apenas sobre algumas cidades, e atendendo a determinadas questões.

3. Plano de Ação

Etapa 1: Documentação

- Documentação sobre o estudo realizado e os artefatos do projeto da aplicação
- Atualização do escopo conforme as mudanças necessárias durante a implementação.
- Registro de informações sobre as linhas de código para facilitar futuros retrabalhos e atualizações.
- Atualização da modelagem conforme as necessidades encontradas durante a implementação.

Etapa 2: Estudo sobre as funcionalidades das APIs do Watson

- Leitura e compreensão da documentação das APIs com a finalidade entender o funcionamento da tecnologia que vai ser utilizada.

Etapa 3: Pesquisa

- Pesquisa sobre as tecnologias a serem utilizadas no projeto e como ela se comunicam.
- Pesquisa sobre sistemas similares.

Etapa 4: Especificação de requisitos

- Definição das características que o sistema deve atender.

Etapa 5: Estudo sobre os possíveis ambientes de desenvolvimento

Etapa 6: Definição da arquitetura de software

Etapa 7: Modelagem

- Criação de modelos de dados de representação do sistema.

Etapa 8: Definição do escopo

- Definição clara e objetiva sobre todos os aspectos da aplicação.
- Detalhamento sobre todas as funcionalidades a serem implementadas.

Etapa 9: Início da Implementação

- Primeiros passos da implementação do sistema.

Etapa 10: Implementação

- Implementação completa do sistema.

Etapa 11: Testes

- Testes automatizados.
- Testes com usuários.
- Documentação dos testes.
- Elaboração de um documento apresentando os resultados finais.

Etapa 12: Ajustes e finalização

- Ajustes finais.
- Revisão
- Finalização da documentação

4. Cronograma

Projeto Final	Ago 17	Set 17	Out 17	Nov 17	Dez 17	Jan 18	Fev 18	Mar 18	Mai 18	Jun 18	Jul 18
Etapa 1: Documentação											
Etapa 2: Estudo sobre as funcionalidades das APIs do Watson											
Etapa 3: Pesquisa											
Etapa 4: Elaboração da proposta											
Etapa 5: Especificação de requisitos											
Etapa 6: Viabilidade											
Etapa 7: Estudo sobre os possíveis ambientes de desenvolvimento											
Etapa 8: Definição da arquitetura de software											
Etapa 9: Modelagem											
Etapa 10: Definição do escopo											
Etapa 11: Início da Implementação											
Etapa 12: Documentação											
Etapa 13: Implementação											
Etapa 14: Testes											
Etapa 15: Ajustes e finalização											

5. Dicionário de Dados

- Redes sociais: é uma estrutura social composta por pessoas ou organizações, conectadas por um ou vários tipos de relações, que compartilham valores e objetivos comuns. Estamos lidando apenas com Facebook e Twitter.

Exemplo: Facebook, Twitter, LinkedIn.

- Características de personalidade: são indicadores numéricos que representam o quanto um usuário possui de certa característica. Entre várias características, temos: emotividade, ansiedade, imaginatividade, curiosidade....

Exemplo: O usuário A tem o indicador de curiosidade igual à 1 (em uma escala de zero à um), o que indica que ele é muito curioso.

- Perfil de usuário: é um conjunto com várias características de personalidade, que juntas podem descrever um usuário.

Exemplo: O perfil do usuário A indica que ele é muito criativo e emotivo, porém pouco ansioso.

- Gerente do sistema: é um usuário especial com credenciais de administrador. Este usuário não possui acesso ao código fonte e não pode recompilar o sistema, porém ele tem acesso à partes restritas do programa que podem alterar os dados armazenados no sistema.

Exemplo: O gerente A do sistema, adicionou informações sobre um restaurante B da cidade C.

- Atrações: são qualquer tipo de pontos turísticos ou estabelecimentos.

Exemplo: Cristo Redentor, Pizza Hut, Praia de Copacabana, Shopping da Gávea, Comuna, mureta da Urca...

- Eventos: são acontecimentos públicos, com dia e hora marcados, que acontecem em locais públicos ou em atrações. Podem ou não ter periodicidade.

Exemplo: Festa Treta, fogos de artifício no réveillon de Copacabana, festival de food truck, feiras de artesanato...

- Perfil de atração ou evento: é um conjunto com várias características de personalidade, que juntas podem descrever a maior parte dos usuários que frequentam/gostam da atração ou do evento. Público alvo.

Exemplo: o perfil da atração “trilha da Pedra Bonita” é de pessoas aventureiras, pessoas que gostam de praticar exercício físico...

- Match de compatibilidade: conta o quanto o perfil de atração ou evento é compatível com o perfil do usuário.

Exemplo: o usuário possui 75% de match de compatibilidade com o a atração B.

- Assistente virtual: um robô que funciona através da API "IBM Watson Conversation" e conversa de forma mais humana possível com o usuário.

Exemplo: o usuário começou uma conversa sobre a sua viagem para o Rio de Janeiro com a assistente virtual a fim de receber dicas sobre o que fazer durante a viagem. A assistente entende o objetivo do usuário e começa a sugerir programas relacionados à conversa atual e ao perfil de usuário.

- Reação do usuário: informação retirada sobre a fala de um usuário sobre uma sugestão da assistente virtual. Essa informação pode ser positiva, neutra ou negativa.

Exemplo: Assistente sugere uma festa de música eletrônica e o usuário responde “não gosto de ambiente muito cheios”. Assistente recebe esta informação como uma reação negativa do usuário.

- Cloud: se refere ao termo computação na nuvem. Consiste na utilização da memória, da capacidade de armazenamento e do processamento de computadores e servidores compartilhados e interligados por meio da internet.

Exemplo: A Cloud da IBM permite a utilização de várias ferramentas nos seus servidores, entre elas temos o Watson.

- Workspace: consiste em um grupo de arquivos e ferramentas que auxiliam ou que fazem parte do desenvolvimento de um software.

Exemplo: É organizar o seu programa em um workspace, possivelmente utilizando vários arquivos.

- Entities: em português significa “entidades”. Este é um termo utilizado pelo Watson Conversation para se referenciar a alvos de uma conversa, ou seja, entidades que podem afetar o resultado da conversa ou da execução de um programa.

Exemplo: Podemos imaginar um aplicativo de viagens onde podemos utilizar linguagem natural para conversar com um robô sobre atrações. Neste caso temos como “entities” os tipos atrações, por exemplo, ao ar livre, artísticas, noturnas ou com multidões.

- Intents: em português significa “intenções”. Este termo é utilizado pelo Watson Conversation para se referenciar às intenções do usuário que afetam as “entities”.

Exemplo: Utilizando o exemplo anterior do item “entities”, as “intentes” seriam os termos sublinhados nos seguintes exemplos: “eu amo atividades ao ar livre”, “não sou uma pessoa que costuma gostar de enfrentar multidões”, “estou à procura de locais que ofereçam almoço barato”.

6. Especificação de Requisitos

Requisitos funcionais:

- O sistema deve pedir acesso ao perfil do Twitter ou Facebook do usuário.
- O sistema deve processar as informações obtidas nas redes sociais do usuário toda vez que houver login.
- As informações processadas sobre as redes sociais no momento do login devem ser utilizadas para formar um perfil de usuário.
- O perfil de usuário deve conter contadores (variáveis que podem ser inteiras, float ou booleanas) que medem características de personalidade.
- O sistema deve poder importar de outro sistema informações sobre atrações e eventos.
- O gerente do sistema deve filtrar e selecionar as atrações e eventos importados pelo sistema.
- O gerente do sistema deve preencher características de personalidade mais comuns em um perfil de atração ou evento.
- O sistema deve relacionar o perfil de atração ou evento com o perfil do usuário e criar um match de compatibilidade entre os perfis.
- O sistema deve contar um chat com uma assistente virtual.
- A assistente virtual deve saber sobre qual cidade o usuário quer conversar.
- A assistente virtual deve conversar com o usuário da maneira mais humana possível.
- A assistente virtual deve procurar informações nas falas do usuário para sugerir uma atração ou evento.
- Ao sugerir atração ou evento, a assistente virtual deve interpretar a reação do usuário.
- Quando houver uma reação do usuário, ela deve ser utilizada para efetuar alterações no perfil de atração ou evento, no perfil do usuário e, conseqüentemente, no match de compatibilidade.

Requisitos não funcionais:

- O sistema deve funcionar em qualquer versão do sistema operacional (iOS/Android) que não foi descontinuada.
- O sistema só deve poder ser acessado após login com senha.

- A assistente virtual deve responder o usuário em menos de 3 segundos.

7. Viabilidade Técnica e Arquitetura de Software

O projeto será implementado para smartphones e utilizará recursos do IBM Watson. Para definir a arquitetura devemos levar em consideração as limitações e necessidades das SDKs do Watson e dos possíveis sistemas operacionais.

I. IBM Watson Conversation:

- a) Línguas: o Conversation permite conversas em diversas linguagens naturais, entre elas estão inglês, português e espanhol. A lista completa com todos os idiomas suportados pode ser encontrada na sessão “[Supported languages](#)” da documentação ^[9].
- b) SDKs: [Java](#), [Node.js](#), [Python](#), [.NET](#), [Swift](#) e [Unity](#).
- c) Preço:

– FREE:

- 10.000 chamadas à API
- Até 5 workspaces
- Até 25 intents
- Até 25 entities

– STANDARD: R\$0,004838 reais por chamada à API

- Até 20 workspaces
- Até 2.000 intents
- Até 1.000 entities
- Cloud pública compartilhada

II. IBM Watson Personality Insights:

- a) Línguas: o Personality Insights permite a importação de diversos textos, incluindo os extraídos de redes sociais. Estes textos devem estar em redigidos em uma das “request languages” suportadas, podendo ser árabe, inglês, japonês, coreano e espanhol. Já a resposta gerada pelo Personality Insights possui outro grupo de possíveis linguagens, o “response language”, que tem mais opções como o português. A lista completa com

todos os idiomas suportados pode ser encontrada na sessão “[Language support](#)” da documentação ^[10].

b) SDKs: [Android](#), [Java](#), [Node.js](#), [Python](#), [Swift](#) e [Unity](#).

c) Preço:

– FREE:

- 1.000 chamadas à API

– TIERED (por faixa/quantidade de chamadas à API):

- 1 a 100 chamadas à API: free.
- 100 a 100.000 chamadas à API: R\$0,041 reais por chamada.
- 100.001 a 250.000 chamadas à API: R\$0,0205 reais por chamada.
- 250.001 ou mais chamadas à API: R\$0,0103 reais por chamada.

