PUC
RIO

Pontifícia Universidade Católica do Rio de Janeiro

Assistente Virtual para Indicações em Viagens

Lucas Hardman Gomes Campos França

Relatório de Projeto Final de Graduação

Centro Técnico Científico - CTC

Departamento de Informática

Curso de Graduação em Ciência da Computação

Rio de Janeiro, Julho de 2019

Lucas Hardman Gomes Campos França

Assistente Virtual para Indicações em Viagens

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

Orientador: Edmundo Torreão

Rio de Janeiro,

Julho de 2019.

Resumo

Neste trabalho foram realizados o projeto e o desenvolvimento de 'aplicativo' em linguagem Swift, para o ambiente iOS, que realizará sugestões para viagens turísticas de acordo com a 'personalidade do usuário'. É utilizada a API 'Watson Personality Insights' da IBM que se utiliza de técnicas de análise linguística para inferir características intrínsecas da personalidade dos indivíduos com base em comunicações digitais, como e-mails, mensagens de texto, tweets, postagens de fóruns e de redes sociais. No caso deste projeto, foram utilizadas as postagens da rede social Facebook. O usuário do 'aplicativo' poderá consultar as atrações/eventos turísticos de uma cidade, podendo selecionar algum filtro como 'gastronomia', 'compras' ou 'natureza'. O 'aplicativo', então, compara o 'perfil de personalidade' do usuário com as características das atrações/eventos turísticos disponibilizadas em sua base de dados, apresentando suas sugestões.

Palavras-chave: aplicativo, turismo, adequação, personalidade.

Abstract

In this work we designed and developed a Swift language 'app' for the iOS environment, which will make suggestions for tourist travel according to the 'user personality'. For this to be possible, an IBM API called Personality Insights is used. This API uses linguistic analysis in order to infer intrinsic characteristics about the user personality based on digital communication texts such as e-mail, text messages, tweets, forums and social media networks posts. In this project is used the social media network Facebook. The app's users may check a city touristic attraction. To do that the user should digit the city and, if necessary, select filters, such as 'gastronomy', 'shopping' or 'nature'. The 'app', then, compares the user's 'personality profile' with the touristic attractions and event's characteristics available at the database, presenting its suggestions.

Keywords: app, tourism, adaptive, personality.

Índice

1. Proposta e Objetivo do Projeto

Este projeto tem como objetivo o desenvolvimento de um aplicativo iOS capaz de sugerir programas de entretenimento, passeios e eventos em viagens de acordo com a personalidade do usuário. Para que isto seja possível, serão utilizadas diversas APIs. O aprendizado da utilização e integração dessas APIs é parte do objetivo do projeto.

A proposta principal do aplicativo é fazer sugestões para viagens de acordo com a personalidade do usuário. Para que isto seja possível é utilizada uma API da IBM chamada Watson Personality Insights [1], que se utiliza de técnicas de análise linguística para inferir características intrínsecas da personalidade dos indivíduos com base em comunicações digitais, como e-mails, mensagens de texto, tweets e postagens de fóruns ou redes sociais. Desta forma, para acessar o aplicativo, a API exige que o usuário faça login pelo Facebook e repasse suas postagens para o Personality Insights.

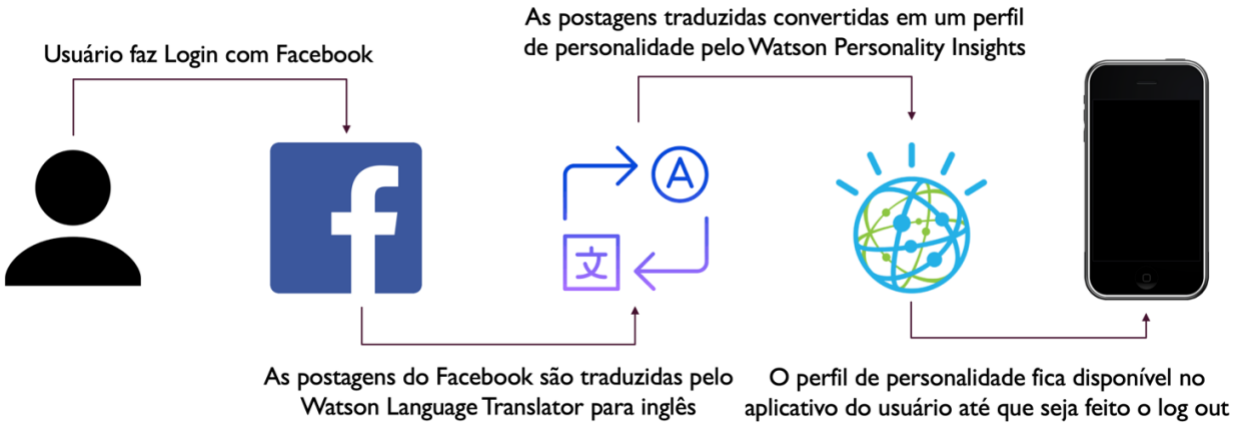


Figura 1 – Passos para a geração do perfil de personalidade

Em paralelo com o aprendizado sobre as APIs utilizadas, está o aprendizado sobre a linguagem de programação Swift e análise sobre as características de IHC e usabilidade do software.

O usuário poderá navegar pelo aplicativo e fazer pesquisas, selecionando a cidade que deseja consultar e podendo utilizar filtros de categorias de sugestão, como programas de entretenimento ‘noturnos’, ao ‘ar livre’ ou ‘artísticos’. O software responderá à pesquisa do usuário com uma lista de sugestões, onde cada uma delas pode ser clicada, causando um redirecionamento para uma página que detalha as informações.

2. Especificação de Requisitos

Requisitos funcionais:

- O aplicativo deve exigir login pelo Facebook antes de dar acesso ao seu conteúdo.
- O aplicativo deve aceitar perfis do Facebook com postagens em português.
- O aplicativo deve utilizar as postagens do Facebook para criar um perfil de personalidade para o usuário.
- O aplicativo deve avisar quando a quantidade de postagens não for o suficiente para a criação de um perfil de personalidade.
- O aplicativo deve ter acesso à uma base de dados com perfis de programas de entretenimento, passeios e eventos para serem oferecidas como sugestões turísticas.
- As sugestões turísticas devem ter perfis com os mesmos indicadores de personalidade do usuário.
- O perfil de uma sugestão turística deve ser baseado a partir de um conjunto de análises/considerações feitas por usuários, disponíveis no site de críticas TripAdvisor.
- O aplicativo deve ter uma janela específica para pesquisa, onde o usuário deve fornecer uma cidade de destino e escolher se quer utilizar filtros de busca.
- Toda sugestão turística apresentada para o usuário deve ter um indicador de porcentagem que represente a compatibilidade entre o seu perfil e o perfil do usuário.
- As sugestões devem ser apresentadas ao usuário através de uma lista ordenada decrescentemente por percentual da compatibilidade com o usuário.
- A lista de sugestões deve exibir, para cada sugesão, o nome, a categoria, o ‘início’ da descrição, uma imagem e uma ‘barra’ que demonstre visualmente a compatibilidade com o usuário.
- O aplicativo deve possibilitar o usuário marcar sugestões turísticas como ‘favoritas’.
- O aplicativo deve ter uma janela específica para consultar as sugestões turísticas marcadas como ‘favoritas’.

Requisitos Não funcionais:

- O aplicativo só pode ser utilizado com conexão à internet.
- Todos os dados sobre o usuário devem estar persistidos em uma base de dados.

3. Viabilidade Técnica e Arquitetura de Software

O projeto será implementado para smartphones iOS e utilizará recursos do IBM Watson, Facebook e Firebase. A seguir temos algumas informações sobre as versões dos recursos utilizados.

Linguagem do aplicativo	Swift 5
Sistema Operacional do aplicativo	iOS 12.2
Sistema Operacional do ambiente de desenvolvimento	MacOS Mojave 10.14.5
IDE	Xcode 10.2
Integração com o Watson	https://github.com/watson-developer-cloud/swift-sdk
Integração com o Facebook	https://github.com/facebook/facebook-sdk-swift
Banco de Dados	Firebase Cloud Firestore

Este aplicativo é uma solução de sistemas de recomendação e é baseado em uma publicação da IBM, 'Sugestões de Entretenimento usando Computação Cognitiva' [2]. A publicação sugere uma aplicação de sugestão de filmes de acordo com o perfil do usuário criado pelo Watson Personality Insights. O perfil do usuário é comparado com o perfil do filme, que possui os mesmos indicadores, e caso o perfil seja similar, o filme é sugerido. Mas como são criados os perfis dos filmes? É escolhido um site de críticas e avaliações, neste caso o IMDb, para que um webcrawler atue buscando críticas positivas (dando nota acima de 7 ao filme, por exemplo) e envie uma lista com essas críticas para o Watson Personality Insights criar um perfil para o filme. Ou seja, o perfil do filme é um perfil criado a partir do 'público' que 'consome' aquele conteúdo.

O foco deste projeto é apenas o aplicativo de sugestões para viagens que será utilizado pelo usuário final. Então a criação dos perfis dos eventos, estabelecimentos e pontos turísticos sugeridos será feita de uma outra forma, ou seja, será desenvolvido um outro aplicativo 'de apoio', também descrito neste documento, para ler um perfil do Facebook das possíveis sugestões, criar um perfil Watson Personality Insights e enviar para o Firebase. Este perfil do

Facebook será criado ‘à mão’ de forma que cada postagem seja uma crítica com nota maior ou igual a 7, retirada de sites de críticas e análises dos consumidores.

A partir deste momento, vamos tratar como “sugestão turística” tudo aquilo que a aplicação pode sugerir para o usuário, como estabelecimentos, eventos e pontos turísticos.

Existem 3 principais processos para este projeto:

1 – Adicionar sugestões turísticas para os usuários no Firebase:

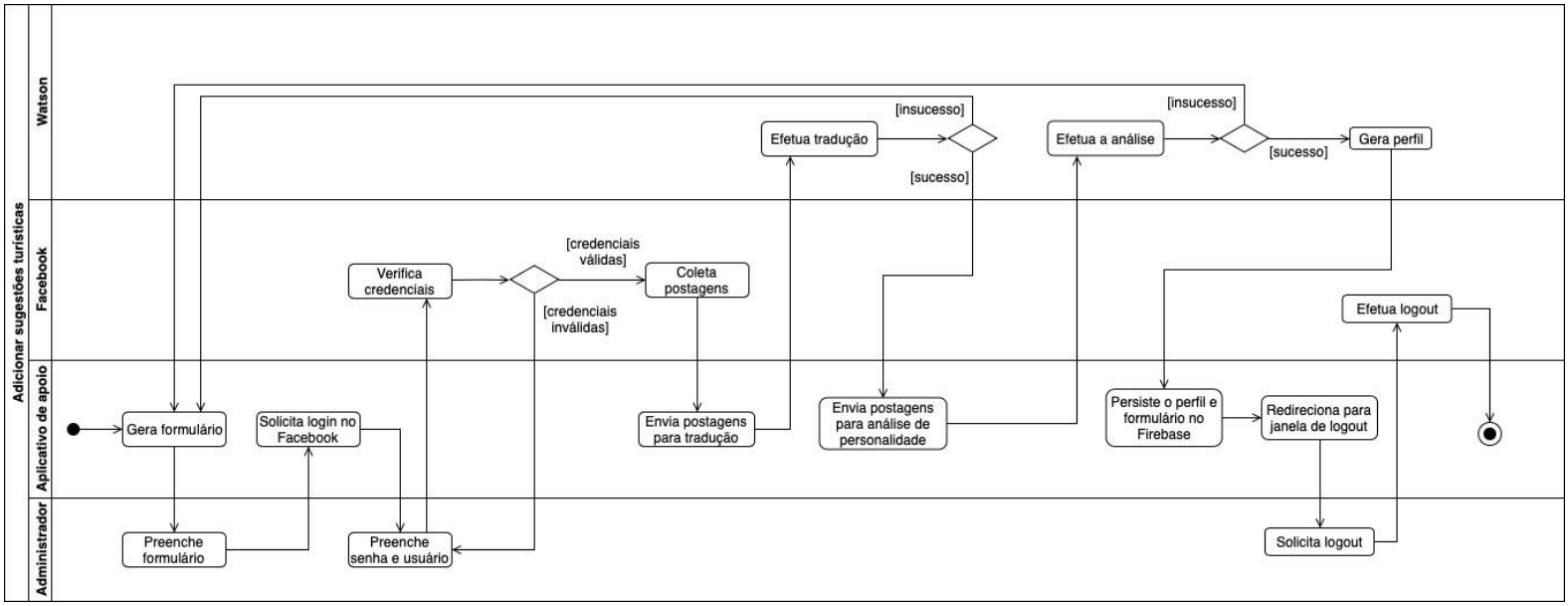


Figura 2: Diagrama de atividades – Adição de sugestão turística

- Foram criados usuários de teste no Facebook para cada sugestão turística. Para cada um desses usuários, foram adicionadas diversas postagens com avaliações com notas máximas retiradas do site TripAdvisor ‘manualmente’.
- Ao entrar no aplicativo de ‘apoio’, o administrador encontra um formulário pedindo nome da sugestão turística, a cidade em que ela se encontra, a sua categoria, o link para o seu site oficial, um link para uma imagem e um texto descritivo.
- Após preencher o formulário, o administrador clica no botão enviar e então o aplicativo de ‘apoio’ pede para fazer login no facebook. Devem ser preenchidas credenciais referentes ao usuário de teste criado para aquela sugestão turística.
- Efetuado o login, o aplicativo de ‘apoio’ lê as postagens do perfil do usuário de teste, envia para o Watson Language Translator traduzir para inglês e depois envia o resultado para

o Watson Personality Insights gerar um perfil de personalidade para esta sugestão turística.

- Com o formulário preenchido e o perfil de personalidade gerado com sucesso, o aplicativo de ‘apoio’ realiza a persistencia das informações obtidas para o Firebase.
- Após as informações serem enviadas para o Firebase, o administrador deve apertar o botão “Concluir”, para fazer logout do Facebook e retornar ao formulário.