Levando em conta as necessidades da IBM Watson, vimos que podemos construir um aplicativo compatível com diversas linguagens naturais. As mais importantes são inglês, português e espanhol, pois são as mais faladas mundialmente. A língua ideal para implementar seria o português, pois além de ser a língua do nosso país, ficaria muito mais fácil testar o programa com usuários brasileiros. Porém, o Personality Insights não é atualmente compatível com idioma na parte de “request language”, o que impossibilita a leitura de redes sociais (como o Facebook e Twitter) em português. Logo, a língua utilizada será o inglês, que é a língua mais falada pela população mundial.

Outra característica importante do sistema é a sua plataforma e a sua linguagem de programação. Como vimos anteriormente, as APIs utilizadas possuem em comum as seguintes SDKs: Java, Node.js, Python, Swift e Unity. Dentre estas, as melhores para smartphones são Java, para sistema operacional Android, e Swift, para sistema operacional iOS. Deste modo, devemos comparar os prós e contras de cada linguagem para decidir qual utilizar.

Requisitos Java:

- Computador com sistema operacional Windows, Linux ou MacOS
- Eclipse (IDE)
- Android Studio (SDK)

Requisitos Swift:

- Computador com sistema operacional MacOS
- Xcode (IDE)

Os requisitos de ambas linguagens são atendidos pelo desenvolvedor. Desta forma, por uma questão de preferência a linguagem definida fica como Swift.

Sabemos que o aplicativo utilizará informações sobre diversas atrações e eventos, e que estes dados são importados de outras aplicações. Estes dados podem ser modificados por gerentes do aplicativo referente a este projeto. As informações sobre as atrações e eventos serão comuns a todos os usuários. Para que isto seja possível, eles devem ficar armazenados em um banco de dados de um servidor, que será especificado posteriormente.

Linguagem do aplicativo	Swift 4
Sistema Operacional do aplicativo	iOS 11
Sistema Operacional do ambiente de desenvolvimento	MacOS High Sierra
IDE	Xcode 9
Integração com o Watson	https://github.com/watson-developer-cloud/swift-sdk
Integração com o Facebook	https://github.com/facebook/facebook-sdk-swift

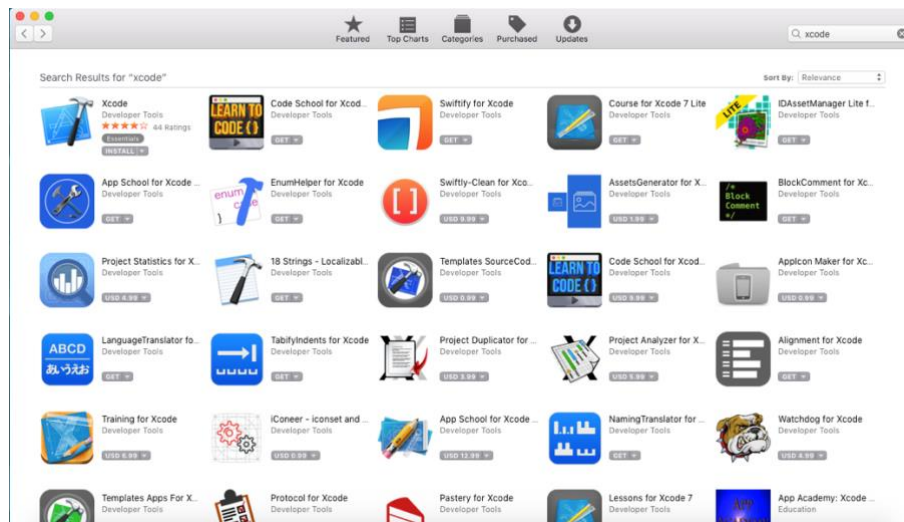
8. Estudos e Pesquisas

Nesta sessão vamos utilizar o passo-a-passo da criação de um protótipo do projeto para entender as funcionalidades do Watson, a utilizar as SDK do Watson e do Facebook, e a juntar todo esse conhecimento para a criação de um aplicativo iOS.

Parte 1: Preparação do ambiente de desenvolvimento

➔ Passo 1: Instalação da última versão do Xcode.