2 – O usuário faz login no aplicativo

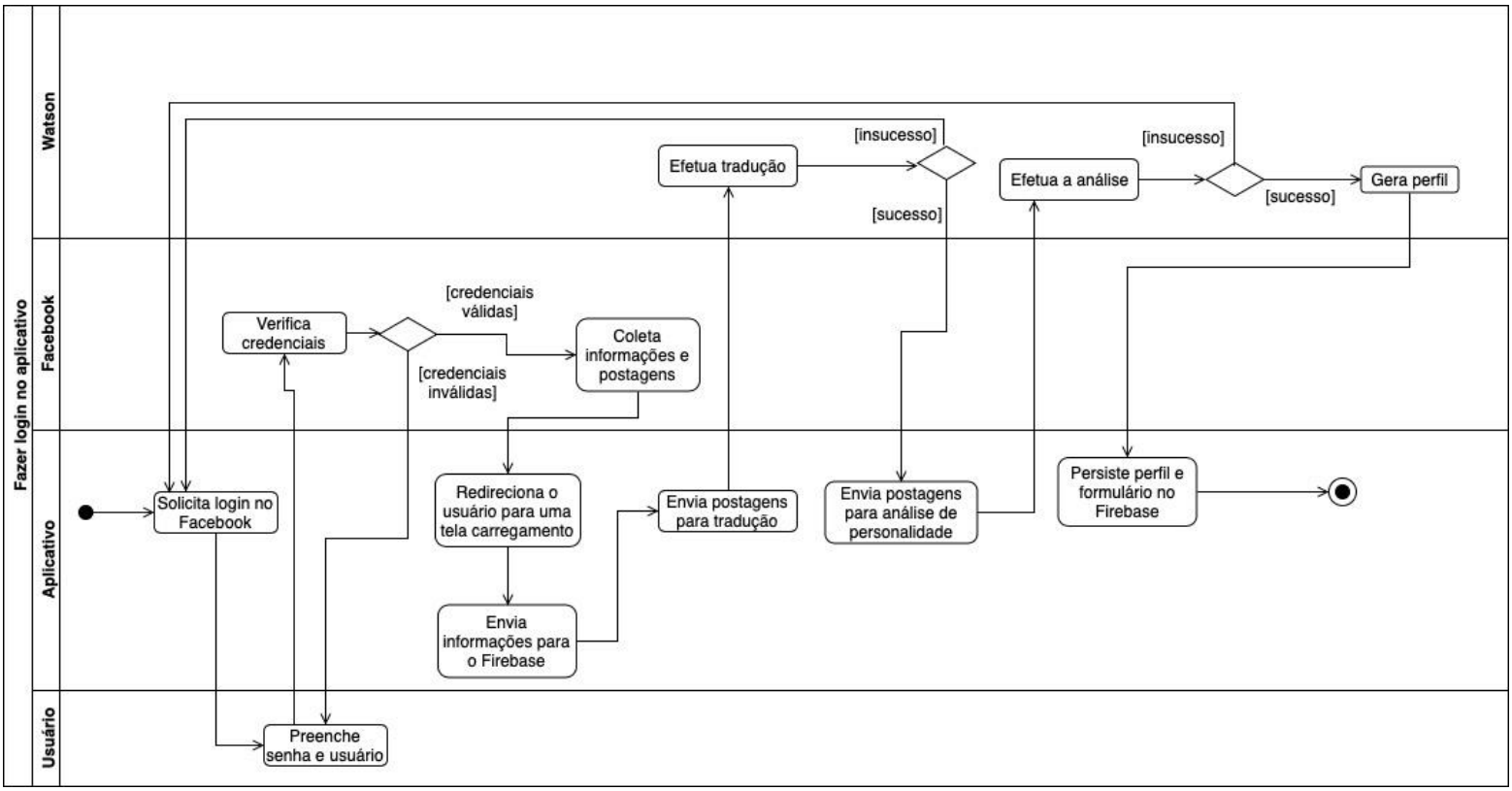


Figura 3: Diagrama de atividades – Login no aplicativo

- Ao abrir o aplicativo, o usuário sempre encontra uma janela inicial com um botão de pedido de login, que ao ser clicado, o Facebook pede o e-mail e a senha.
- Caso o login seja efetivado com sucesso, o Facebook transfere para o aplicativos os seguintes dados: nome, gênero, e-mail, id e postagens.
- Usuário é transferido para uma janela de carregamento. Nesta parte, o aplicativo processa os dados obtidos do Facebook, traduzindo as postagens para inglês pelo Language Translator, gerando o perfil de personalidade pelo Personality Insights e persistindo-os no Firebase.

- O aplicativo redireciona o usuário para a sua janela principal, que possui 3 guias: uma onde o usuário pode fazer pesquisas, uma onde o usuário pode consultar as suas sugestões turísticas marcadas como ‘favoritas’ e uma com outras informações e opções como logout.

3 – O usuário faz uma consulta no aplicativo

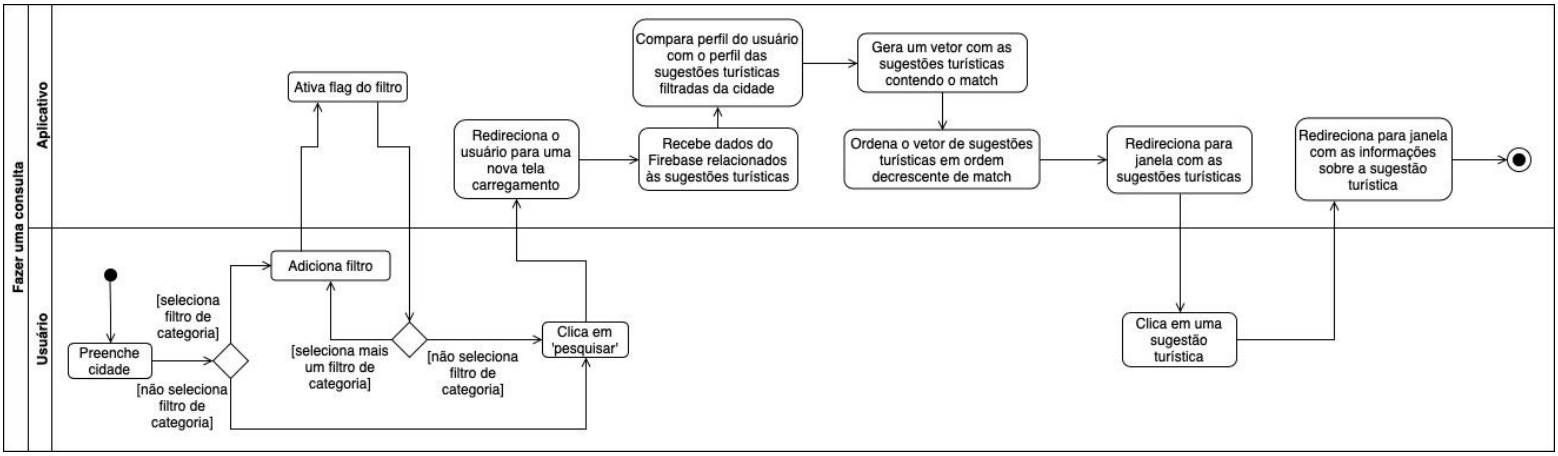


Figura 4: Diagrama de atividades – Efetuar consulta

- Para efetuar uma consulta, o usuário precisa escrever um nome de cidade. Caso não haja uma sujestão turística com a cidade escrita (isto se aplica aos casos de escrever o nome da cidade errado), o resultado da pesquisa é um aviso de que não foi encontrado nenhum resultado.
- Além disso, o usuário pode optar por clicar em botões de filtros por categoria, ativando uma ‘flag’ da categoria clicada.
- O aplicativo apresenta uma janela de carregamento.
- O aplicativo faz uma consulta no Firebase, buscando os dados das sugestões turísticas filtrados pela cidade e filtros e os transfere para um vetor de sugestões.
- O aplicativo percorre o vetor calculando o ‘match’ do usuário com cada sugestão turística e adiciona esta informação à um campo do vetor de sugestões.
- O vetor gerado é ordenado por ordem decrescente de ‘match’.
- O aplicativo apresenta uma janela onde cada sugestão turística é listada para o usuário e pode ser clicada, redirecionando-o para uma janela com as informações completas da sugestão: uma imagem, o nome, a cidade, a descrição completa e um link para o site oficial.

Comparação do perfil do usuário com o perfil das sugestões turísticas

Quando o usuário fizer uma pesquisa, o aplicativo irá buscar no Firebase as sugestões turísticas referentes a categoria e cidade informadas. Ao receber essas sugestões o programa irá calcular a compatibilidade entre o perfil de personalidade delas e o do usuário, e o resultado desta comparação chamamos de match.

Para realizar este cáculo, o aplicativo compara cada indicador de personalidade do usuário com o da sugestão turística. Para realizar esta comparação, é chamada a função compareIndicators, que os parâmetros, ‘user’ e ‘suggestion’ que representam, respectivamente, o indicador a ser comparado do usuário e da sugestão turística, e retorna o resultado desta comparação.

```
private func compareIndicators(user: Double, suggestion: Double) -> Double{

        var result: Double = 0

        if user > suggestion{
            result = 1 - (user - suggestion)
        }else{
            result = 1 - (suggestion - user)
        }
        return result
    }
```

Esta função primeiramente compara o valor do indicador do usuário e da sugestão turística para saber qual é maior. Com este resultado, ele diminui o número maior do menor, resultando na porção destes números que é diferente. E por fim ele diminui este valor de 1, resultando na porção destes números que é igual, tanto no quesito de possuir quanto não possuir esta personalidade. Observe a seguir uma imagem explificando.

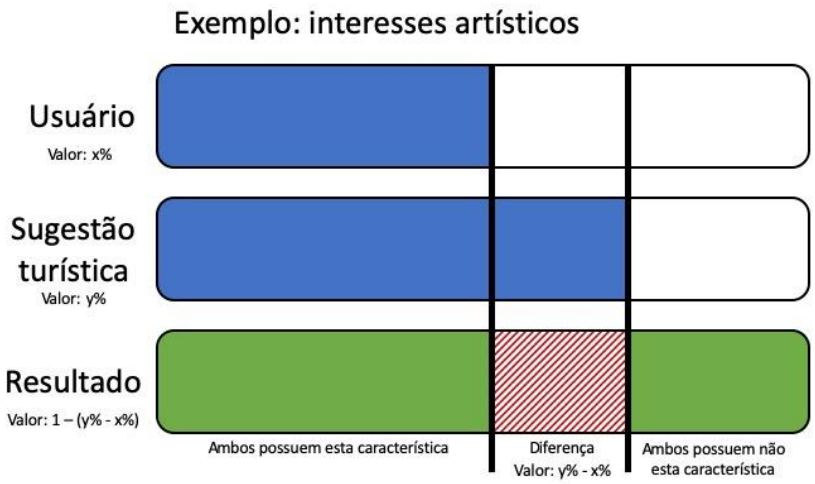


Figura 4 – Exemplo visual do cálculo do match

Os resultados das chamadas da função compareIndicators, relativos a todos os indicadores, são somados e divididos pelo número de indicadores de personalidade, resultando em uma média, que por fim é considerada como o 'match'.

Ordenando as sugestões turísticas de forma decrescente

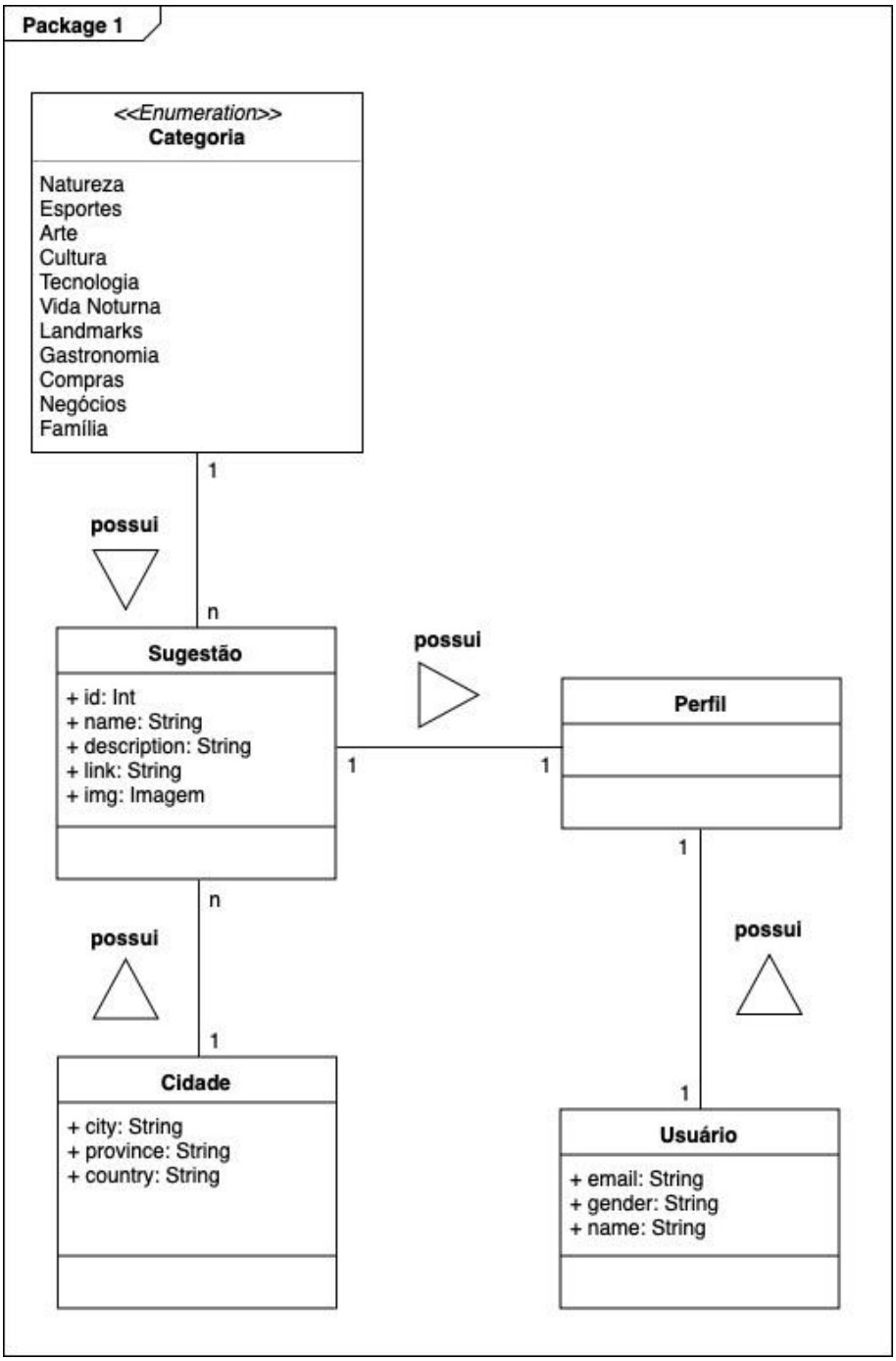
O programa recebe as informações do Firebase, depois calcula o 'match' e então armazena os dados no vetor de sugestões turísticas e, dessa forma, ele não é ordenado. Porém temos um requisito que pede as sugestões turísticas sejam mostradas de forma decrescente para o usuário.

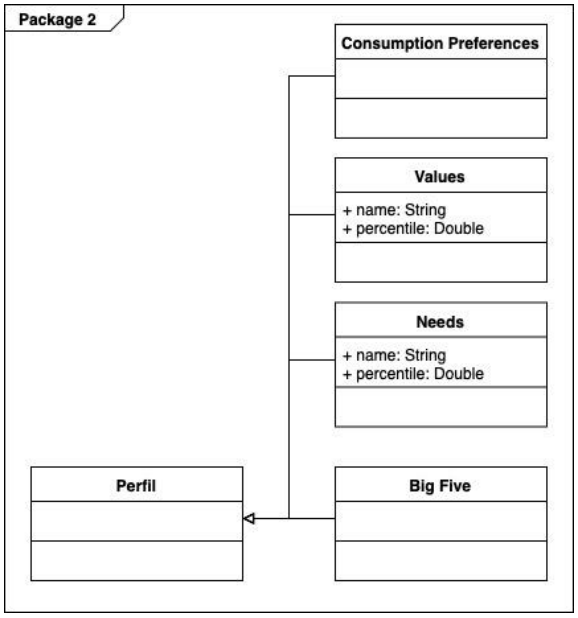
O objeto Array do Swift possui diversos métodos de ordenação, o que é bastante prático para poupar esforço de desenvolvimento. O método utilizado foi o Array.sorted(by:) que ordena o vetor em um dado elemento (que neste caso é a função que retorna o 'match'), com uma dada condição (por exemplo < ou >) e retorna o vetor ordenado. Abaixo está a linha com o código desta ordenação:

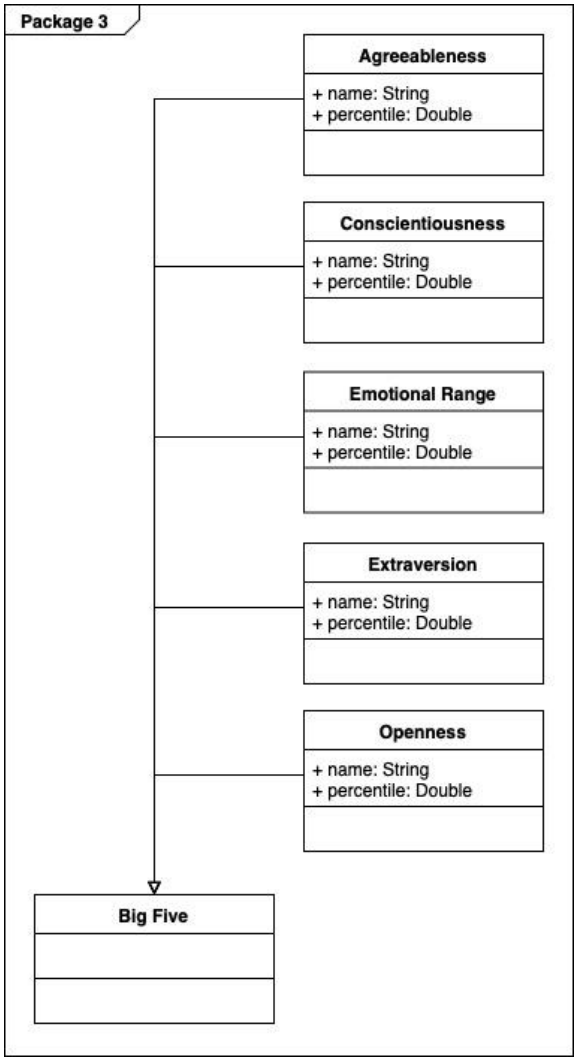
```
self.listOfSuggestions = self.listOfSuggestions.sorted(by: {$0.getMatch() > $1.getMatch()})
```

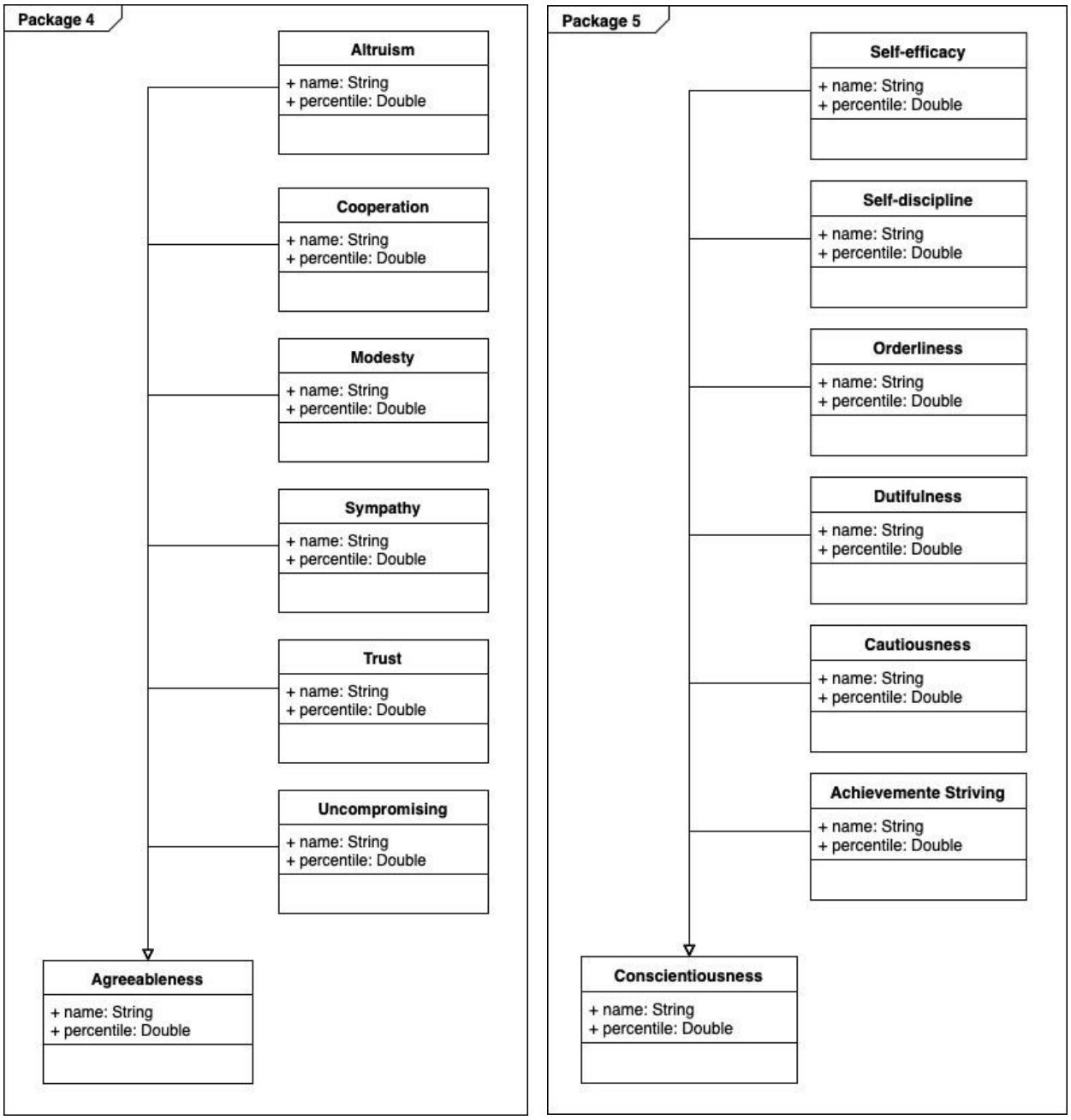
Este algoritmo possui complexidade O[n * log(n)] [3], onde n é o tamanho da entrada, ou seja, o número de sugestões turísticas. Além disso, a Apple não divulga muitos detalhes sobre o algoritmo, o que dificuldade uma análise mais profunda sobre a performance de sua utilização, como por exemplo o uso de memória e a estabilidade. Porém a complexidade é boa em comparação aos algoritmos de ordenação mais conhecidos, como o merge sort, quicksort, bubble sort e o heapsort.

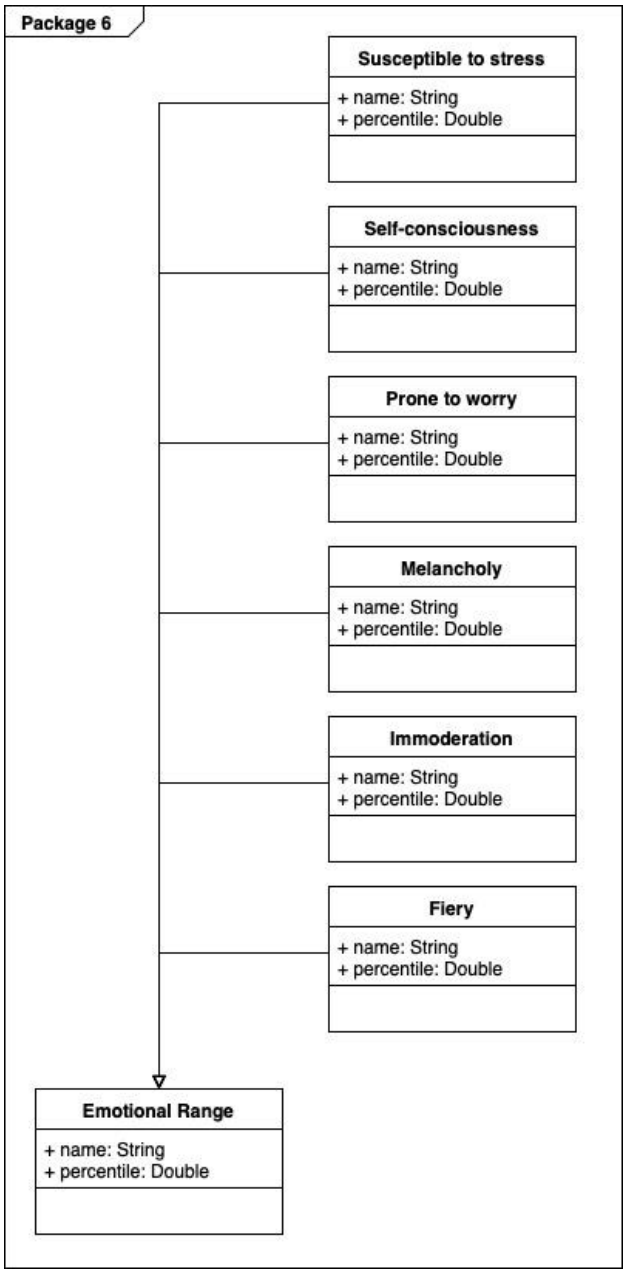
4. Diagrama de Classes

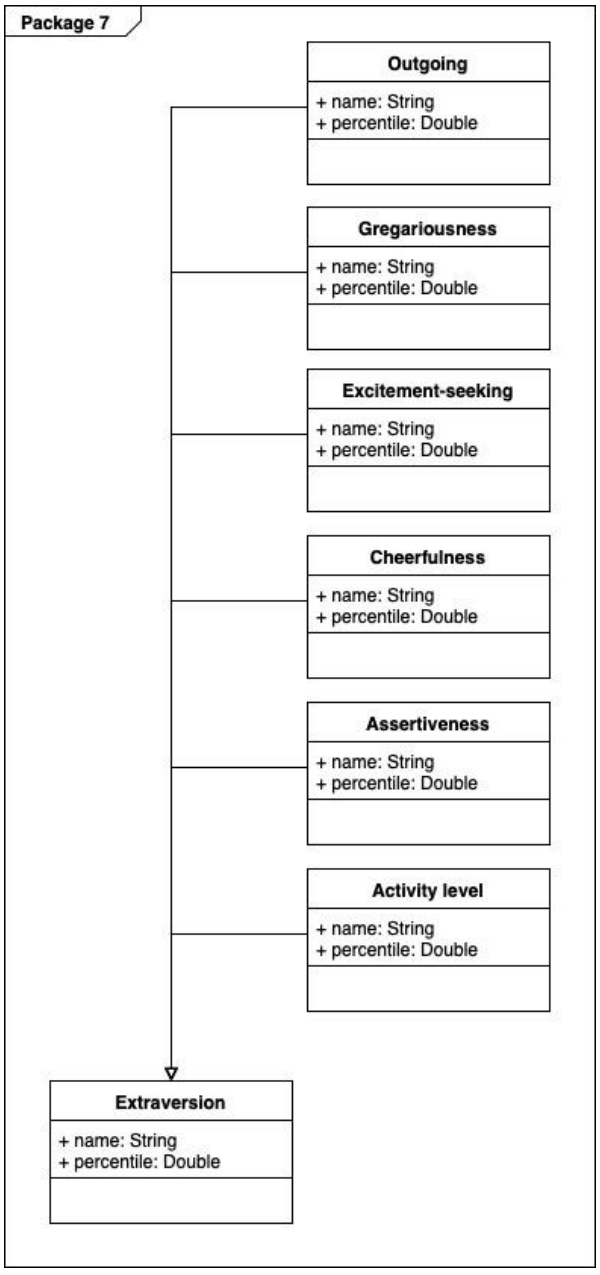


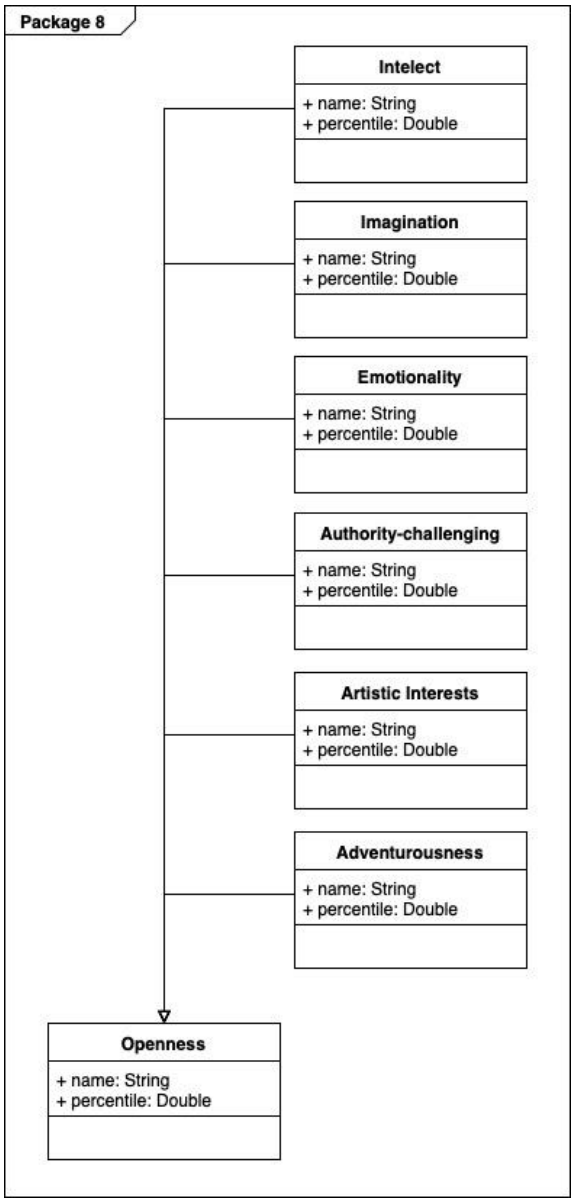

Package 2
Consumption Preferences
Values
+ name: String
+ percentile: Double
Needs
+ name: String
+ percentile: Double
Perfil
Big Five