1. Abrir o App Store
2. Digitar “xcode” no campo de pesquisa
3. Selecionar o aplicativo Xcode



4. Apertar o botão “Install”



➔ Passo 2: Instalação do Carthage

1. Abrir o Terminal
2. Digitar o seguinte código para instalar o Homebrew

```
/usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

3. Digitar o seguinte código para instalar o Carthage

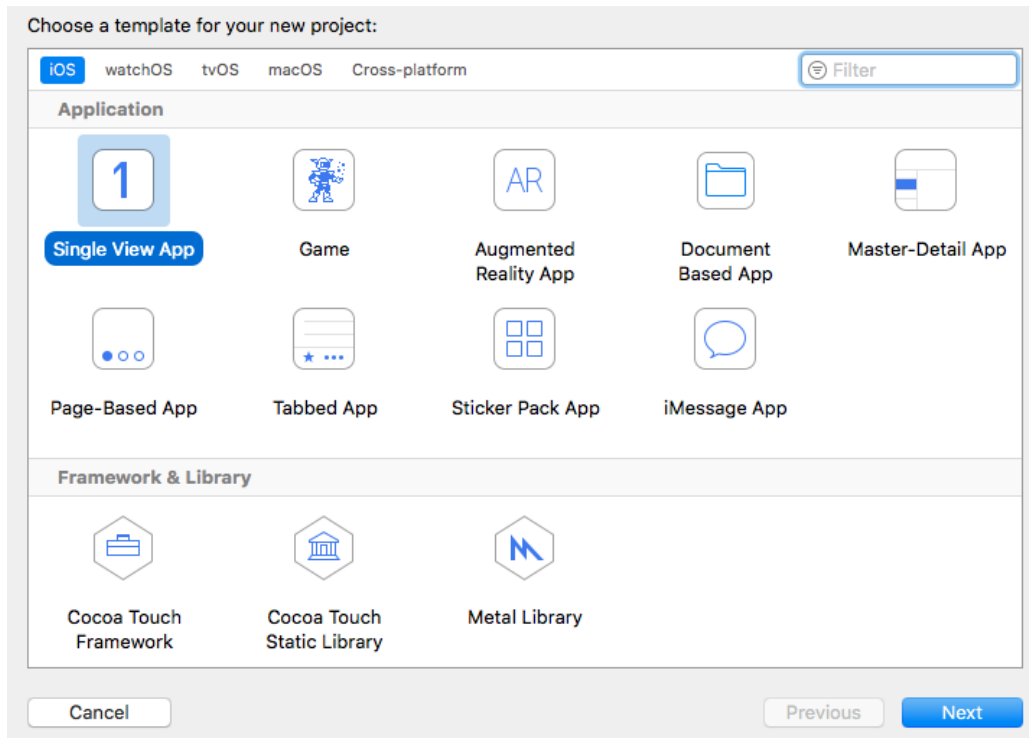
```
brew install carthage
```

Parte 2: Inicialização do projeto e adição das SDK ^[11]

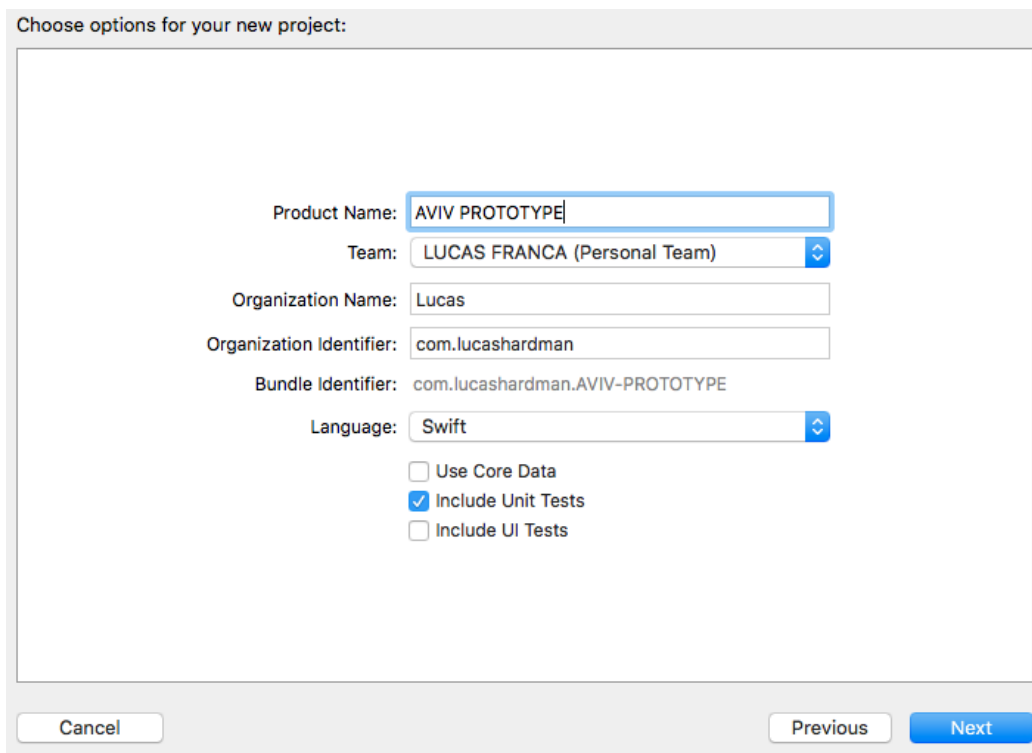
1. Abrir o Xcode
2. Criar um novo projeto selecionando “Create a new Xcode project”.



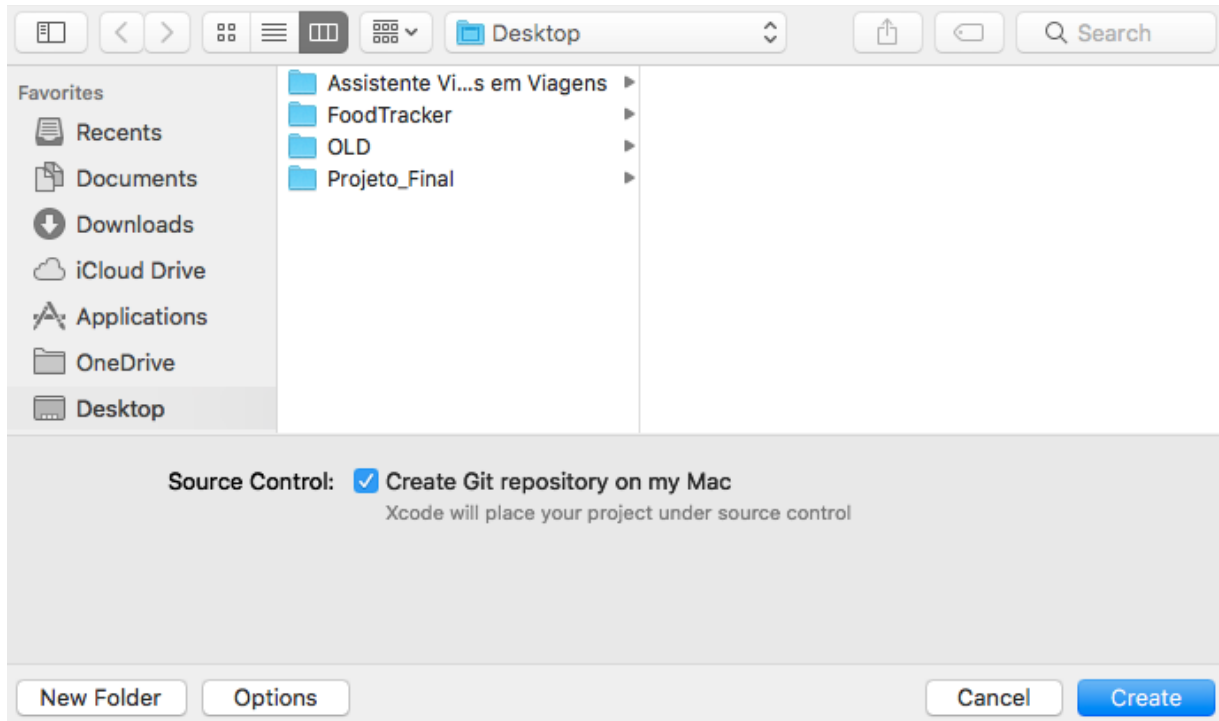
3. Selecionar iOS
4. Selecionar “Single View App”



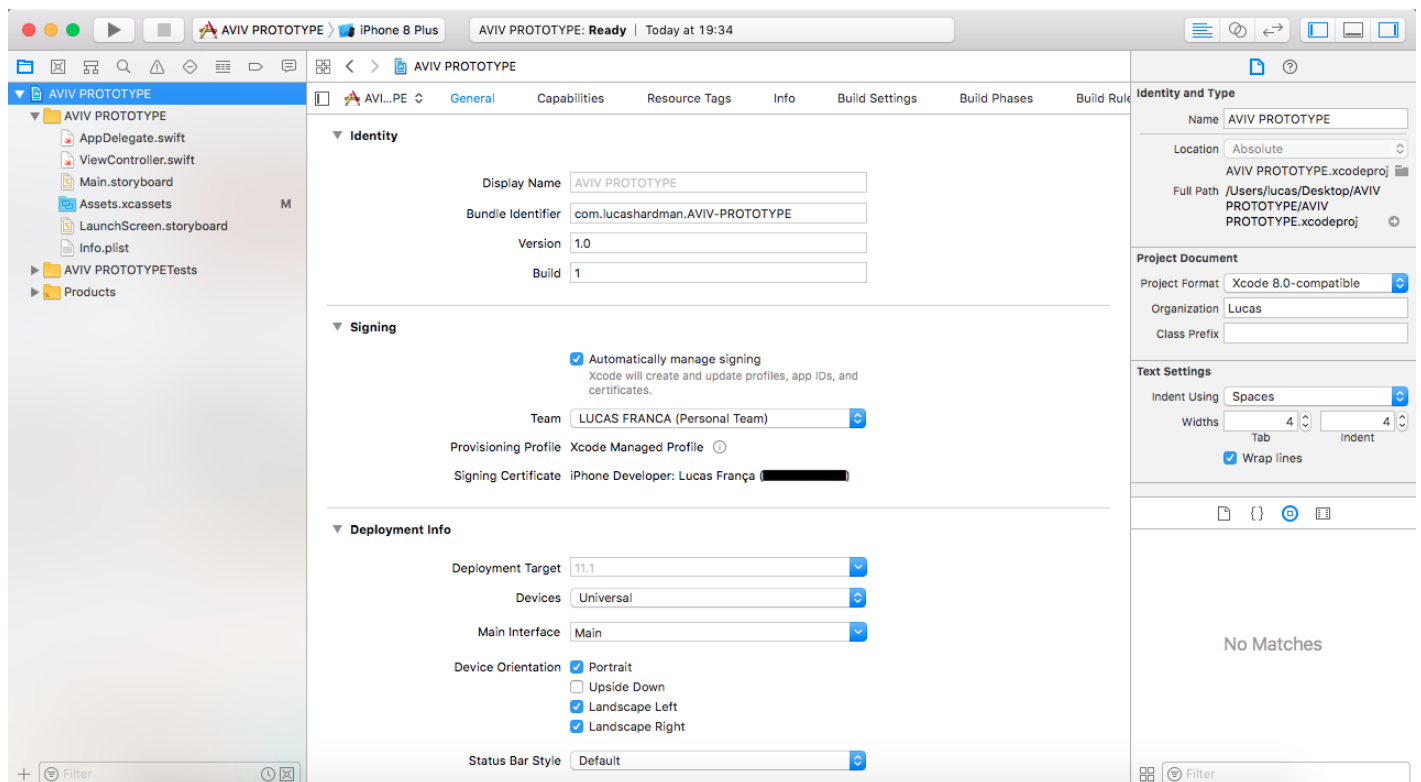
5. Na próxima janela, crie um nome para o projeto e adicione um team (caso tenha)



6. Para finalizar, escolha um diretório para ficar a pasta com o projeto.



O Xcode irá inicializar o projeto com a seguinte configuração:



A coluna da esquerda é chamada de Navegador Area e através dela é possível acessar os arquivos do projeto. Conforme necessidade, vamos ver o significado de cada arquivo. A coluna da direita é chamada de Utility Area, e a coluna do meio é chamada de Editor Area.

Agora já temos um projeto criado e precisamos adicionar as frameworks das SDKs do Watson e do Facebook.

7. Abrir uma nova janela do Finder e navegar até a pasta do projeto.
8. Criar um novo arquivo chamado "Cartfile" e adicionar os seguintes comandos.

```
github "facebook/facebook-sdk-swift"
github "watson-developer-cloud/swift-sdk"
```

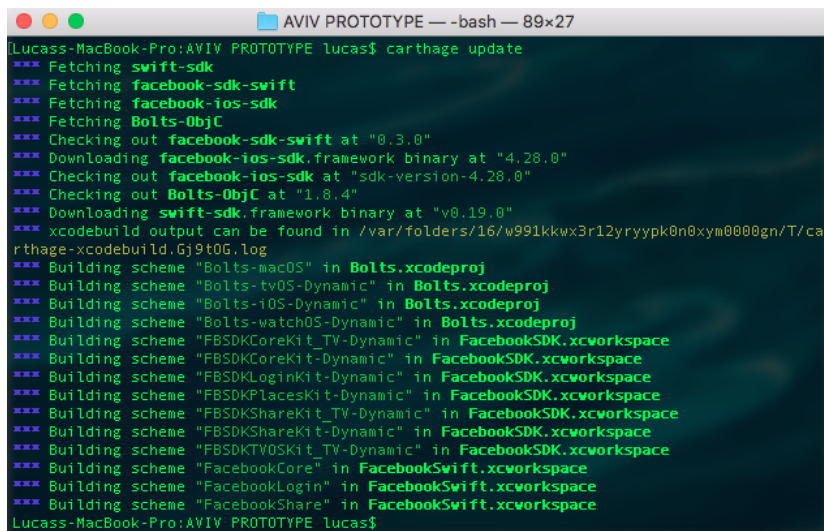
A pasta do projeto deverá ficar assim:

Name	^	Date Modified	Size	Kind
▶ AVIV PROTOTYPE		Today 13:41	--	Folder
AVIV PROTOTYPE.xcodeproj		Today 13:41	29 KB	Xcode Project
▶ AVIV PROTOTYPETests		Today 13:41	--	Folder
Cartfile		Today 10:28	78 bytes	TextEdit

9. Abrir o Terminal e navegar até a pasta do projeto.
10. Digitar o seguinte comando para adicionar as frameworks.