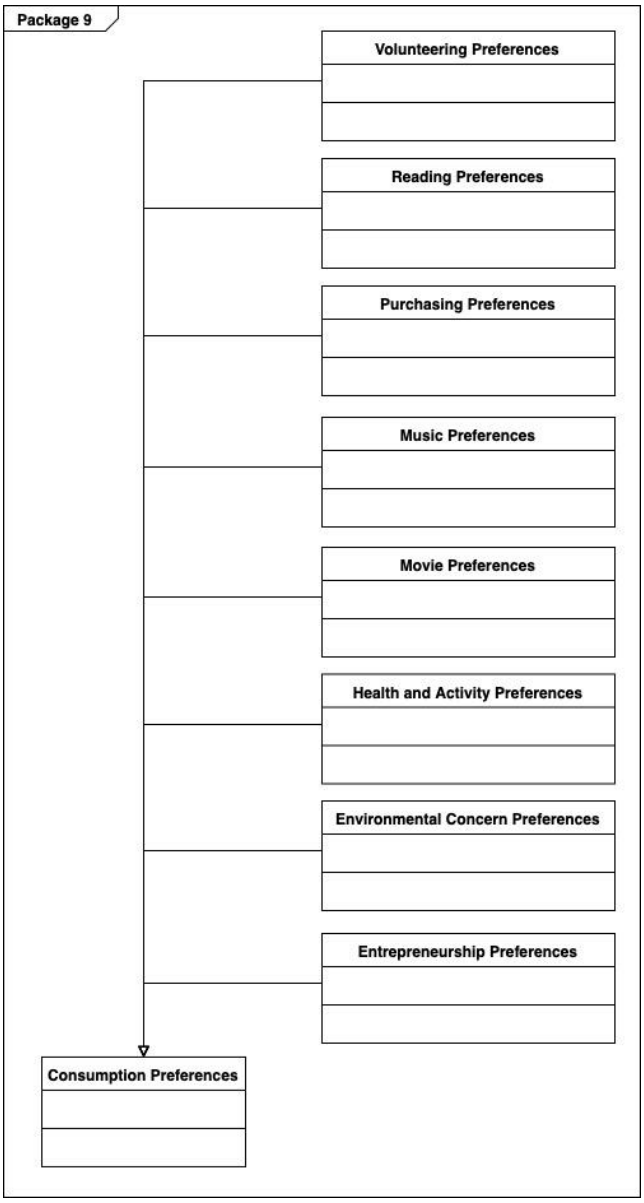

Package 3
Agreeableness
+ name: String
+ percentile: Double
Conscientiousness
+ name: String
+ percentile: Double
Emotional Range
+ name: String
+ percentile: Double
Extraversion
+ name: String
+ percentile: Double
Openness
+ name: String
+ percentile: Double
Big Five

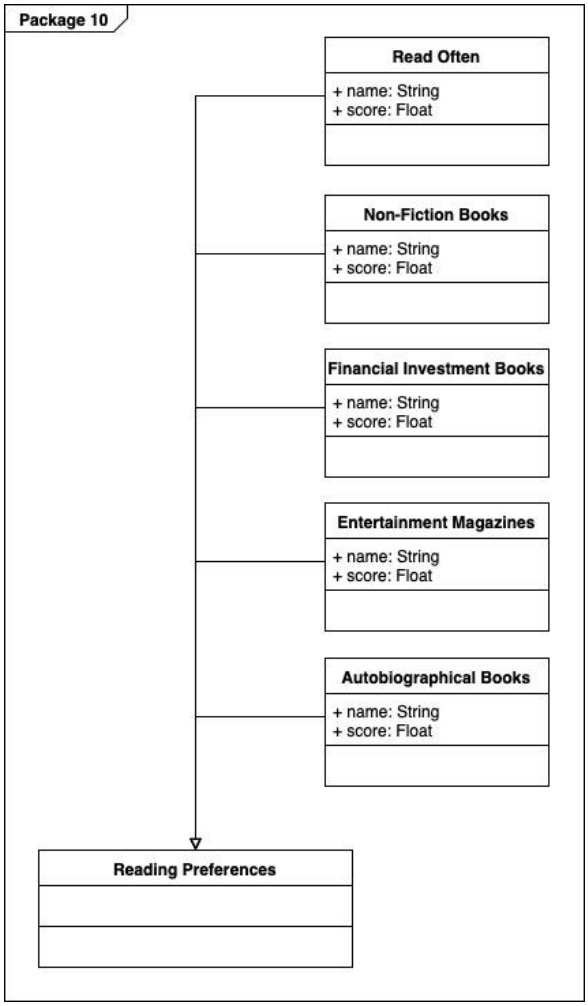

Package 4
Altruism
+ name: String
+ percentile: Double
Cooperation
+ name: String
+ percentile: Double
Modesty
+ name: String
+ percentile: Double
Sympathy
+ name: String
+ percentile: Double
Trust
+ name: String
+ percentile: Double
Uncompromising
+ name: String
+ percentile: Double
Agreeableness
+ name: String
+ percentile: Double
Package 5
Self-efficacy
+ name: String
+ percentile: Double
Self-discipline
+ name: String
+ percentile: Double
Orderliness
+ name: String
+ percentile: Double
Dutifulness
+ name: String
+ percentile: Double
Cautiousness
+ name: String
+ percentile: Double
Achievemente Striving
+ name: String
+ percentile: Double
Conscientiousness
+ name: String
+ percentile: Double

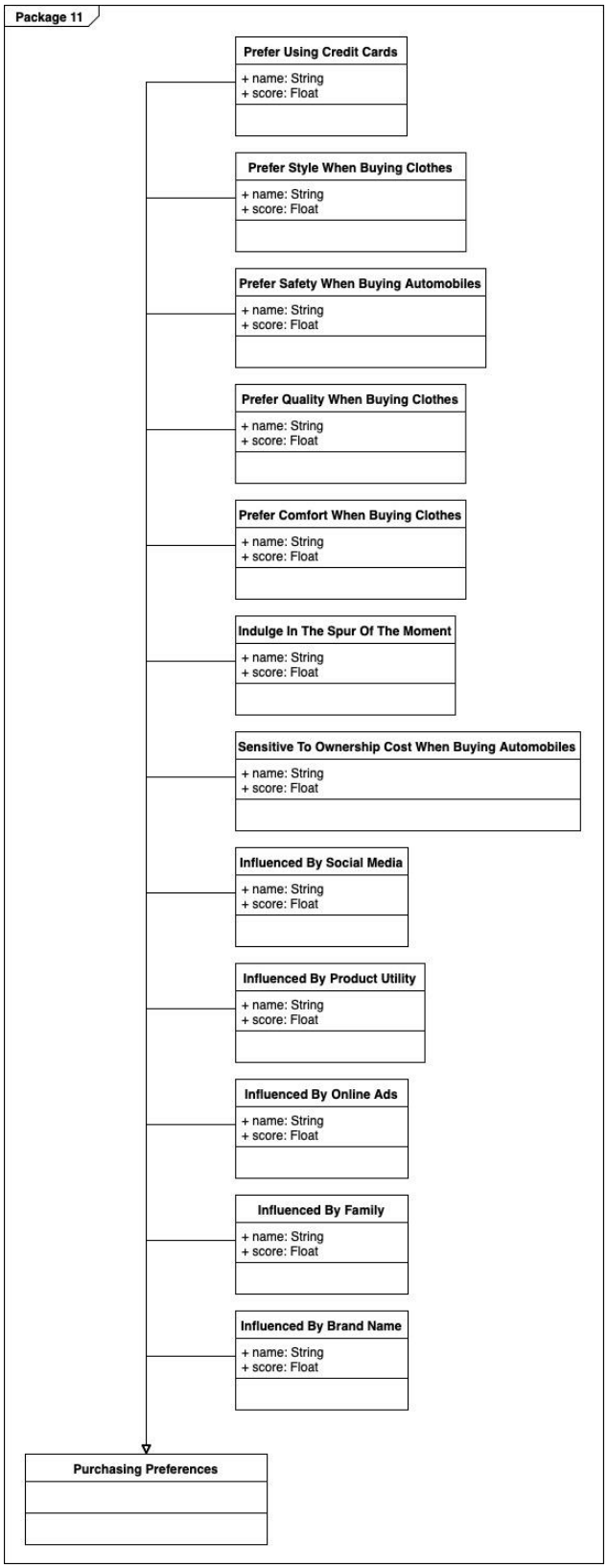

Package 6
Susceptible to stress
+ name: String
+ percentile: Double
Self-consciousness
+ name: String
+ percentile: Double
Prone to worry
+ name: String
+ percentile: Double
Melancholy
+ name: String
+ percentile: Double
Immoderation
+ name: String
+ percentile: Double
Fiery
+ name: String
+ percentile: Double
Emotional Range
+ name: String
+ percentile: Double

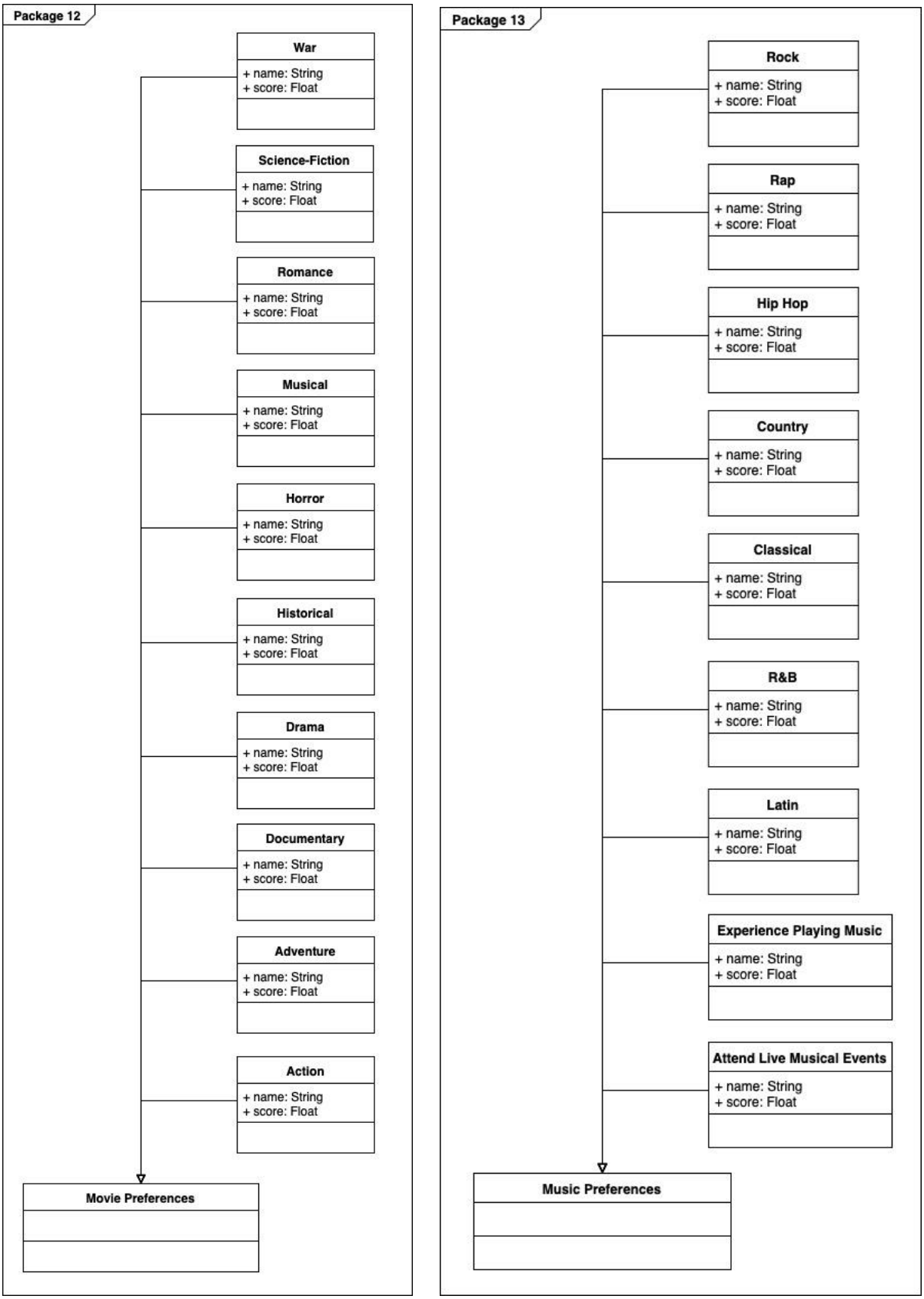

Package 7
Outgoing
+ name: String
+ percentile: Double
Gregariousness
+ name: String
+ percentile: Double
Excitement-seeking
+ name: String
+ percentile: Double
Cheerfulness
+ name: String
+ percentile: Double
Assertiveness
+ name: String
+ percentile: Double
Activity level
+ name: String
+ percentile: Double
Extraversion
+ name: String
+ percentile: Double

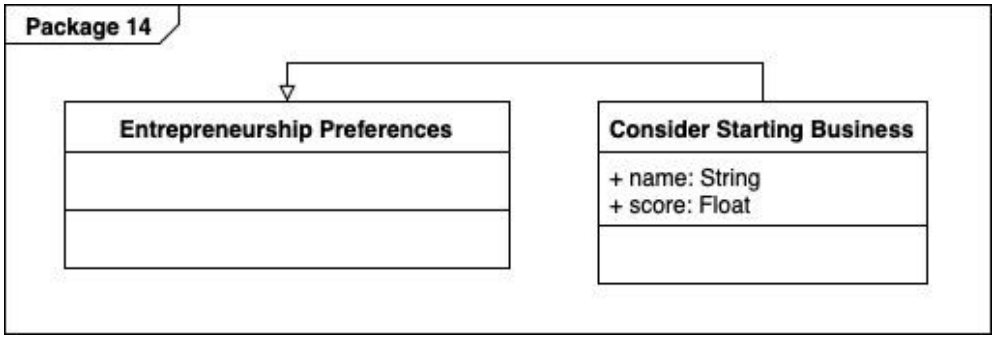

Package 8
Intelect
+ name: String
+ percentile: Double
Imagination
+ name: String
+ percentile: Double
Emotionality
+ name: String
+ percentile: Double
Authority-challenging
+ name: String
+ percentile: Double
Artistic Interests
+ name: String
+ percentile: Double
Adventurousness
+ name: String
+ percentile: Double
Openness
+ name: String
+ percentile: Double

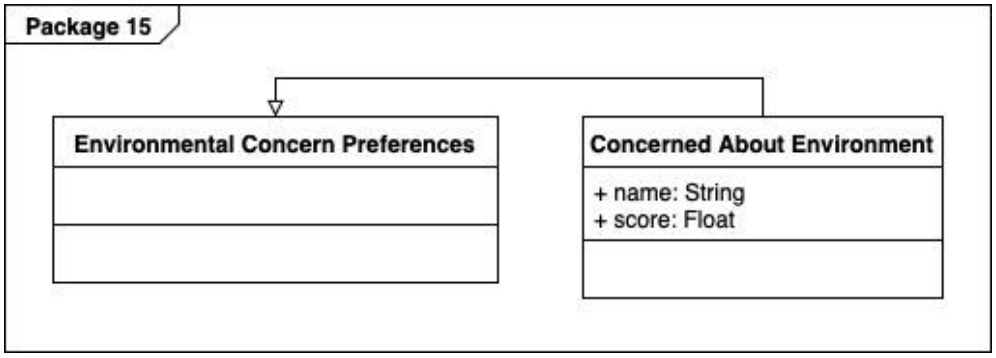

Package 9
Volunteering Preferences
Reading Preferences
Purchasing Preferences
Music Preferences
Movie Preferences
Health and Activity Preferences
Environmental Concern Preferences
Entrepreneurship Preferences
Consumption Preferences

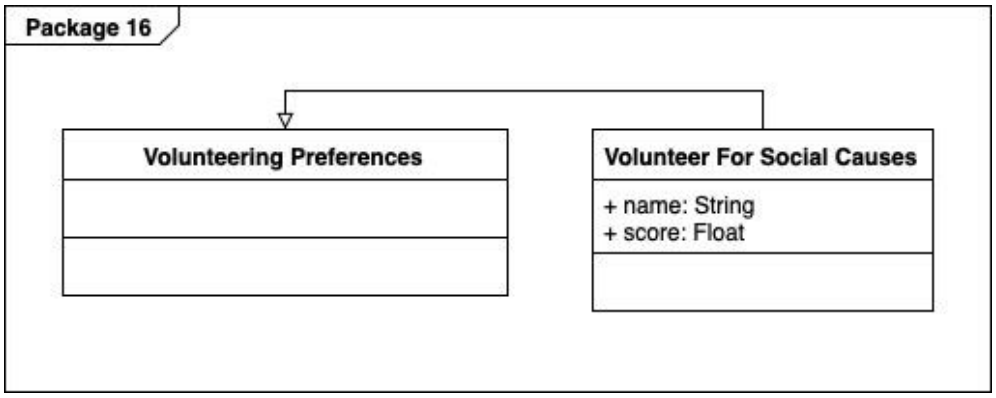

Package 10
Read Often
+ name: String
+ score: Float
Non-Fiction Books
+ name: String
+ score: Float
Financial Investment Books
+ name: String
+ score: Float
Entertainment Magazines
+ name: String
+ score: Float
Autobiographical Books
+ name: String
+ score: Float
Reading Preferences

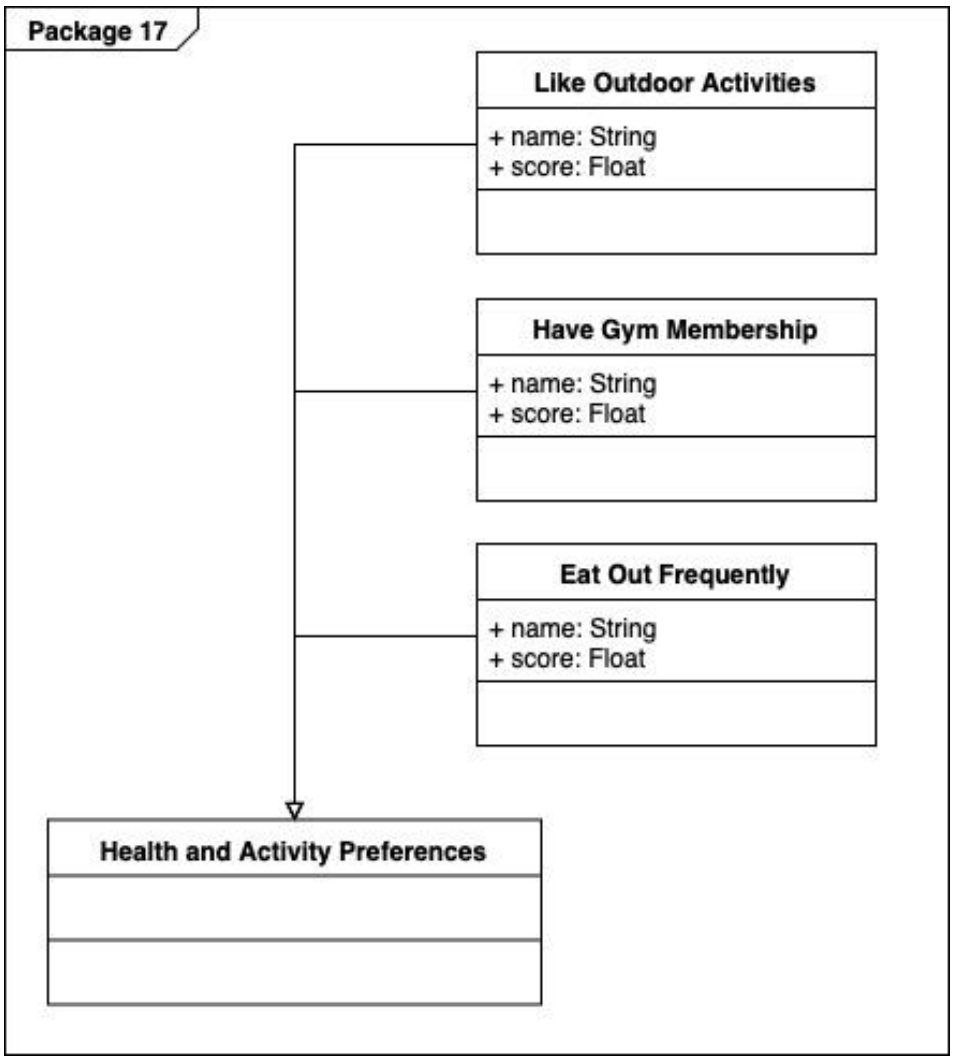

Package 11
Prefer Using Credit Cards
+ name: String
+ score: Float
Prefer Style When Buying Clothes
+ name: String
+ score: Float
Prefer Safety When Buying Automobiles
+ name: String
+ score: Float
Prefer Quality When Buying Clothes
+ name: String
+ score: Float
Prefer Comfort When Buying Clothes
+ name: String
+ score: Float
Indulge In The Spur Of The Moment
+ name: String
+ score: Float
Sensitive To Ownership Cost When Buying Automobiles
+ name: String
+ score: Float
Influenced By Social Media
+ name: String
+ score: Float
Influenced By Product Utility
+ name: String
+ score: Float
Influenced By Online Ads
+ name: String
+ score: Float
Influenced By Family
+ name: String
+ score: Float
Influenced By Brand Name
+ name: String
+ score: Float
Purchasing Preferences


Package 12
War
+ name: String
+ score: Float
Science-Fiction
+ name: String
+ score: Float
Romance
+ name: String
+ score: Float
Musical
+ name: String
+ score: Float
Horror
+ name: String
+ score: Float
Historical
+ name: String
+ score: Float
Drama
+ name: String
+ score: Float
Documentary
+ name: String
+ score: Float
Adventure
+ name: String
+ score: Float
Action
+ name: String
+ score: Float
Movie Preferences
Package 13
Rock
+ name: String
+ score: Float
Rap
+ name: String
+ score: Float
Hip Hop
+ name: String
+ score: Float
Country
+ name: String
+ score: Float
Classical
+ name: String
+ score: Float
R&B
+ name: String
+ score: Float
Latin
+ name: String
+ score: Float
Experience Playing Music
+ name: String
+ score: Float
Attend Live Musical Events
+ name: String
+ score: Float
Music Preferences


Package 14
Entrepreneurship Preferences
Consider Starting Business
+ name: String
+ score: Float


Package 15
Environmental Concern Preferences
Concerned About Environment
+ name: String
+ score: Float


Package 16
Volunteering Preferences
Volunteer For Social Causes
+ name: String
+ score: Float



Obs: a descrição completa dos indicadores “big five”, “needs”, “values” e “consumption preferences” estão na sessão 5.3.1.

5. Descrição e Utilização das APIs

Neste projeto são utilizadas as seguintes APIs: Watson Language Translator, Watson Personality Insights, Facebook e Firebase. Nesta sessão está descrita a relação entre cada uma dessas APIs e para cada uma delas temos uma explicação de como ela é aplicada no projeto.

5.1. Introdução ao Watson

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 [4].

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.

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. Um outro serviço do Watson é o Language Translator, que permite a tradução de texto entre diferentes línguas através da computação cognitiva, ou seja, a tradução não é apenas literária, mas também leva em conta termos, gírias e combinações de palavras, cujo significado foi aprendido por machine learning.

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. 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.

5.2. Watson Language Translator

Um dos primeiros problemas encontrados no estudo das APIs para este projeto, foi que o Watson Personality Insights não aceita textos de entrada na língua portuguesa. A primeira solução pensada foi fazer o programa totalmente em inglês, o que seria uma opção razoável apesar de dificultar os testes. Uma outra solução mais completa, seria fazer o projeto em português e utilizar uma ferramenta de tradução para traduzir as postagens do Facebook do usuário para inglês antes de repassá-las para o Watson Personality Insights. A opção escolhida para ser essa ferramenta de tradução foi o Watson Language Translator, pois já está incluído no pacote de serviços que este projeto utiliza e tem o poder de fazer uma tradução mais precisa como veremos mais para frente [5].

O Watson Language Translator é uma ferramenta poderosa de tradução que utiliza técnicas de deep learning para melhorar a velocidade e a precisão da tradução, o que ajuda a identifcar o signficado de termos, gírias e combinações de palavras. Todos os pares de idiomas utilizam a tradução da rede neural do Watson. O serviço ainda é capaz de identificar o idioma de origem do texto, o que é de grande ajuda para este projeto pois o aplicativo poderar ler um perfil do Facebook independente da língua escrita nas postagens.

O serviço garante a preservação da privacidade dos dados e funciona com os seguintes idiomas: Árabe, Catalão, Chinês (Simplificado e Tradicional), Tcheco, Dinamarquês, Holandês, Inglês, Finlandês, Francês, Alemão, Hindi, Húngaro, Italiano, Japonês, Coreano, Norueguês, Polonês, Português (Brasil), Russo, Espanhol, Espanhol, Sueco e Turco. Existe também uma característica interessante, que não possui aplicação neste projeto, que possibilita a tradução de documentos preservando a formatação do mesmo e pode ser utilizado em arquivos Microsoft Office, Open Office, PDF, HTML, JSON, TXT e XML.

Aplicação do serviço no projeto:

A partir do momento em que a API do Facebook enviar para o aplicativo as postagens do usuário, o Language Translator irá traduzir cada postagem separadamente antes do aplicativo repassar as postagens para o JSON que será lido pelo Personality Insights.

5.3. Watson Personality Insights

O Personality Insights é uma API do Watson que efetua uma análise linguística para inferir as características de personalidade dos indivíduos, incluindo necessidades, valores, preferências de consumos e características como agradabilidade e capacidade receptiva, a partir de comunicações digitais, como e-mail, blogs, tweets e postagens no fórum. Com isso podemos observar as inclinações do usuário para buscar produtos, serviços e diferentes atividades, como compras, músicas e filmes.

Este serviço utiliza como entrada um ou mais textos, que neste projeto são postagens do Facebook para os usuários e críticas do site TripAdvisor com nota 4 ou 5 para as sugestões turísticas. As linguagens de entrada são: Árabe, Inglês, Japonês, Coreano e Espanhol. Portanto utilizamos a API Language Translator para traduzir as postagens do Facebook e críticas do TripAdvisor de portugês para inglês quando necessário[1].

Ao analisar os textos, o Personality Insights retorna um perfil de personalidade através de um objeto *"Profile",* que é um objeto JSON que será detalhado posteriormente. Mas para que este retorno seja válido, é necessário que haja uma quantidade mínima de palavras na chamada. É possível enviar o serviço com até 20 MB de conteúdo de entrada. Mas a precisão se estabiliza em aproximadamente 3000 palavras de entrada e adicionar mais conteúdo não contribui para a precisão do perfil. Portanto, o serviço extrai e usa apenas os primeiros 250 KB de conteúdo, sem contar qualquer marcação de HTML ou JSON, de solicitações 'grandes'[1].

- 3000 palavras são suficientes para alcançar a precisão máxima do serviço.
- Pelo menos 1200 palavras resulta em um erro médio absoluto que está na margem de dois por cento do melhor erro médio absoluto que o serviço pode retornar.
- Entre 600 e 1.200 palavras resulta em um erro médio absoluto que está na margem de três por cento do melhor erro médio absoluto que o serviço pode retornar.
- Menos de 600 palavras gera um aviso, mas o serviço ainda analisa a entrada.
- Menos de 100 palavras gera um erro.

Outro detalhe importante sobre o JSON "profile" é que todos indicadores de personalidade, são automaticamente normalizados pelo próprio Personality Insights. Para obter os valores não-normalizados, deve ser especificado "true" para o parâmetro de consulta "raw_scores" da solicitação.

Desta forma, além dos valores normalizados o seviço relatará o valor bruto para cada característica da personalidade. Os valores brutos representam a pontuação para uma característica que é baseada exclusivamente no texto do autor e na computação de valores pelo Watson para essa característica, sem comparar com uma população de amostra. Valores brutos podem ser interpretados como os valores que o autor receberia se fizesse um teste de personalidade. O serviço disponibiliza valores brutos para usuários que desejam aplicar uma normalização customizada para um cenário específico ou que não requerem uma comparação com uma população de amostra.

5.3.1. Descrição dos indicadores 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” [6].
- **Needs:** indica quais aspectos de um produto podem combinar com uma pessoa [6].
- **Values:** indica fatores de motivação que influenciam a tomada de decisão de uma pessoa [6].
- **Consumption Preferences:** 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 [6].

1. **Big Five:**
   - 1.1. **Agreeableness:** mede a tendência de uma pessoa a ser compassiva e compreensiva com outras pessoas. Em Português, seria agradabilidad. Possui os seguintes facets:
     - a. **Altruism / Altruistic:** ajudar o próximo está mais perto de ser uma realização pessoal do que ser um sacrifício. Português: altruísmo.
     - b. **Cooperation / Accommodating / Compilance:** 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 autoestima. Português: modesto.
     - d. **Morality / Uncompromising / Sincerity:** não são pretensiosos 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.

1.2. Conscientiousness: mede a tendência de uma pessoa a agir de forma planejada e organizada. Português: conscienciosidade. Possui os seguintes facets:

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 responsability:** 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.

1.3. Extraversion: mede a tendência de uma pessoa a buscar estímulo na companhia de outras pessoas. Português: extrovertido. Possui os seguintes facets:

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. **Gargariousness / Sociable:** acham que a companhia dos outros é agradavelmente estimulante e gratificante. Tendem a gostar da emoção das multidões. Português: sociável.