```
carthage update
```

O Carthage vai começar a baixar e adicionar as frameworks e se não houver erro, no final o terminal deverá estar da seguinte forma.



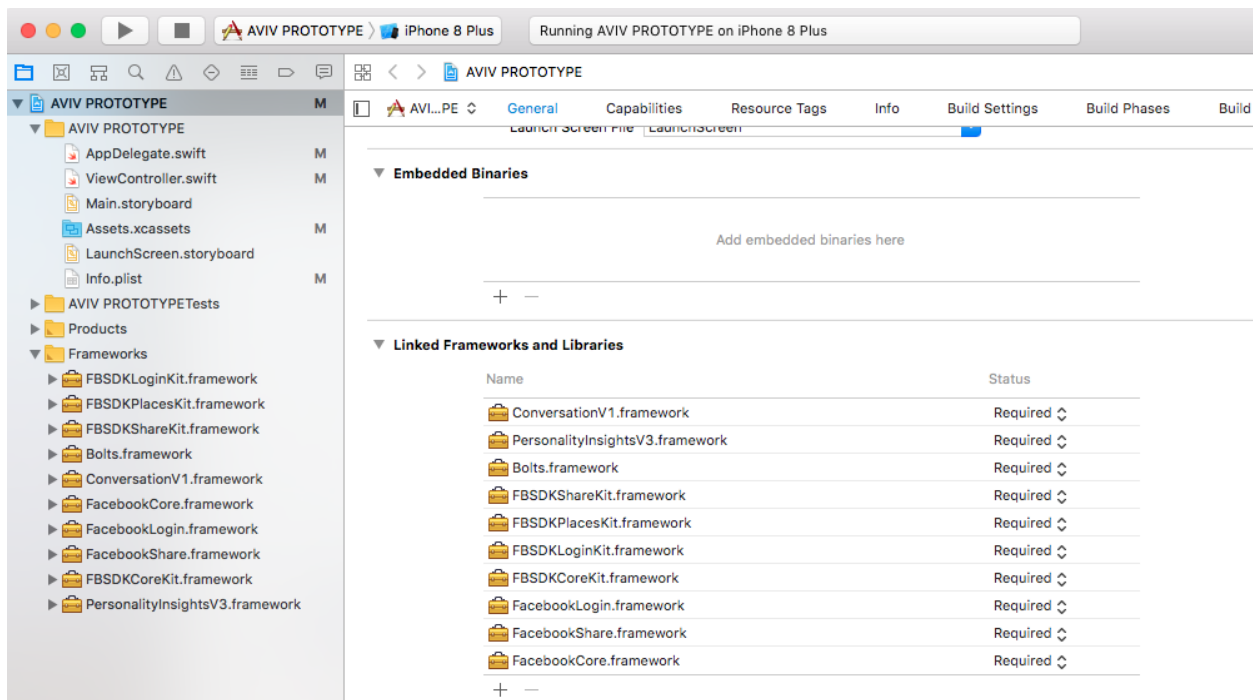
```
Lucass-MacBook-Pro:AVIV PROTOTYPE lucas$ carthage update
*** Fetching swift-sdk
*** Fetching facebook-sdk-swift
*** Fetching facebook-ios-sdk
*** Fetching Bolts-ObjC
*** Checking out facebook-sdk-swift at "0.3.0"
*** Downloading facebook-ios-sdk.framework binary at "4.28.0"
*** Checking out facebook-ios-sdk at "sdk-version-4.28.0"
*** Checking out Bolts-ObjC at "1.0.4"
*** Downloading swift-sdk.framework binary at "v0.19.0"
*** xcodebuild output can be found in /var/folders/16/w991kkwx3r12yryypk0n0xym0000gn/T/ca
rthage-xcodebuild.Gj9t0G.log
*** Building scheme "Bolts-macOS" in Bolts.xcodeproj
*** Building scheme "Bolts-tvOS-Dynamic" in Bolts.xcodeproj
*** Building scheme "Bolts-iOS-Dynamic" in Bolts.xcodeproj
*** Building scheme "Bolts-watchOS-Dynamic" in Bolts.xcodeproj
*** Building scheme "FBSDKCoreKit_TV-Dynamic" in FacebookSDK.xcworkspace
*** Building scheme "FBSDKCoreKit-Dynamic" in FacebookSDK.xcworkspace
*** Building scheme "FBSDKLoginKit-Dynamic" in FacebookSDK.xcworkspace
*** Building scheme "FBSDKPlacesKit-Dynamic" in FacebookSDK.xcworkspace
*** Building scheme "FBSDKShareKit_TV-Dynamic" in FacebookSDK.xcworkspace
*** Building scheme "FBSDKShareKit-Dynamic" in FacebookSDK.xcworkspace
*** Building scheme "FBSDKTVOSKit_TV-Dynamic" in FacebookSDK.xcworkspace
*** Building scheme "FacebookCore" in FacebookSwift.xcworkspace
*** Building scheme "FacebookLogin" in FacebookSwift.xcworkspace
*** Building scheme "FacebookShare" in FacebookSwift.xcworkspace
Lucass-MacBook-Pro:AVIV PROTOTYPE lucas$
```

No final do processo, a pasta do projeto deverá ficar semelhante à figura abaixo:

Name	Date Modified	Size	Kind
▶ AVIV PROTOTYPE	Today 13:41	--	Folder
AVIV PROTOTYPE.xcodeproj	Today 13:41	29 KB	Xcode Project
▶ AVIV PROTOTYPETests	Today 13:41	--	Folder
Cartfile	Today 10:28	78 bytes	TextEdit
Cartfile.resolved	Today 17:55	19...ytes	Document
▼ Carthage	Today 18:42	--	Folder
▼ Build	Today 18:42	--	Folder
▶ iOS	Today 17:55	--	Folder
▶ Checkouts	Today 17:55	--	Folder

As frameworks se encontram na pasta Carthage/Build/iOS

11. Voltando para o Xcode, selecione o nome do projeto no Navegador Area
12. Selecione a aba General
13. Selecione “Linked Frameworks and Libraries”
14. Selecione “+” e “Add Other”
15. Selecione as seguintes frameworks: PersonalityInsightsV3.framework, ConversationV1.framework, Bolts.framework, FBSDKCoreKit.framework, FBSDKLoginKit.framework, FBSDKPlacesKit.framework, FBSDKShareKit.framework, FacebookCore.framework, FacebookShare.framework e FacebookLogin.framework



16. Seleccione a aba Build Phases
17. Seleccione “+” e depois “New Run Script Phase”
18. Seleccione “Run Script”
19. Preencha o campo Shell com:

```
/bin/sh
```

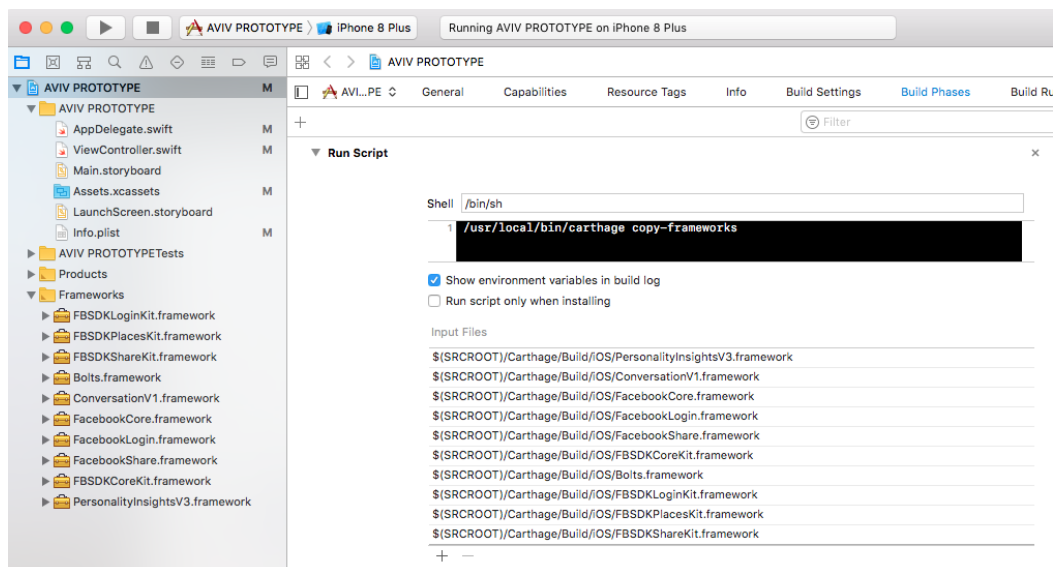
20. Preencha o campo abaixo de Shell com:

```
/usr/local/bin/carthage copy-frameworks
```

21. Marque a opção “Show environment variables in build log”
22. Desmarque a opção “Run script only when installing”
23. Preencha o campo Input Files com:

```
$(SRCROOT)/Carthage/Build/iOS/PersonalityInsightsV3.framework
$(SRCROOT)/Carthage/Build/iOS/ConversationV1.framework
$(SRCROOT)/Carthage/Build/iOS/FacebookCore.framework
$(SRCROOT)/Carthage/Build/iOS/FacebookLogin.framework
$(SRCROOT)/Carthage/Build/iOS/FacebookShare.framework
$(SRCROOT)/Carthage/Build/iOS/FBSDKCoreKit.framework
$(SRCROOT)/Carthage/Build/iOS/FBSDKLoginKit.framework
$(SRCROOT)/Carthage/Build/iOS/FBSDKPlacesKit.framework
$(SRCROOT)/Carthage/Build/iOS/FBSDKShareKit.framework
$(SRCROOT)/Carthage/Build/iOS/Bolts.framework
```

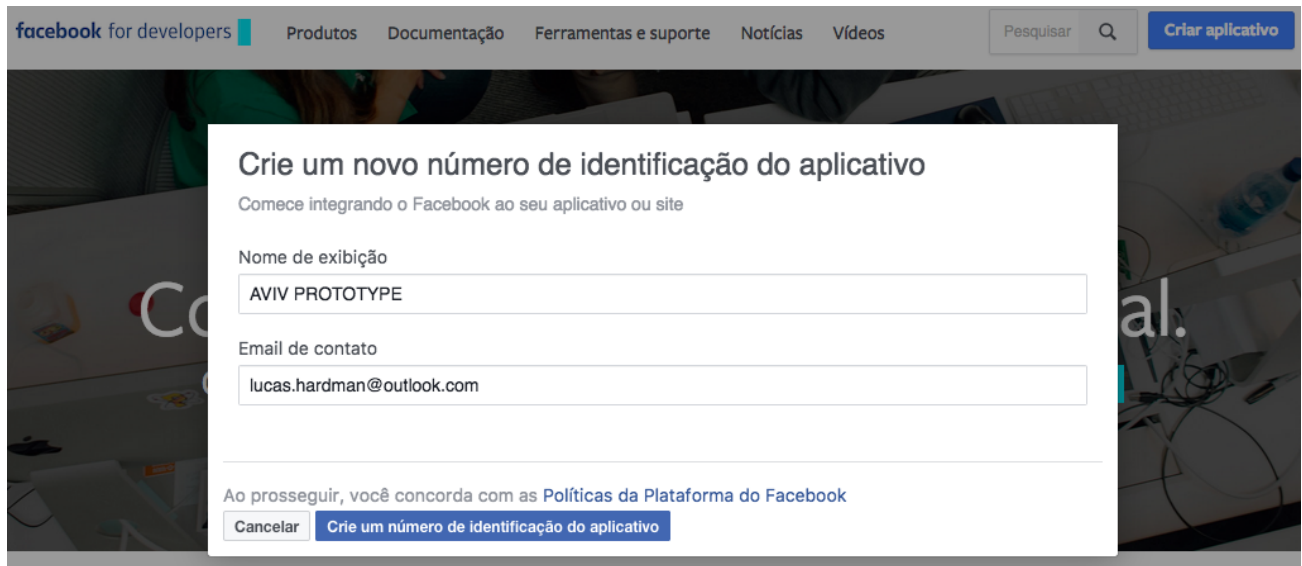
Ao final a sessão “Script Run” deverá ficar semelhante à figura abaixo:



Neste ponto já é possível compilar e rodar o programa sem gerar erros. Para isto selecione o botão de play no canto superior esquerdo do Xcode. O simulador irá iniciar, porém irá apresentar apenas uma tela branca neste momento.

Parte 3: Integração com o Facebook

1. Acessar o website <https://developers.facebook.com>
2. Clicar em “Criar Aplicativo”
3. Digitar um nome de exibição
4. Digitar um e-mail de contato
5. Clicar em “Crie um número de identificação do aplicativo”

A screenshot of the Facebook Developers website showing the 'Create New App ID' form. The form is titled 'Crie um novo número de identificação do aplicativo' and includes a sub-header 'Comece integrando o Facebook ao seu aplicativo ou site'. It has two input fields: 'Nome de exibição' with the value 'AVIV PROTOTYPE' and 'Email de contato' with the value 'lucas.hardman@outlook.com'. At the bottom, there is a checkbox for 'Ao prosseguir, você concorda com as Políticas da Plataforma do Facebook' and two buttons: 'Cancelar' and 'Crie um número de identificação do aplicativo'.

6. Envie a verificação de segurança
7. Será listado vários produtos, clique em “Login do Facebook”
8. Será oferecido um tutorial de início rápido, siga as instruções até o passo 5.

Nesse ponto o seu projeto já está completamente integrado com a SDK do Facebook. No passo 5 do tutorial anterior foi adicionado um trecho de código xml ao arquivo Info.plist, este trecho contém o nome de exibição e o número de identificação do aplicativo criado na conta de desenvolvedor do Facebook.

Nosso objetivo com o Facebook é coletar publicações do usuário e utilizar esses dados para enviar para o Watson Personality Insights. Então o primeiro passo é criar um botão que possibilite o login na conta da rede social.

1. Entre no arquivo AppDelegate.swift. Este é o arquivo responsável por mudanças de estado do aplicativo. Importe a biblioteca FBSDKCoreKit e procure a função application e reescreva-a da seguinte forma:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    FBSDKApplicationDelegate.sharedInstance().application(application,
didFinishLaunchingWithOptions: launchOptions)
    return true
}

func application(_ app: UIApplication, open url: URL, options:
[UIApplicationOpenURLOptionsKey: Any] = [:]) -> Bool {
    // Override point for customization after application launch.
    let handled = FBSDKApplicationDelegate.sharedInstance().application(app, open: url,
sourceApplication: options [UIApplicationOpenURLOptionsKey.sourceApplication] as! String!,
annotation: options [UIApplicationOpenURLOptionsKey.annotation])
    return handled
}
```

Assim temos duas versões da função. Estas funções são responsáveis por manter o funcionamento do aplicativo enquanto o login é efetuado. Isto é necessário pois para efetuar o login é necessário sair e voltar ao aplicativo novamente.

2. Entre no arquivo Main.storyboard. Aqui é possível “desenhar” o seu aplicativo, colocando cada peça no local desejado de forma visual.
3. Na parte inferior da Utility Area selecione “Show the Object Library”
4. Procure por “Container View” e arraste para a Editor Area dentro de “View Controller Scene > View Controller > View”

5. Selecione o Container View adicionado e selecione “Add New Constraints”
6. Adicione 5 no topo e 250 na direita, 16 na esquerda e 615 em baixo, depois selecione “Add 4 Constraints”
7. Selecione “Show the Identity inspector”
8. No campo Class digite “LoginButton” e selecione “Inherit Module From Target”
9. Selecione “Show the Assistant editor”
10. Pressione a tecla “control” e segure-a enquanto arrasta o Container View para o ViewController.swift, exatamente acima do método viewDidLoad()
11. Digite o nome “facebookLoginButton”, deixe o resto das opções como está e clique em “Connect”. Isso irá adicionar a seguinte linha de código no ViewController.swift:

```
@IBOutlet weak var facebookLoginButton: LoginButton!
```

Agora que adicionamos uma Container View, vamos adicionar o botão de login do Facebook nela.

1. Entre no arquivo ViewController.swift. Aqui é possível alterar as views do aplicativo.
2. Importe a biblioteca FacebookLogin
3. A classe ViewController possui o método viewDidLoad, que possibilita adicionar novas funcionalidades imediatamente após a view ser carregada. Adicione o botão utilizando o método LoginButton.
4. Adicione o botão criado ao Container View criado utilizando o método addSubview(). Observe o código a seguir para ver os detalhes.

```
import UIKit
import FacebookLogin

class ViewController: UIViewController {

    // MARK: Properties
    @IBOutlet weak var facebookLoginButton: LoginButton!

    // MARK: Override functions
    override func viewDidLoad() {
```

```

super.viewDidLoad()

let temp = LoginButton(readPermissions: [ .publicProfile, .userPosts, .email ])
temp.frame = facebookLoginButton.bounds
facebookLoginButton.addSubview(temp)
}
}

```

Agora nosso usuário já pode efetuar o login no Facebook. Com as permissões de leitura cedidas, temos acesso ao perfil público (o que contém informações como nome e aniversário), ao e-mail às postagens.

O próximo passo é acessar as postagens do usuário. É importante enfatizar que precisamos acessar os “posts”, e não “timeline” ou o “feed”. Isso se deve ao fato de que o “feed” e a “timeline” contém postagens de outras pessoas, como amigos e empresas, por exemplo, e não queremos confundir o Personality Insights com textos de terceiros.

Para acessar as postagens vamos utilizar o método `FBSDKGraphRequest` que retorna um objeto do tipo “Any”, que vamos chamar de “result”. Este objeto pode ser convertido para string utilizando `result.debugDescription`. O objeto “result” contém as postagens do usuário, e são elas que precisamos enviar para o Personality Insights.

Porém, o Personality Insights só aceita os seguintes tipos de texto: “plain text”, HTML, JSON e CSV, e o nosso objeto “result” não é de nenhum destes tipos. Precisamos então efetuar uma conversão ^[10]. Os tipos de texto mais acessíveis em Swift são o “plain text” e o JSON, mas qual é a melhor opção?

O formato JSON é o melhor formato para o conteúdo de redes sociais que consistem de várias postagens e textos, como o Facebook e o Twitter. Outra opção seria concatenar todas as postagens, criando uma grande string no formato “plain text”. Mas o formato JSON possui a vantagem de ter os textos separados, e assim o Personality Insights pode saber quais textos estão relacionados.

Então é necessário criar um método para converter a string `result.debugDescription` para JSON. Para isto, criei uma classe `AVIVUtilities` com o método

`convertFacebookResultToJSON`. Este método recebe uma string “result” (`result.debugDescription`) e retorna uma string `inputJSON`, que é a string no formato JSON pronta para ser utilizada pelo Personality Insights. Não serão explicitados os detalhes deste método neste tutorial. Mas é importante saber como é o formato das strings que estamos trabalhando.

A seguir temos um exemplo com duas postagens da string retornada pelo `FBSDKGraphRequest (result.debugDescription)`:

```
Optional({
  data = (
    {
      "created_time" = "2017-11-14T22:05:13+0000";
      id = "10155060941237546_10155057821802546";
      message = "Pedro Gomes e Clara Malizia combinando de ir pro
teatro sabado";
      story = "Lucas Hardman shared Desiludindo S/A's post.";
    },
    {
      "created_time" = "2017-11-06T17:28:13+0000";
      id = "10155060941237546_10155038260237546";
      story = "Lucas Hardman shared Nintendo Switch's video.";
    }
  );
  paging = {
    next =
"https://graph.facebook.com/v2.11/10155060941237546/posts?limit=5&format=json
&access_token=EAAOxWzzApBUBAFAPxII2bFF6VTGu7ZCg3qq5PUg62v05RuY5FdtDUWqZChwHmn
0gHU8vseZCKpWJOtdUSfMFahFL6rtj783uZAP2V1XReXAZA8k6o7CZB4B3hmGW9u2oZA2T5heZAD
plu05yyyZBnBhsVZCsYKfRVWpuUULGo7NPTSVtYk5c05tFyE0epjIROIM8TQanjAsKFAwzSSL1hLx
OW0jHPx1uFtGpfsllhuPvLD0QZDZD&until=1509989293&__paging_token=enc_AddDUFB19DZC
wi49Qpf9tmAKllalZAhcgP4Y2pn9h1bBkgeK9EbZAKH8eP7ByKUfwBoP0jtd4gDfS2zQKDlAx58pg
r4f";
    previous =
"https://graph.facebook.com/v2.11/10155060941237546/posts?limit=5&format=json
&since=1510697113&access_token=EAAOxWzzApBUBAFAPxII2bFF6VTGu7ZCg3qq5PUg62v05R
uY5FdtDUWqZChwHmn0gHU8vseZCKpWJOtdUSfMFahFL6rtj783uZAP2V1XReXAZA8k6o7CZB4B3h
mGW9u2oZA2T5heZADplu05yyyZBnBhsVZCsYKfRVWpuUULGo7NPTSVtYk5c05tFyE0epjIROIM8TQ
anjAsKFAwzSSL1hLxOW0jHPx1uFtGpfsllhuPvLD0QZDZD&__paging_token=enc_AddDWZCrWlP
0FhuiRwWAYnoo2A97aq4zkPIipxG7EzfbZB77yCbL8Rysf2ncaFw8PlxYDqmZAEo9LL4FU3KINKM3e
fve&__previous=1";
  };
})
```

A seguir temos um exemplo de JSON aceito pelo Personality Insights:

```
{
  "contentItems": [
    {
      "content": "Wow, I liked @TheRock before, now I really SEE how special he is. The daughter story was IT for me. So great! #MasterClass",
      "contenttype": "text/plain",
      "created": 1447639154000,
      "id": "666073008692314113",
      "language": "en"
    },
    {
      "content": ".@TheRock how did you Know to listen to your gut and Not go back to football? #Masterclass",
      "contenttype": "text/plain",
      "created": 1447638226000,
      "id": "666069114889179136",
      "language": "en"
    }
  ]
}
```

A estrutura desse JSON precisa necessariamente seguir o modelo definido pelo objeto Content, que é um array de objetos do tipo ContentItem. O único atributo obrigatório de ContentItem é o “content” que é o texto que será analisado pelo Watson.

Vamos criar uma classe `InputMessages` para armazenar um array de strings que será utilizado no input do `Personality Insights`. Porque fazer uma classe e não uma variável global? Este input será modificado diversas vezes, por diversos módulos do programa. Criar esta classe possibilita adicionar um “observer” e facilitar o acesso em tempo real.

```
import Foundation

class InputMessages: NSObject {

    @objc dynamic var input: [String]?

    init() {
        super.init()
        self.input = []
    }
}
```

Voltando para o nosso `ViewController`.