1.4. Emotional Range: também conhecida como neuroticismo, é a medida em que uma pessoa é sensível às emoções negativas do ambiente. Possui os seguintes facets:

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. Sentem-se envergonhados e inconfortáveis 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 a uma situação emergencial.

1.5. Openess: mede a tendência de uma pessoa estar aberta a novas experiências. Português: capacidade receptiva. Possui os seguintes facets:

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.

2. 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 ânsia 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.

3. **Values:**
   a. **Self-transcedence / Helping others:** demonstram preocupação com o bem-estar e os interesses dos outros. Português: auto transcendência.
   b. **Conservation / Tradition:** enfatizam a autorrestrição e ordem. Resistem às 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.

4. **Consumption Preferences:**
   4.1. Compras:
      a. **consumption_preferences_automobile_ownership_cost:** consideram o gasto total da aquisição de automóveis antes de concluir uma escolha.
      b. **consumption_preferences_automobile_safety:** consideram os fatores de segurança para comprar automóveis.
      c. **consumption_preferences_clothes_quality:** preferem qualidade ao comprar roupas.
      d. **consumption_preferences_clothes_style:** preferem estilo e moda ao comprar roupas.
      e. **consumption_preferences_clothes_comfort:** preferem conforto ao comprar roupas.
      f. **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 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.

4.2. 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.

4.3. 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.

4.4. 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.

4.5. 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 prática de esporte.
c. **consumption_preferences_outdoor:** tendem a gostar de atividades ao ar livre.

4.6. Empreendedorismo:

a. **consumption_preferences_start_business:** podem considerar começar um novo negócio nos próximos anos.

4.7. Meio ambiente:

a. **consumption_preferences_concerned_environment:** demonstram interesse e preocupação sobre questões ecológicas.

4.8. Voluntariado:

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

5.3.2. Estrutura do JSON

O Watson Personality Insights, ao ler um texto ou conjunto de textos, retorna um objeto JSON contendo os indicadores de personalidade[1]. Este objeto se chama *Profile* e 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 descrevem 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.
- **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.

Exemplo:

```
{
  "word_count": 15223,
  "processed_language": "en",
  "personality": [
     . . .
  ],
  "needs": [
     . . .
  ],
  "values": [
     . . .
  ],
  "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 contem a pontuação bruta da característica. Este campo é opcional, só retorna caso o parâmetro raw_scores seja true na chamada do método getProfile().
- **siginificant:** 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

Exemplo:

```
{
  . . .
  "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
    },
    . . .
  ],
  . . .
}
```


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 contem a pontuação que indica a probabilidade do usuário preferir tal item.

Exemplo:

```
{
  . . .
  "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
        },
        . . .
      ]
    },
    . . .
}
```

5.3.3. Interpretando os indicadores de personalidade

Neste ponto já estudamos o objeto Profile completo e sabemos que ele detalha resultados em objetos do tipo Trait (personalidade: Big Five e facet e ConsumptionPreferences (preferências de consumo). Todos estes objetos detalham características da personalidade do usuário com indicadores numéricos do tipo double[1].

a. Percentil de características de personalidade (*percentile*):

Para cada requerimento, o serviço fornece uma pontuação através do atribulo *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. 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.

5.4. Facebook

A Graph API do Facebook é a principal maneira de obter dados dentro e fora da plataforma do Facebook [7]. É uma API baseada em HTTP que os aplicativos podem usar para consultar dados programaticamente, publicar novas histórias, gerenciar anúncios, fazer upload de fotos e realizar uma grande variedade de outras tarefas. Ela fornece para este projeto duas funcionalidades: login e requisição de dados do usuário, que utilizadas em conjunto, nos permitem obter os dados dos usuários suficientes para diferenciá-los no banco de dados da aplicação e para fornecer um serviço mais personalizado.

Ao fazer o requerimento do login, o aplicativo passa um vetor de pedido de permissão de leitura. Neste vetor passamos os parametros:

- publicProfile: permite o aplicativo acessar informações públicas e básicas do usuário, sendo elas nome completo e foto de perfil.
- email: permite o aplicativo acessar o e-mail do usuário.
- gender: permite o aplicativo acessar o gênero do usuário.
- userPosts: permite o aplicativo acessar todas as postagens/publicações feitas pelo usuário na sua linha do tempo.

Após o usuário fazer o requerimento do login, o aplicativo redireciona o usuário para uma página do Facebook, onde ele deve forner as credenciais de acesso (usuário e senha). Em caso de sucesso, esta página do Facebook adiverte o usuário sobre a utilização dos seus dados e ele pode cancelar ou concordar em cede-los. Concordando, o Facebook o redireciona de volta para o aplicativo e este faz um requerimento de conexão ‘GraphRequestConnection’ e então de fato busca as informações ditas anteriormente e transfere para o Firebase.

5.5. Firebase

O banco de dados utilizado neste projeto é o Cloud Firestore, que é um NoSQL hospedado na nuvem que os apps do iOS, do Android e da Web podem acessar diretamente por meio de SDKs nativos [8]. Foi escolhido por ser gratuito e por ter sido considerado como tendo funções simples, de fácil entendimento e configuração.

O armazenamento dos dados é feito em documentos (*'documents'*) que contêm mapeamentos de campos (*'fields'*) para valores. Esses documentos são armazenados em coleções (*'collections'*), que são contêineres de documentos que você pode usar para organizar dados e criar consultas. Cada documento pode conter mais coleções, ou seja, subcoleções, criando estruturas de dados hierárquicas que podem ser escalonadas à medida que o banco de dados cresce.

Neste projeto, existem duas coleções: 'users' e 'suggestions'. Ambas possuem documentos referenciados pelo id do Facebook, que são números únicos gerados pela API do Facebook. Na primeira coleção, cada documento representa um usuário que já utilizou o aplicativo pelo menos uma vez, efetuando o login com o Facebook. Já na segunda coleção, cada documento representa uma sugestão turística, cadastrada por um administrador.

Os documentos da coleção 'users', possuem os seguintes campos:

Id: número identificador do usuário, retirado do Facebook. Chave primária.

Nome: nome do usuário, retirado do Facebook.

Gênero: gênero do usuário, retirado do Facebook.

E-mail: e-mail do usuário, retirado do Facebook.

Indicadores de personalidade: todos os indicadores gerados pelo Personality Insights (Big Five, Needs, Values e Consumption Preferences).

Os documentos da coleção 'suggestions', possuem os seguintes campos ('fields'):

Id: número identificador da sugestão turística, retirado do Facebook. Chave primária.

Nome: nome da sugestão turística, adicionado manualmente pelo administrador, utilizando um software de apoio.

Cidade: cidade onde a sugestão turística se encontra, adicionado manualmente pelo administrador, utilizando um software de apoio.

Categoria: categoria da sugestão turística, adicionado manualmente pelo administrador, utilizando um software de apoio. Só pode ser uma das seguintes opções:

- Natureza. *Exemplos: parques nacionais, fazendas, grutas, praias.*
- Esportes. *Exemplos: eventos esportivos, ginásios, estádios, trilhas, pontos de práticas esportivas.*
- Arte. *Exemplos: peças teatrais, concertos musicais.*
- Cultura. *Exemplos: museus, pontos históricos.*
- Tecnologia. *Exemplos: eventos de jogos ou tecnologia, empresas.*
- Vida Noturna. *Exemplos: bares, pubs, boates, feiras noturnas, festas.*
- Landmarks. *Exemplos: Torre Eiffel, Cristo Redentor, Estátua da Liberdade, Muralha da China.*
- Gastronomia. *Exemplos: restaurantes, cafés, feiras gastronômicas.*
- Compras. *Exemplos: shopping centers, bazares, feiras.*
- Negócios. *Exemplos: workshops, conferências, eventos de finanças, marketing, etc.*
- Família. *Exemplos: parques de diversão.*

Link: link para o site da sugestão turística, adicionado manualmente pelo administrador, utilizando um software de apoio.

Imagem: link para a imagem principal da sugestão turística, adicionado manualmente pelo administrador, utilizando um software de apoio.

Descrição: breve descrição da sugestão turística, adicionado manualmente pelo administrador, utilizando um software de apoio.

Indicadores de personalidade: todos os indicadores gerados pelo Personality Insights (Big Five, Needs, Values e Consumption Preferences).

6. Casos de Uso e Especificação dos Testes a Serem Realizados

6.1. Casos de uso, telas e casos de teste

- Efetuar login:

Caso de Uso:

Código:	CDU1
Objetivo:	Efetuar login
Atores:	• Usuário
Pré-condições:	• O usuário possui uma conta no Facebook
Trigger:	Usuário aperta botão de login
Fluxo principal:	1. Aplicativo pede permissão para usar o Facebook para realizar login 2. Aplicativo redireciona para tela de login do Facebook 3. Usuário preenche as credenciais de login 4. Aplicativo pede confirmação de que o usuário deseja realizar o login 5. Aplicativo redireciona para tela de carregamento 6. Aplicativo recebe as informações do usuário 7. As postagens recebidas do Facebook são traduzidas 8. As postagens são convertidas em um perfil de personalidade 9. Os dados do usuário e o perfil de personalidade são persistidos no Firebase. 10. Aplicativo redireciona o usuário para a janela principal, na guia de pesquisa.
Fluxo alternativo:	2a. Usuário não da permissão 1. Aplicativo apresenta janela com botão de login. 4a. Credenciais inválidas 1. Aplicativo exibe mensagem de credenciais inválidas 2. O usuário continua o fluxo principal a partir do passo 3 4b. Usuário clica em “cancelar

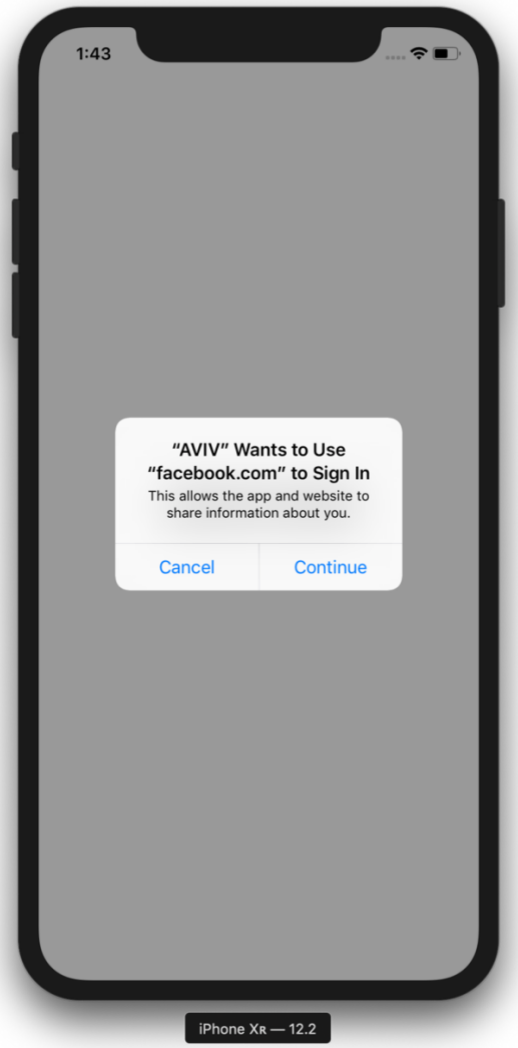
	1. Aplicativo apresenta janela com botão de login 6a. Falha de comunicação com o Facebook 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 7a. Falha na comunicação com o Watson Language Translator 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 8a. Falha na comunicação com o Watson Personality Insights 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 8b. Usuário não possui a quantidade suficiente de palavras nas postagens para gerar um perfil de personalidade 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 9a. Falha na comunicação com o Firebase 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login

Casos de teste:

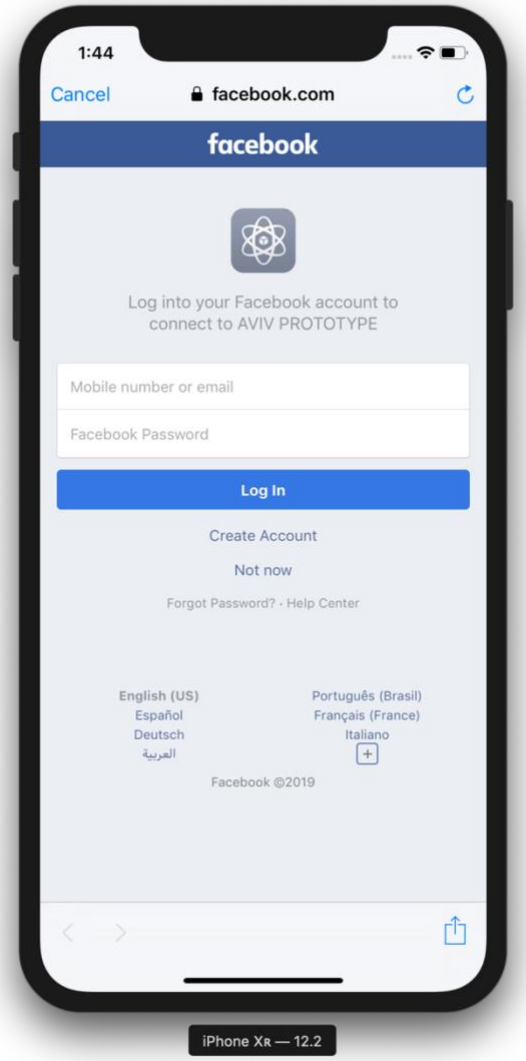
Elemento	Input / Ação / Interferência	Resposta esperada
CDU1 – 1	Cancel	CDU1 – 2a.1
CDU1 – 3	Usuário e senha válidos	CDU1 – 4
CDU1 – 3	Usuário ou senha inválidos	CDU1 – 4a.1
CDU1 – 3	Cancel	CDU1 – 4b.1
CDU1 – 6	Conexão com a internet interrompida	CDU1 – 6a.1
CDU1 – 7	Conexão com a internet interrompida	CDU1 – 7a.1
CDU1 – 8	Conexão com a internet interrompida	CDU1 – 8a.1
CDU1 – 8	Perfil do Facebook com menos de 300 palavras em suas postagens	CDU1 – 8b.1
CDU1 – 9	Conexão com a internet interrompida	CDU1 – 9a.1
CDU1 – 10	Perfil do Facebook com mais de 300 palavras em suas postagens e conexão com a internet estável	CDU1 – 10a.1

Capturas de tela:

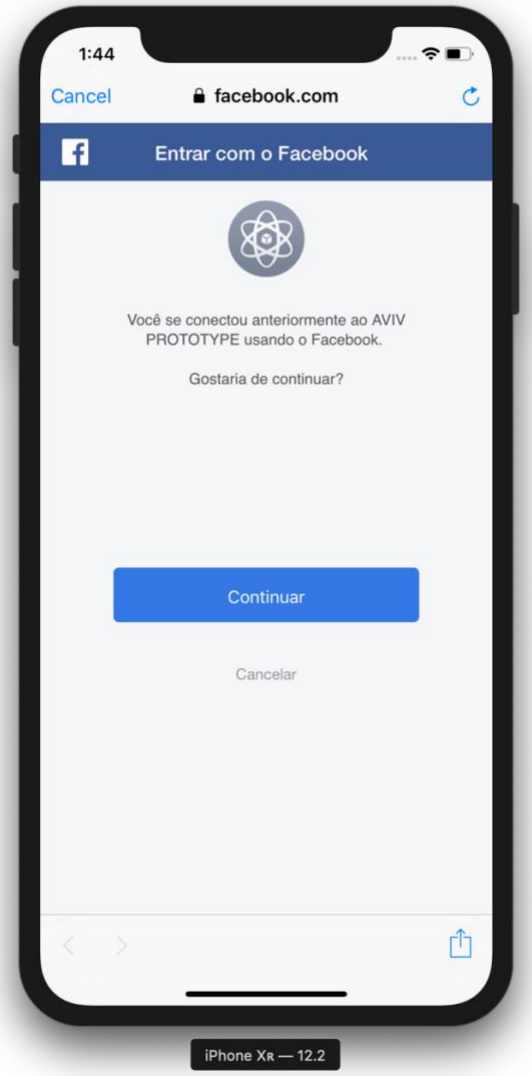
Captura 1 – Login



Captura 2 – Login



Captura 3 – Login



Captura 4 – Login

Captura 5 – Login

- Fazer pesquisa

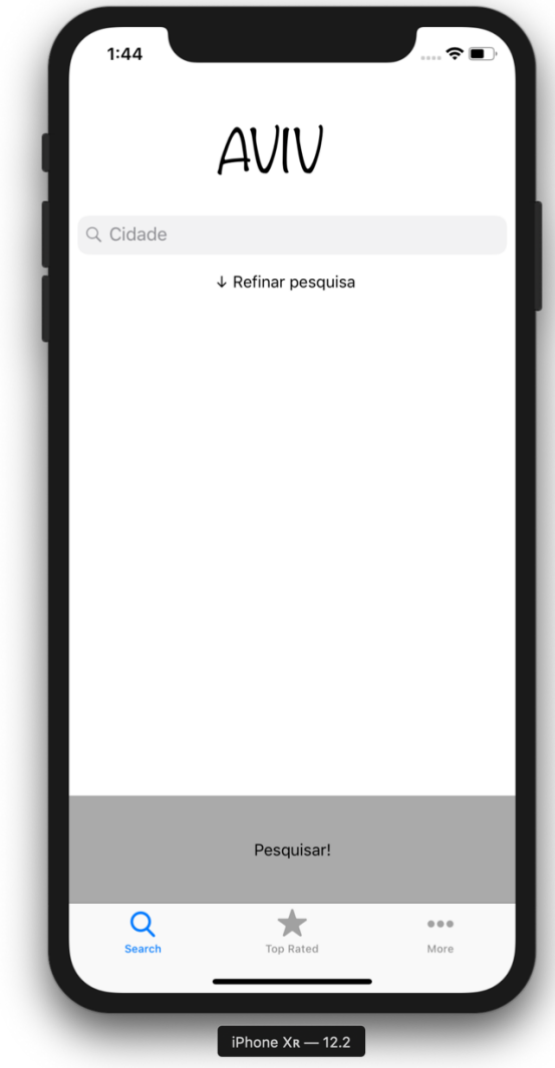
Caso de uso:

Código:	CDU2
Objetivo:	Fazer pesquisa
Atores:	• Usuário
Pré-condições:	• Caso de teste CDU1 finalizado com sucesso
Fluxo principal:	1. Usuário preenche cidade 2. Usuário clica no botão “Pesquisar!” 3. Aplicativo faz a busca no Firebase com os dados fornecidos 4. O aplicativo compara o perfil de personalidade de cada resultado com o do usuário, gerando um match 5. O aplicativo apresenta os resultado em ordem decrescente de match 6. O usuário seleciona um resultado 7. O aplicativo apresenta em uma nova janela o resultado selecionado com mais informações
Fluxo alternativo:	2a. Usuário clica no botão “Refinar pesquisa” 1. Usuário seleciona 0 ou mais filtros 2. O usuário continua o fluxo principal a partir do passo 2 3a. Não foi encontrado nenhum resultado 1. O aplicativo apresenta uma mensagem avisando que nenhum resultado foi encontrado 2. O usuário clica em “Ok” 3. O aplicativo redireciona o usuário para o passo 1 do fluxo principal 3b. Falha na comunicação com o Firebase 1. O aplicativo apresenta uma mensagem avisando que nenhum resultado foi encontrado 2. O usuário clica em “Ok” 3. O aplicativo redireciona o usuário para o passo 1 do fluxo principal

Casos de teste:

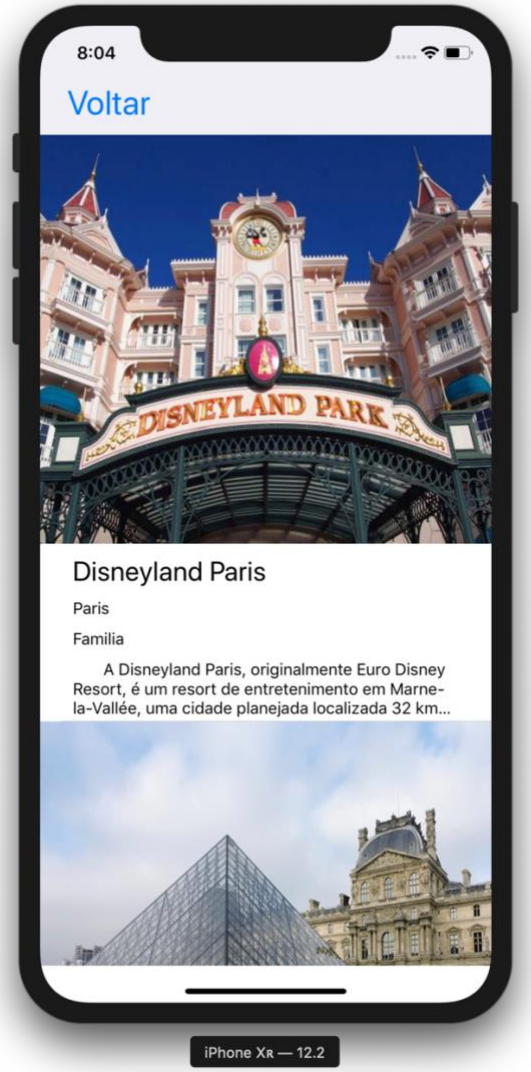
Elemento	Input / Ação / Interferência	Resposta esperada
CDU2 – 2	Cidade válida	CDU2 – 5
CDU2 – 2	Cidade inválida (escrito errado ou não cadastrada)	CDU2 – 3a.1
CDU2 – 2a	Selecionar 1 ou mais filtros / Encontra	CDU2 – 5
CDU2 – 2a	Selecionar 1 ou mais filtros / Não encontra	CDU2 – 3a.1
CDU2 – 3	Conexão com a internet interrompida	CDU2 – 3b.1
CDU2 – 6	Seleciona um item qualquer da lista	CDU2 - 7

Captura de tela:

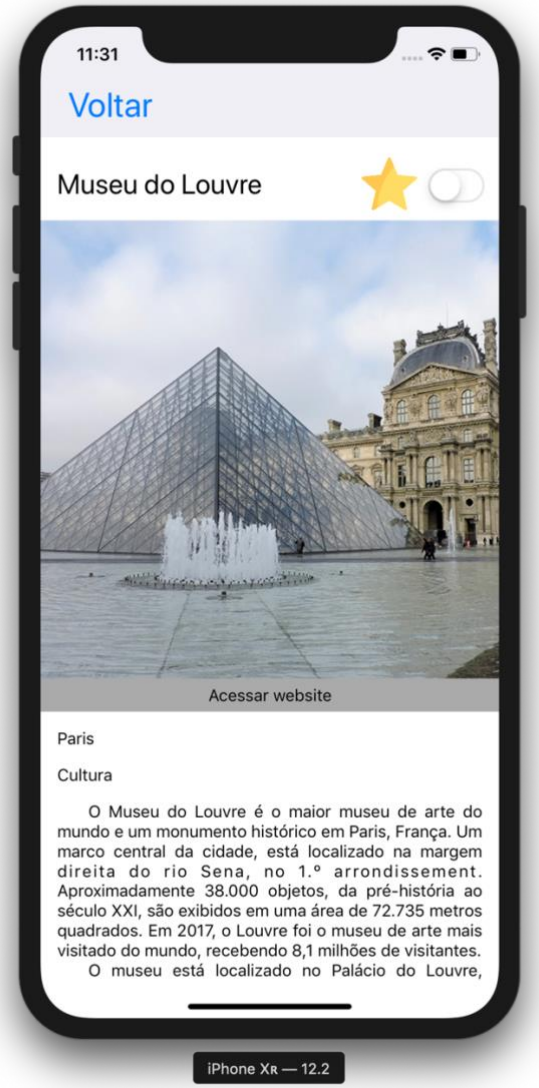


Captura 1 – Pesquisa

Captura 2 – Pesquisa



Captura 3 – Pesquisa



Captura 4 – Pesquisa

- Adicionar sugestão turística ao Firebase

Caso de uso:

Código:	CDU3
Objetivo:	Adicionar sugestão turística ao Firebase
Atores:	• Administrador
Pré-condições:	• Possuir um usuário de teste do Facebook para a sugestão turística a ser cadastrada associado à conta de desenvolvedor do Facebook utilizada pelo administrador do aplicativo
Fluxo principal:	1. Administrador preenche todos os campos 2. Aplicativo redireciona para tela de login do Facebook 3. Administrador preenche as credenciais de login 4. Aplicativo pede confirmação de que o administrador deseja realizar o login 5. Aplicativo redireciona para tela de carregamento 6. Aplicativo recebe as informações da sugestão turística 7. As postagens recebidas do Facebook são traduzidas 8. As postagens são convertidas em um perfil de personalidade 9. Os dados da sugestão turística e o perfil de personalidade são persistidos no Firebase. 10. Aplicativo redireciona o adiminstrador para a tela de conclusão 11. Administrador clica em “Concluir” 12. Aplicativo realiza log out.
Fluxo alternativo:	1a. Algum campo não foi preenchido 1. Aplicativo apresenta uma mensagem avisando 2. O administrador continua o fluxo principal a partir do passo 1 novamente. 2a. Administrador não da permissão 1. Aplicativo apresenta janela com botão de login. 4a. Credenciais inválidas 1. Aplicativo exibe mensagem de credenciais inválidas 2. O administrador continua o fluxo principal a partir do passo 3 4b. Administrador clica em “cancelar 1. Aplicativo apresenta janela com botão de login

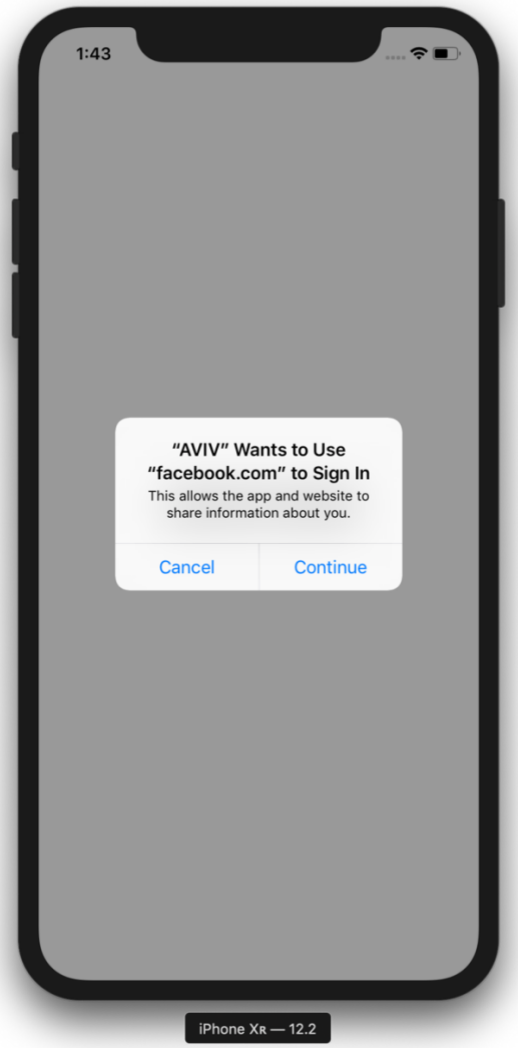
	6a. Falha de comunicação com o Facebook 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 7a. Falha na comunicação com o Watson Language Translator 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 8a. Falha na comunicação com o Watson Personality Insights 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 8b. Sugestão turística não possui a quantidade suficiente de palavras nas postagens para gerar um perfil de personalidade 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login 9a. Falha na comunicação com o Firebase 1. Aplicativo apresenta mensagem de erro 2. Aplicativo apresenta janela com botão de login