1. Vamos adicionar a classe `InputMessages` às propriedades do nosso `ViewController`.

```
// MARK: Properties
@IBOutlet weak var facebookLoginButton: LoginButton!
@objc var inputMessages = InputMessages.init(text: "inputMessages is empty")
```

2. Na função `viewDidLoad()` vamos adicionar um observer ao `inputMessages`.

```
// Add an Observer to inputMessages
self.inputMessages.addObserver(self, forKeyPath: #keyPath(InputMessages.input), options:
.new , context: &inputContext)
```

3. Reescrever o método `observeValue` na classe `ViewController` para imprimir no console a string `inputMessages.input` e assim poderemos observar os resultados.

```
override fun observeValue(forKeyPath keyPath: String?, of object: Any?, change:
[NSKeyValueChangeKey : Any]?, context: UnsafeMutableRawPointer?) {

    if context == &inputContext{
        print("-> InputMessage has changed...\nNew value:\n")
        if let newInput = change??.newKey as? [String] {
            print(newInput)
        }
    }
}
```

4. Voltando para o método `viewDidLoad()`, finalmente vamos chamar o `FBSDKGraphRequest` para obter as postagens do usuário.

```
FBSDKGraphRequest(graphPath: "/me/posts", parameters: ["fields": "", "limit": 250]).start {
    (connection, result, err) in

    if err != nil{
        print("Failed to start graph request: ", err!);
        return
    }

    self.inputMessages.input =
    utilities.convertFacebookResultToJSON(result.debugDescription)
}
```

Em `graphPath`, mandamos o endereço das postagens do usuário. Nos parâmetros `fields`, devemos informar quais informações das postagens queremos (por exemplo: `id` e

created_time). Mas no nosso caso, desejamos apenas as mensagens, então não precisamos enviar nenhum parâmetro deste tipo. Já o parâmetro limit (que é opcional), devemos informar a quantidade de postagens que desejamos ler. Por default o limit é um número “baixo”, com menos de 30 postagens, porém devemos que enviar um numero alto, com pelo menos 200 postagens. Isso se deve ao fato de que o Personality Insights precisa de pelo menos 600 palavras para fornecer dados suficientemente precisos.

Ao compilar e rodar o aplicativo no simulador, podemos observar os resultados no console do Xcode. Esses resultados foram enviados através do `print(inputMessages.input)` que escrevemos no “observer”.

A integração do Facebook com o aplicativo já terminou. Agora precisamos criar usuários de teste. Para isto, vamos criar:

- 3 usuários com perfis diferentes.
- Durante os testes, cada usuário será utilizado por uma pessoa escolhida pelo desenvolvedor. É importante que elas possuam gostos e interesses diferentes, pois o Personality Insights reconhece as características das pessoas a partir do texto.
- Cada usuário deverá possuir várias postagens, para prover informação suficiente para o Watson.

Para isto siga os seguintes passos:

1. Acesse <https://developers.facebook.com/>
2. Selecione “Meus aplicativos”
3. Selecione “AVIV PROTOTYPE”
4. Selecione “Funções”
5. Selecione “Usuários de Teste”
6. Já vai ter um usuário criado. Vamos adicionar mais dois.
7. Selecione “Adicionar”
8. Selecione “2” em “Número de usuários a serem criados”
9. Selecione “Criar usuários de teste”
10. Altere os nomes conforme a sua vontade

Os usuários de teste são contas do Facebook temporárias que você pode criar para testar vários recursos do seu aplicativo. [?]

Usuários de teste				Excluir	Adicionar
<input type="checkbox"/>	Nome	ID do usuário	Email		
<input type="checkbox"/>	Usuario Tres	██████████	usuario_██████_tres@tbnw.net	Editar	
<input type="checkbox"/>	Usuario Dois	██████████	usuario_██████_dois@tbnw.net	Editar	
<input type="checkbox"/>	Usuario Um	██████████	usuario_██████_um@tbnw.net	Editar	

AnteriorAvançar

Parte 4: Conhecendo o Watson Personality Insights

Parte 4.1: Adicionando o Personality Insights ao projeto

Para continuar precisamos ter uma conta no IBM Cloud para utilizar os diversos serviços fornecidos pelos seus servidores. Após criar sua conta, siga os seguintes passos:

1. Acesse o seu dashboard: <https://console.bluemix.net/dashboard>
2. Acesse o menu no canto superior esquerdo
3. Selecione “Watson” no final da lista
4. Selecione “Wason Services > Browse Services”
5. Selecione “Personality Insights”
6. Selecione “Add Services”
7. Preencha os campos e selecione “Create Project”
8. Selecione “Service credentials”
9. Selecione “View credentials ” e anote o seu usuário (username) e a sua senha (password)

Voltando para o Xcode, acesse a sua classe ViewController. Vamos adicionar a ela o serviço do Personality Insights.

1. Importe a framework PersonalityInsightsV3

```
import PersonalityInsightsV3
```

2. Adicione o Personality Insights às propriedades da classe ViewController

```
private let personalityInsights = PersonalityInsights(username: "digite o usuário aqui", password: "digite a senha aqui", version: "digite a data de hoje aqui")
```

3. Crie uma função `printlnInput` que recebe um array de strings, converte ele para array de `ContentItem` e passa esse array como parâmetro para o método `getProfile` do `Personality Insights`.

```
// MARK: Functions
func printlnInput (text: [String]){

    let error = { (error: Error) in print(error) }

    var content: [ContentItem] = []

    // Passa os valores de text para content
    for t in text{
        print ("\n",t,"\n")
        content.append(ContentItem.init(content: t))
    }

    personalityInsights.getProfile(fromContentItems: content, failure: error){ profile in
        for big_five in profile.personality{
            print("\n-> ",big_five.name," : ",big_five.percentile)
            for facet in big_five.children!{
                print(facet.name," : ",facet.percentile)
            }
        }
    }
}
```

4. Chame a função criada no “observer”.

```
override func observeValue(forKeyPath keyPath: String?, of object: Any?, change:
[NSKeyValueChangeKey : Any]?, context: UnsafeMutableRawPointer?) {

    if context == &inputContext{
```

```
print("-> InputMessages has changed...\nNew value:\n")
if let newInput = change?[.newKey] as? [String] {
    printInput(text: newInput)
}
}
```

Na prática o input do método `getProfile` difere da documentação. Não precisamos nos preocupar com tratamentos especiais de JSON, pois o Watson já interpreta o vetor de `ContentItem` como um JSON.

Parte 4.2: Descrição dos modelos de personalidade

- **Big Five:** são características de personalidade representadas pelo modelo descrevendo como uma pessoa se envolve com o mundo. Elas possuem sub-características (traços) que são chamadas de “facet”.
 - **Needs:** descreve quais aspectos de um produto podem combinar com uma pessoa.
 - **Values:** descreve fatores de motivação que influenciam a tomada de decisão de uma pessoa.
-
- a. **Big Five Agreeableness:** mede a tendência de uma pessoa a ser compassiva e compreensiva com outras pessoas. Em Português, seria agradabilidade.
 - a. **Altruism / Altruistic:** ajudar o próximo está mais perto de ser uma realização pessoal do que ser um sacrifício. Português: altruísmo.
 - b. **Cooperation / Accommodating / Compliance:** não gosta de confronto. Estão dispostos a se comprometer e abrir mão de suas necessidades para agradar o próximo. Português: cooperação.
 - c. **Modesty / Modest:** são despretensiosos e humildes. No entanto não necessariamente têm autoconfiança ou auto-estima. Português: modesto.
 - d. **Morality / Uncompromising / Sincerity:** não são pretenciosos e manipuladores. Por outro lado, são francos e genuínos. Português: moralidade / sinceridade.
 - e. **Sympathy / Empathetic:** são carinhosos e compassivos. Português: simpáticos
 - f. **Trust:** assumem que a maior parte das pessoas são justas, honestas e possuem boas intenções. Normalmente estão dispostas a perdoar e esquecer.
 - b. **Big Five Conscientiousness:** mede a tendência de uma pessoa a agir de forma planejada e organizada. Português: conscienciosidade.
 - a. **Achievement striving / Driven:** se esforça para alcançar a excelência. A sua busca por ser reconhecido como caso de sucesso o mantém focado em trabalhar para alcançar seus objetivos. Português: esforço para alcançar seu objetivo.
 - b. **Cautiousness / Deliberate:** pensam cuidadosamente sobre as possibilidades e consequências de suas ações antes de tomar alguma decisão. Português: cauteloso.

- c. **Dutiful / Sense of responsibility:** possuem um grande senso de responsabilidade e dever. Português: obediente / atencioso.
 - d. **Orderliness / Organized:** são bem organizados. Português: organizado.
 - e. **Self-discipline / Persistent:** possuem autodisciplina e força de vontade para persistir em tarefas difíceis ou desagradáveis até serem concluídas. Português: autodisciplina / persistência.
 - f. **Self-efficacy / Sense of competence:** são confiantes na sua habilidade de finalizar tarefas. Português: senso de competência.
- c. **Big Five Extraversion:** mede a tendência de uma pessoa a buscar estímulo na companhia de outras pessoas. Português: extrovertido.
 - a. **Activity level / Energetic:** se movimentam e fazem atividades de maneira rápida, energética e vigorosa. Normalmente estão envolvidos em várias tarefas. Português: ativo / energético.
 - b. **Assertiveness:** gostam de assumir o controle de direcionar as tarefas de outras pessoas. Tendem a ser líderes de grupos. Português: assertividade.
 - c. **Cheerfulness / Positive emotions:** frequentemente experienciam sentimentos positivos, como felicidade, prazer e otimismo. Português: alegre.
 - d. **Excitement-seeking:** normalmente precisam de estímulos elevados para não ficarem entediados. Português: intensos.
 - e. **Friendliness / Outgoing / Warmth:** normalmente gostam de outras pessoas e demonstram sentimentos positivos publicamente. Português: caloroso.
 - f. **Gregariousness / Sociable:** acham que a companhia dos outros é agradavelmente estimulante e gratificante. Tendem a gostar da emoção das multidões. Português: sociável.
- d. **Big Five Emotional Range:** também conhecida como neuroticismo, é a medida em que uma pessoa é sensível às emoções negativas do ambiente.
 - a. **Anger / Fiery:** tendência de sentir raiva. Português: raiva.
 - b. **Anxiety:** frequentemente sente que algo desagradável, perigoso ou ameaçador está para acontecer. Português: ansiedade.
 - c. **Depression / Melancholy:** tendem a reagir mais prontamente aos altos e baixos da vida. Português: depressão / melancolia.

- d. **Immoderation / Self-indulgence:** tendem a ser orientados para prazeres e recompensas de curto prazo em vez de consequências de longo prazo. Português: imediatismo.
 - e. **Self-consciousness:** são sensíveis ao que os outros pensam sobre eles. Estão preocupados com rejeições e acontecimentos vergonhosos. Se sentem envergonhados e desconfortáveis perto de outras pessoas com frequência. Português: envergonhados.
 - f. **Vulnerability / Sensitivity to stress:** não sabem lidar com estresse. Podem se sentir confusos, sozinhos e em pânico quando estão sob pressão ou quando estão diante de uma situação emergencial.
- e. **Big Five Openness:** mede a tendência de uma pessoa estar aberta a novas experiências. Português: capacidade receptiva.
- a. **Adventurousness:** estão sempre procurando experimentar novas atividades. Para eles a familiaridade e a rotina devem ser evitadas. Português: aventureiro.
 - b. **Artistic Interests:** amam a beleza, tanto na arte quanto na natureza. São facilmente envolvidos em eventos naturais e artísticos. Português: interesses artísticos.
 - c. **Emotionality / Depth of emotions:** possuem consciência de seus próprios sentimentos. Português: emotividade.
 - d. **Imagination:** usam a fantasia como um caminho para criar um mundo interior mais rico e interessante do que o mundo real. Português: imaginação.
 - e. **Intellect:** são intelectualmente curiosos e tendem a pensar em símbolos e abstrações. Português: intelectual.
 - f. **Liberalism / Tolerance for diversity:** estão de prontidão para desafiar a autoridade, a convenção e os valores tradicionais. Português: liberalismo / tolerância a diversidade.
- f. **Needs:**
- a. **Excitement:** querem sair e viver a vida, ter emoções animadas e se divertir. Português: excitação.
 - b. **Harmony:** apreciam as outras pessoas, seus pontos de vista e seus sentimentos. Português: harmonia.
 - c. **Curiosity:** tem a curiosidade de conhecer coisas novas, aprender e crescer. Português: curiosidade.

- d. **Ideal:** almejam a perfeição e a melhoria da comunidade. Português: idealista.
 - e. **Closeness:** preferem estar conectadas à família e estabelecer um lar. Português: valorizam a família.
 - f. **Self-expression:** gostam de aprender mais sobre si mesmos e se autoafirmar. Português: auto-expressão.
 - g. **Liberty:** possuem gostos por moda e novidades. Precisam de espaço para estar livres. Português: liberdade.
 - h. **Love:** gostam de contato social, tanto em grupo como em casal. Se sentem envolvidos em campanhas de reunir as pessoas. Português: amável.
 - i. **Practicality:** desejam ter o trabalho feito/pronto. Buscam eficiência e habilidade, que podem incluir experiência e expressão física. Português: praticidade.
 - j. **Stability:** buscam por estabilidade. Preferem o que foi provado e testado. Português: estabilidade.
 - k. **Challenge:** possuem ansia em alcançar os objetivos e serem bem-sucedidos. Português: desafio.
 - l. **Structure:** tem a necessidade de que as coisas precisam estar organizadas e sob controle. Português: não toleram a inexistência de um padrão.
- g. **Values:**
- a. **Self-transcendence / Helping others:** demonstram preocupação com o bem-estar e os interesses dos outros. Português: auto-transcendência.
 - b. **Conservation / Tradition:** enfatizam a auto-restrição e ordem. Resistem à mudanças. Português: conservador.
 - c. **Hedonism / Taking pleasure in life:** procuram prazer e gratificação para si mesmos. Português: hedonismo.
 - d. **Self-enhancement / Achieving success:** procuram sucesso e melhorias para si mesmos. Português: auto-aprimoramento.
 - e. **Open to change / Excitement:** estão abertos a novas experiências e enfatizam ações independentes. Português: aberto a mudanças.

Parte 4.3: Entendendo o perfil retornado pelo serviço

A função `printlnInput` por enquanto está apenas imprimindo no console o valor de “profile”, que é um JSON no qual o objeto `Profile` é o objeto de maior nível. Este objeto possui as seguintes propriedades:

- **word_count:** inteiro que representa a quantidade de palavras lidas do arquivo de entrada. Este inteiro pode conter um valor menor do que o valor real de palavras do arquivo, caso este arquivo seja muito grande.
- **processed_language:** string que representa o nome da linguagem natural que o serviço utilizou para processar o arquivo de entrada. Por exemplo: en (inglês) ou es (espanhol).
- **personality:** array recursivo de objetos `Trait` que descreve cada característica do usuário (Big Five dimensions e facets), obtida do arquivo de entrada.
- **needs:** array de objetos `Trait` que descreve cada necessidade do usuário (Needs) obtida do arquivo de entrada.
- **values:** objeto de `Trait` que descreve cada valor do usuário (Values) obtido do arquivo de entrada.
- **behavior:** array de objetos `Behavior` que descreve a distribuição do comportamento do usuário, através da informação gerada distribuída pelos dias das semanas e horas dos dias. Este campo só é retornado caso o arquivo JSON de entrada possua um campo de horário de data.
- **consumption_preferences:** array de objetos `ConsumptionPreferencesCategory` que fornecem resultados para cada categoria de preferência de consumo. O retorno deste campo é opcional e para acontecer o parâmetro `consumption_preferences` deve ser setado como `true`.
- **warnings:** array de objetos `Warning` que geram mensagens de alerta associadas ao arquivo de entrada. Caso não haja alertas o array é vazio.

```
{
  "word_count": 15223,
  "processed_language": "en",
  "personality": [
    . . .
  ],
  "needs": [
    . . .
  ],
  "values": [
    . . .
  ]
}
```

```

],
"behavior": [
    . . .
],
"consumption_preferences": [
    . . .
],
"warnings": []
}

```

O objeto Profile sempre inclui arrays de objetos Trait que representam Personality, Needs e Values. Para Needs e Values, estes arrays possuem apenas um nível, contendo em cada posição informações sobre uma necessidade ou valor. Já para o Personality, o array é duplo e representa as características Big Five. Cada característica Big Five possui um campo “children” que representa um novo array de objetos Trait, e em cada posição possui uma característica específica de cada Big Five. Cada posição do array de objetos Trait possui os seguintes campos:

- **trait_id**: id único, em forma de string, para cada característica. Este id possui os formatos abaixo, trocando “characteristic” pelo nome da característica.
 - **big5_characteristic**: dimensões Big Five. Ex: big5_openess.
 - **facet_characteristic**: características da dimensão Big Five. Ex: facet_altruism.
 - **need_characteristic**: necessidades Need. Ex: need_challenge.
 - **value_characteristic**: valores Value. Ex: value_conservation.
- **name**: uma string legível pelo usuário contendo o nome da característica.
- **category**: categoria da característica. Pode ser uma das três opções abaixo:
 - **personality**: para dimensões Big Five ou facets.
 - **needs**: para necessidades.
 - **values**: para valores.
- **percentile**: um número do tipo double que contém a pontuação da característica em percentil.
- **raw_score**: é um número do tipo double que contém a pontuação bruta da característica. Este campo é opcional, só retorna caso o parâmetro raw_scores seja true na chamada do método getProfile().
- **significant**: é um booleano que sempre é true para inglês, espanhol e japonês. Mas pode ser false para as línguas árabe e coreano, que ainda possuem certa limitação no serviço. Devido a esta limitação os resultados podem não ser confiáveis.

- **children:** é um array de objetos Trait que fornece resultados detalhados para cada característica (facet) de cada dimensão Big Five. Só é retornado quando o trait_id for uma big5_characteristic

```
{
  . . .
  "personality": [
    {
      "trait_id": "big5_openness",
      "name": "Openness",
      "category": "personality",
      "percentile": 0.8011555009553,
      "raw_score": 0.77565404255038,
      "significant": true,
      "children": [
        {
          "trait_id": "facet_adventurousness",
          "name": "Adventurousness",
          "category": "personality",
          "percentile": 0.89755869047319,
          "raw_score": 0.54990704031219,
          "significant": true
        },
        . . .
      ]
    },
    {
      "trait_id": "big5_conscientiousness",
      . . .
    },
    {
      "trait_id": "big5_extraversion",
      . . .
    },
    {
      "trait_id": "big5_agreeableness",
      . . .
    },
    {
      "trait_id": "big5_neuroticism",
      . . .
    }
  ],
  "needs": [
    {
      "trait_id": "need_challenge",
      "name": "Challenge",
      "category": "needs",
      "percentile": 0.67362332054511,
      "raw_score": 0.75196348037675,
      "significant": true
    },
    . . .
  ],
}
```

```

"values": [
  {
    "trait_id": "value_conservation",
    "name": "Conservation",
    "category": "values",
    "percentile": 0.89268222856139,
    "raw_score": 0.72135308187423,
    "significant": true
  },
  . . .
],
. . .
}

```

Caso o JSON de entrada contenha marcadores de tempo (dia e hora), o objeto Profile irá conter um campo com um objeto do tipo Behavior para cada dia da semana e cada hora do dia. Este objeto inclui:

- **trait_id**: id único, em forma de string, para cada característica. Este id possui os formatos abaixo, trocando “day” pelo dia da semana e “hour” pelo horário do dia.
 - **behavior_day**: para dias da semana. Ex: behavior_sunday.
 - **behavior_hour**: para horas do dia: Ex: behavior_0100, 1hora da manhã.
- **name**: string legível pelo usuário contendo o dia da semana ou horário do dia. Ex: “Sunday”, ou “0:00 am”
- **category**: string contendo a categoria da característica, sempre será “behavior”
- **percentage**: um número double que representa a percentagem de objetos ContentItems que ocorreram durante certa hora do dia ou dia da semana.

Para o projeto em questão o objeto Behavior não é importante, então não há preocupação em adicionar marcadores de dias e horas no JSON de entrada.

```

{
  . . .
  "behavior": [
    {
      "trait_id": "behavior_sunday",
      "name": "Sunday",
      "category": "behavior",
      "percentage": 0.21392532795156
    },
    . . .
  ],
  . . .
}

```

Como vimos anteriormente, ao fazer o requerimento de um Profile, podemos passar como parâmetro true no campo consumption_preferences, e assim receber um objeto ConsumptionPreferencesCategory que detalha as preferências de consumo do usuário. Este objeto contém:

- **consumption_preference_category_id:** string contendo um id único para cada categoria de preferência de consumo. Ex: "consumption_preferences_shopping"
- **name:** string contendo o nome da categoria de forma legível para o usuário.
- **consumption_preferences:** um array de objetos do tipo ConsumptionPreferences, que contém resultados detalhados de cada preferência de consumo do usuário.

Algumas categorias de preferência de consumo possuem apenas uma preferência, outras possuem várias. Cada preferência é detalhada a partir de um objeto ConsumptionPreferences, que contém os seguintes campos:

- **consumption_preference_id:** string contendo um id único para cada preferência de consumo. Ex: "consumption_preferences_automobile_ownership_cost".
- **name:** string contendo o nome da categoria de forma legível para o usuário. Ex: Likely to be sensitive to ownership cost when buying automobiles".
- **score:** número double que contém a pontuação que indica a probabilidade do usuário preferir tal item.

```
{
  . . .
  "consumption_preferences": [
    {
      "consumption_preference_category_id":
"consumption_preferences_shopping",
      "name": "Purchasing Preferences",
      "consumption_preferences": [
        {
          "consumption_preference_id":
"consumption_preferences_automobile_ownership_cost",
          "name": "Likely to be sensitive to ownership cost when buying
automobiles",
          "score": 0
        },
        . . .
      ]
    },
    . . .
  ]
}
```


Parte 4.4: Interpretando os valores numéricos

Neste ponto já estudamos o objeto Profile completo e sabemos que ele detalha resultados em objetos do tipo Trait (personalidade: Big Five e facet), Behavior (comportamento) e ConsumptionPreferences (preferências de consumo). Todos estes objetos detalham características da personalidade do usuário com indicadores numéricos do tipo double.

a. Percentil de características de personalidade (percentile):

Para cada requerimento, o serviço fornece uma pontuação através do atributo percentile para cada característica de personalidade Big Five, Needs e Values. Esta pontuação representa um valor de percentagem (variando de 0 a 1) para cada característica baseado no texto de entrada, e é normalizada através de uma comparação com dados obtidos de uma amostra da população.

Por exemplo, o percentile 0.64980796071382 da característica big5_extraversion indica que o usuário pontuou esta característica mais do 64% população e menos do que 34%.

b. Valores brutos de características de personalidade (raw score):

Os valores brutos raw score são os mesmos que os percentile, porém obtidos sem a normalização com uma amostra da população. Eles podem ser interpretados como resultados obtidos em uma prova de personalidade.

Estes valores são fornecidos para desenvolvedores que querem implementar uma normalização personalizada. Por exemplo, criando uma normalização diferente levando em conta a região e cultura. Por isto estes valores não são retornados por default, e para retornar é necessário passar true como parâmetro raw scores na chamada do método.

c. Percentagens de características de comportamento: (percentage):

Caso haja indicadores de tempo no JSON de entrada, o serviço retorna uma percentagem para cada característica de comportamento. Isto serve para identificar uma distribuição temporal dos dados analisados, ou seja, o valor indica qual percentagem dos textos de entrada está distribuído em cada hora do dia ou dia da semana.

Por exemplo, a pontuação da característica behavior_sunday foi 0.4561049445005. Isto significa que 46% dos textos analisados (ContentItems) foram criados no domingo.

d. Pontuação por preferência de consumo (scores):

Conforme interesse, o Personality Insights pode retornar uma análise sobre os interesses de consumo do usuário do texto analisado. Para isto basta especificar como true o parâmetro consumption_preferences na hora de fazer o requerimento de um objeto Profile.

Para representar os interesses de consumo, cada objeto do tipo ConsumptionPreferences possui um atributo score, que é um indicador numérico double. Este atributo é um indicador de preferência e representa a probabilidade de o usuário preferir tal item.

O atributo score pode possuir um dos três valores a seguir:

- **0.0:** Interesse muito baixo em certo item.
- **0.5:** O usuário é neutro quanto ao item.
- **1.0:** Alto nível de interesse no item.

Parte 4.5: Descrição dos modelos de preferência de consumo

Baseado nos modelos de características de personalidade Big Five, Needs e Values, o Watson é capaz de produzir um relatório sobre as preferências de consumo do usuário a partir dos seus dados de entrada. Para isto, é necessário que o parâmetro consumption_preferences seja enviado como true na hora da chamada ao método getProfile.

a. Compras:

- consumption_preferences_automobile_ownership_cost:** consideram o gasto total da aquisição de automóveis antes de concluir uma escolha.
- consumption_preferences_automobile_safety:** consideram os fatores de segurança para comprar automóveis.
- consumption_preferences_clothes_quality:** preferem qualidade ao comprar roupas.
- consumption_preferences_clothes_style:** preferem estilo e moda ao comprar roupas.
- consumption_preferences_clothes_comfort:** preferem conforto ao comprar roupas.
- consumption_preferences_influence_brand_name:** são influenciados pela marca do produto ao fazer compras.

- g. **consumption_preferences_influence_utility:** são influenciados pela utilidade do produto antes de efetuar a compra.
 - h. **consumption_preferences_influence_online_ads:** são influenciados por propagandas online antes de efetuar compras.
 - i. **consumption_preferences_influence_social_media:** são influenciados por redes sociais antes de efetuar compras.
 - j. **consumption_preferences_influence_family_members:** são influenciados por membros da família antes de efetuar compras.
 - k. **consumption_preferences_spur_of_moment:** podem efetuar compras por impulso.
 - l. **consumption_preferences_credit_card_payment:** preferem utilizar cartão de crédito para efetuar compras.
- b. Filmes:**
- a. **consumption_preferences_movie_romance:** tendem a preferir filmes de romance.
 - b. **consumption_preferences_movie_adventure:** tendem a preferir filmes de aventura.
 - c. **consumption_preferences_movie_horror:** tendem a preferir filmes de horror.
 - d. **consumption_preferences_movie_musical:** tendem a preferir musicais.
 - e. **consumption_preferences_movie_historical:** tendem a preferir filmes históricos.
 - f. **consumption_preferences_movie_science_fiction:** tendem a preferir filmes de ficção científica.
 - g. **consumption_preferences_movie_war:** tendem a preferir filmes de guerra.
 - h. **consumption_preferences_movie_drama:** tendem a preferir filmes de drama.
 - i. **consumption_preferences_movie_action:** tendem a preferir filmes de ação.
 - j. **consumption_preferences_movie_documentary:** tendem a preferir documentários.
- c. Música:**
- a. **consumption_preferences_music_rap:** tendem a gostar do estilo Rap.
 - b. **consumption_preferences_music_country:** tendem a gostar do estilo Country.
 - c. **consumption_preferences_music_r_b:** tendem a gostar do estilo R&B.

- d. **consumption_preferences_music_hip_hop:** tendem a gostar do estilo Hip Hop.
 - e. **consumption_preferences_music_live_event:** tendem a gostar de shows e eventos ao vivo.
 - f. **consumption_preferences_music_playing:** possivelmente possuem experiências em tocar algum instrumento.
 - g. **consumption_preferences_music_latin:** tendem a gostar de música latina
 - h. **consumption_preferences_music_rock:** tendem a gostar do estilo Rock.
 - i. **consumption_preferences_music_classical:** tendem a gostar de música clássica.
- d. **Leitura e aprendizado:**
- a. **consumption_preferences_read_frequency:** tendem a ler com frequência.
 - b. **consumption_preferences_books_entertainment_magazines:** tendem a ler revistas de entretenimento.
 - c. **consumption_preferences_books_non_fiction:** tendem a ler livros que não são de ficção.
 - d. **consumption_preferences_books_financial_investing:** tendem a ler livros sobre investimento financeiro.
 - e. **consumption_preferences_books_autobiographies:** tendem a ler autobiografias.
- e. **Saúde e atividade física:**
- a. **consumption_preferences_eat_out:** tendem a comer em restaurantes ou fora de casa com frequência.
 - b. **consumption_preferences_gym_membership:** tendem a estar inscritos em alguma academia ou pratica de esporte.
 - c. **consumption_preferences_outdoor:** tendem a gostar de atividades ao ar livre.
- f. **Empreendedorismo:**
- a. **consumption_preferences_start_business:** podem considerar começar um novo negócio nos próximos anos.
- g. **Meio ambiente:**
- a. **consumption_preferences_concerned_environment:** demonstram interesse e preocupação sobre questões ecológicas.

h. Voluntariado:

- a. consumption_preferences_volunteer:** tendem a se voluntariar para ajudar em causas sociais.

9. Referências Bibliográficas

[1] – YouTube: IBM Watson: How it Works. Acesso em 20/09/2017. Disponível em:

https://www.youtube.com/watch?v=_Xcmh1LQB9I&t=3s

[2] – IBM: Conheça o Watson e seu uso na saúde. Acesso em 20/09/2017. Disponível em:

<https://www.ibm.com/blogs/robertoa/2017/03/conheca-o-watson-e-seu-uso-na-saude/>

[3] – IBM: Beyond bots and robots: Exploring the unrealized potential of cognitive computing in the travel industry. Acesso em 20/09/2017. Disponível em:

<https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=GBE03776USEN&>

[4] – Forbes: Bringwater, Adrian. Come Fly With AI, IBM Cloud Builds 'Chatbot' Virtual Travel Agent. Acesso em 20/09/2017. Disponível em:

<https://www.forbes.com/sites/adrianbridgwater/2016/11/22/come-fly-with-ai-ibm-cloud-builds-chatbot-virtual-travel-agent/#1088409b4813>

[5] – WayBlazer: Acesso em 20/09/2017. Disponível em:

<https://www.wayblazer.ai/>

[6] – Baseline: Greengard, Samuel. WayBlazer's Journey Leads to Cognitive Computing. Acesso em 24/09/2017. Disponível em:

<http://www.baselinemag.com/cloud-computing/wayblazers-journey-leads-to-cognitive-computing.html>

[7] – Watson Documentação: Acesso em 20/09/2017. Disponível em:

<https://console.bluemix.net/docs/services/personality-insights/models.html#models>

[8] – Watson Documentação: Acesso em 20/09/2017. Disponível em:

<https://console.bluemix.net/docs/services/conversation/index.html#about>

[9] – Watson Documentação: Acesso em 10/10/2017. Disponível em:

<https://console.bluemix.net/docs/services/conversation/lang-support.html#supported-languages>

[10] – Watson Documentação: Acesso em 10/10/2017. Disponível em:

<https://console.bluemix.net/docs/services/personality-insights/user-overview.html#overviewLanguage>

[11] – Watson GitHub: Acesso em 15/11/2017. Disponível em:

<https://github.com/watson-developer-cloud/swift-sdk/blob/master/docs/quickstart.md>