Casos de teste:

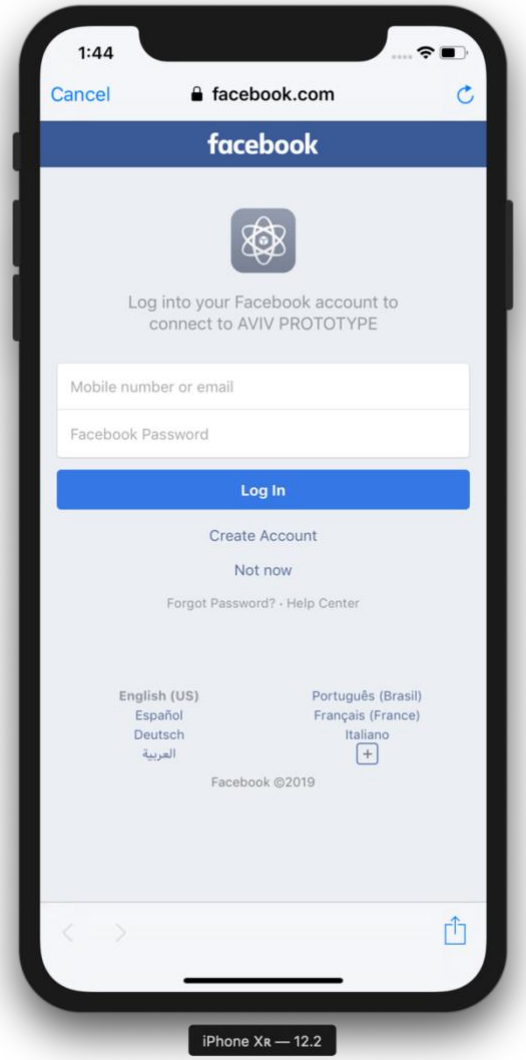
Elemento	Input / Ação / Interferência	Resposta esperada
CDU3 – 1	Todos os dados preenchidos	CDU3 – 2
CDU3 – 1	Algum dado em branco	CDU3 – 1a.1
CDU3 – 3	Usuário e senha válidos	CDU3 – 4
CDU3 – 3	Usuário ou senha inválidos	CDU3 – 4a.1
CDU3 – 3	Cancel	CDU3 – 4b.1
CDU3 – 6	Conexão com a internet interrompida	CDU3 – 6a.1
CDU3 – 7	Conexão com a internet interrompida	CDU3 – 7a.1
CDU3 – 8	Conexão com a internet interrompida	CDU3 – 8a.1
CDU3 – 8	Perfil do Facebook com menos de 300 palavras em suas postagens	CDU3 – 8b.1
CDU3 – 9	Conexão com a internet interrompida	CDU3 – 9a.1
CDU3 – 10	Perfil do Facebook com mais de 300 palavras em suas postagens e conexão com a internet estável	CDU3 – 10a.1

Captura de tela:

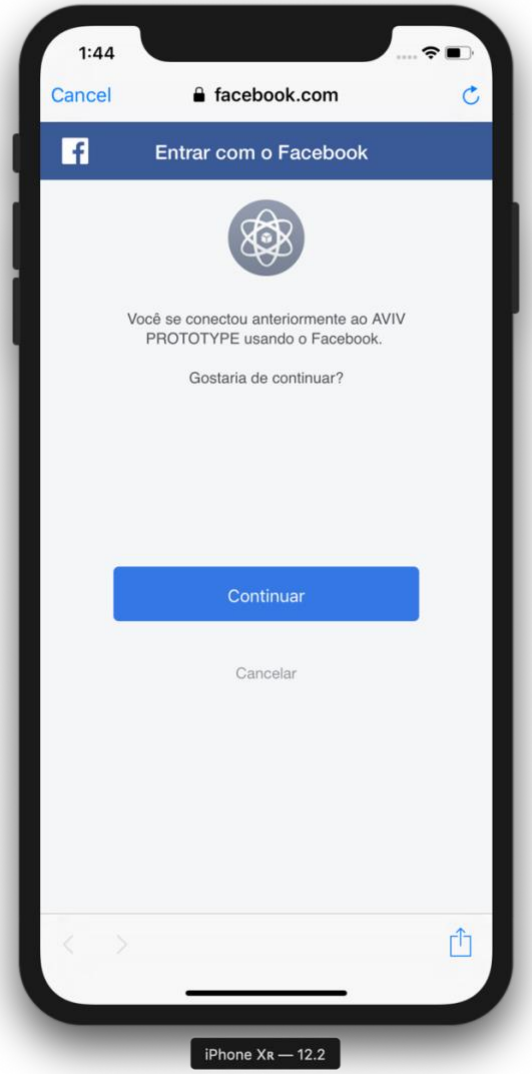
Captura 1 – Cadastro



Captura 2 – Cadastro



Captura 3 – Cadastro



Captura 4 – Cadastro

Captura 5 – Cadastro

6.2. Descrições dos perfis de teste utizados no projeto

Perfis de indicações para viagens:

Nome: Roland Garros
Email: perfil_atvqfuy_garros@tfbnw.net
Cidade: Paris Categoria: Esportes
Link para o site: https://www.rolandgarros.com/en-us/ Link para a imagem: https://firebasestorage.googleapis.com/v0/b/aviv-prototype.appspot.com/o/roland_garros.jpg?alt=media&token=4c6f5af5-1040-43a3-9d30-5a0eec514b1f
Descrição: O Torneio de Roland Garros (Internationaux de France, French Open ou Aberto da França) é um torneio de tênis realizado em Paris, na França. Tem seu nome em homenagem a Roland Garros, francês pioneiro da aviação. Com o Australian Open, o Torneio de Wimbledon e o US Open, o torneio de Roland Garros compõe os quatro torneios do Grand Slam de tênis. É disputado em quadra de saibro, em melhor de 5 sets para os homens e 3 sets para as mulheres. Em 1968, o torneio de Roland Garros foi o primeiro torneio do Grand Slam a ser "aberto", permitindo a participação tanto de amadores como de profissionais. Fonte: https://pt.wikipedia.org/wiki/Torneio_de_Roland_Garros

Nome: Torre Eiffel
Email: perfil_orxfzue_eiffel@tfbnw.net
Cidade: Paris Categoria: Landmarks
Link para o site: https://www.toureiffel.paris/en Link para a imagem: https://firebasestorage.googleapis.com/v0/b/aviv-prototype.appspot.com/o/Torre%20Eiffel.jpg?alt=media&token=df13cf2d-163f-48c8-b9ba-9524f3415fad
Descrição: A Torre Eiffel é uma torre treliça de ferro do século XIX localizada no Champ de Mars, em Paris, a qual se tornou um ícone mundial da França. A torre, que é o edifício mais alto da cidade, é o monumento pago mais visitado do mundo, com milhões de pessoas frequentando-o anualmente. Nomeada em homenagem ao seu projetista, o engenheiro Gustave Eiffel, foi construída como o arco de entrada da Exposição Universal de 1889. Possui 324 metros de altura e fica cerca de 15 centímetros mais alta no verão, devido à dilatação térmica do ferro. Foi a estrutura mais alta do mundo desde a sua conclusão até 1930, quando perdeu o posto para o Chrysler Building, em Nova Iorque, Estados Unidos. Não incluindo as antenas de transmissão, é a segunda estrutura mais alta do país, atrás apenas do Viaduto de Millau, concluído em 2004. A torre tem três níveis para os visitantes. Os ingressos podem ser adquiridos nas escadas ou elevadores do primeiro e do segundo nível. A caminhada para o primeiro nível é superior a 300 degraus. O terceiro e mais alto nível só é acessível por elevador. Do primeiro andar vê-se a cidade inteira, tendo o andar sanitários e várias lojas, e o segundo nível tem um restaurante. A torre tornou-se o símbolo mais proeminente de Paris e da França, sendo parte de cenários de filmes que se passam na cidade. Seu estatuto de ícone é tão determinado que ainda serve como um símbolo para todo o país, como quando a torre foi usada como o logotipo da candidatura francesa para sediar os Jogos Olímpicos de Verão de 1992. Fonte: https://pt.wikipedia.org/wiki/Torre_Eiffel

Nome: Museu do Louvre
Email: perfil_rshzcws_louvre@tfbnw.net
Cidade: Paris Categoria: Cultura
Link para o site: https://www.louvre.fr/en Link para a imagem: https://firebasestorage.googleapis.com/v0/b/aviv-prototype.appspot.com/o/museu_louvre.jpg?alt=media&token=32aa7b5b-5f50-4a2e-be65-b54d02b8fb3f
Descrição: O Museu do Louvre é o maior museu de arte do mundo e um monumento histórico em Paris, França. Um marco central da cidade, está localizado na margem direita do rio Sena, no 1.º arrondissement. Aproximadamente 38.000 objetos, da pré-história ao século XXI, são exibidos em uma área de 72.735 metros quadrados. Em 2017, o Louvre foi o museu de arte mais visitado do mundo, recebendo 8,1 milhões de visitantes. O museu está localizado no Palácio do Louvre, originalmente construído como uma fortaleza no final do século XII ao XIII sob o reinado de Filipe II. Os restos da fortaleza são visíveis no porão do museu. Devido à expansão urbana da cidade, a fortaleza acabou por perder sua função defensiva e, em 1546, foi convertida por Francisco I na residência principal dos reis franceses. O prédio foi ampliado muitas vezes até formar o atual Palácio do Louvre. Em 1682, Luís XIV escolheu o Palácio de Versalhes como sua residência, deixando o Louvre principalmente como um lugar para exibir a coleção real, incluindo, a partir de 1692, uma coleção de esculturas antiga grega e romana. Em 1692, o edifício foi ocupado pela Académie des Inscriptions et Belles-Lettres e pela Academia Real de Pintura e Escultura. A Académie permaneceu no Louvre por 100 anos. Durante a Revolução Francesa, a Assembleia Nacional Constituinte decretou que o Louvre deveria ser usado como um museu para exibir as obras-primas da nação. Fonte: https://pt.wikipedia.org/wiki/Museu_do_Louvre

Nome: L'Entrecôte de Paris
Email: perfil_hzwppkx_de_paris@tfbnw.net
Cidade: Paris Categoria: Gastronomia
Link para o site: https://www.lentrecotedeparis.fr/ Link para a imagem: https://firebasestorage.googleapis.com/v0/b/aviv-prototype.appspot.com/o/lentrecote_paris.jpg?alt=media&token=2bf1ec38-c25a-438a-aee3-0f07032f3f1d
Descrição: A França tem duas famosas fórmulas secretas. Uma é a poção mágica do druida. Panoramix, que dava força sobre-humana aos irredutíveis gauleses Asterix e Obelix. A outra é a receita do molho que guarnece o famoso entrecôte dos célebres restaurantes, conhecidos por todos os parisienses. No que estavam cobertos de razão: com um prato tão bom assim no cardápio, quem precisa de outros? Por mais de 50 anos, o suculento entrecôte fatiado, servido com fritas e recoberto com o molho secreto, tem feito a fama destes restaurantes. O L'Entrecôte de Paris é um restaurante com prato único. Isto mesmo, apenas um único prato é servido todos os dias, semanas, meses, anos em nossos restaurantes. Este conceito, já bastante apreciado e difundido na Europa e no mundo, foi introduzido no Brasil no fim de 2009. O Entrecôte é um prato simples. Ou melhor, um prato simplesmente perfeito: um corte especial de carne, preparado de acordo com a sua preferência, acompanhado por deliciosas batatas fritas à vontade e por nosso grande segredo: um molho especial, de receita guardada a sete chaves, que leva vinte e um ingredientes e mais de trinta e seis horas para ser preparado. Uma perfeita combinação de sabores e aromas. Nosso prato principal é servido em quatro versões: Executive, Classique, Exclusif et Les Enfants. Consultar disponibilidade na unidade de sua escolha. Fonte: https://www.lentrecotedeparis.com.br/lentrecote/

Nome: Le Procope
Email: perfil_bgtujlp_le_procope@tfbnw.net
Cidade: Paris Categoria: Gastronomia
Link para o site: https://www.procope.com/en/ Link para a imagem: https://firebasestorage.googleapis.com/v0/b/aviv-prototype.appspot.com/o/le_procope.jpg?alt=media&token=1976cd30-ca8e-45a5-b20c-8cc40cb1936b
Descrição: O café Procope, que existe desde 1686, foi o ponto de encontro de grandes nomes das artes como La Fontaine, Voltaire, Rousseau, Beaumarchais, Balzac, Hugo, Verlain. Eles eram visitantes assíduos do restaurante. Durante o século 18, ele foi casa de ideias liberais e, também, a história da Enciclopédia. Os encontros de Robespierre, Danton e Marat para discutir sobre a Revolução eram feitos por lá. Até Bonaparte deixou um chapéu no Le Procope que fica em exposição na entrada do restaurante à esquerda. Este estabelecimento conta com opções de lanche, almoço e jantar. Lanche a partir das 11:30 da manhã tem opções com preço acessível a partir de 7 euros que engloba café expresso, suco de laranja e um salgado. O almoço tem um combo a partir de EUR 20,70 (entrada+ prato principal ou prato principal + sobremesa) que é servido do meio dia até as 8 horas da noite e o jantar conta com um menu a la carte e um combo dos filósofos que fica EUR 38,00. A carta de vinhos também é bastante variada: de vinhos acessíveis até vinhos muito, muito, caros. Quando o pedido é feito, é possível perguntar ao garçom qual é o vinho que ele recomenda para acompanhar o prato e eles dão dicas certeiras para cada ocasião. Fonte: https://www.precisoviajar.com/le-procope-historia-e-boa-comida-em-paris/

Nome: Disneyland Paris
Email: perfil_umjcgdj_paris@tfbnw.net
Cidade: Paris Categoria: Família
Link para o site: https://www.disneylandparis.com Link para a imagem: https://firebasestorage.googleapis.com/v0/b/aviv-prototype.appspot.com/o/disney_paris.jpg?alt=media&token=a57c996c-1585-461e-9518-18da2a911365
Descrição: A Disneyland Paris, originalmente Euro Disney Resort, é um resort de entretenimento em Marne-la-Vallée, uma cidade planejada localizada 32 km a leste do centro de Paris, sendo a atração mais visitada em toda a França e Europa. Sua proprietária e administradora é a Euro Disney S.C.A. uma empresa de capital aberto na qual a The Walt Disney Company possui uma participação minoritária.[2] O resort cobre uma área de 19 km² e compreende dois parques temáticos, alguns hotéis resort, um complexo de compras, alimentação e entretenimento e um campo de golfe, além de alguns locais de entretenimento. O Disneyland Park é o parque temático original do complexo e foi aberto com o resort em 12 de abril de 1992. Um segundo parque temático, o Walt Disney Studios Park, foi aberto em 2002. O parque é o segundo mais visitado parque da Disney em número de visitantes, com 15,6 milhões de pessoas em 2013, colocando-o entre a Disneyland de Anaheim na Califórnia e o Walt Disney World na Flórida, Estados Unidos. O resort é o segundo parque da Disney a abrir fora dos Estados Unidos, após o Tokyo Disney Resort, e o primeiro a ser operado pela Disney (através da Euro Disney S.C.A.). O resort foi projetado especificamente para seguir o modelo estabelecido pelo Walt Disney World na Flórida. Fonte: https://pt.wikipedia.org/wiki/Disneyland_Resort_Paris

7. Comentários sobre o Projeto e a Implementação

O projeto começou com o objetivo de obter aprendizado e prática em desenvolvimento de aplicativos e ainda não havia conhecimento sobre a linguagem Swift e sobre as APIs do Watson, Facebook e Firebase. Desta forma, como o conhecimento sobre as APIs foi sendo obtido progressivamente, os requisitos mudaram conforme a evolução do entendimento da funcionalidade efetivamente disponível pelos componentes citados.

Além disso, para chegar no 'produto final', foram pensados em 5 principais funcionalidades a serem implementadas, sendo elas:

1) Login e leitura dos dados do Facebook.
2) Interpretação das postagens do Facebook pelo Personality Insights, gerando um perfil.
3) Adição das sugestões turísticas ao banco de dados.
4) Comparação dos perfis do usuário com a das sugestões turísticas.
5) Apresentação do resultado para o usuário.

Devido à dependência entre essas funcionalidades, houve muito retrabalho pelo fato de métodos não funcionarem conforme o documentado e/ou esperado.

A seguir estão descritas algumas dificuldades de cada uma dessas funcionalidades:

1) O grande desafio deste primeiro passo foi entender a API do Facebook. A documentação disponível era difícil de entender e parecia estar desatualizada. Muitos exemplos de iOS ainda estavam em Objective-C ou utilizavam versões antigas da API. Além disso, não havia muita discussão sobre a API em Swift no Google, ou em foruns, Stack Overflow, e outras comunidades.
2) A API do Watson é muito bem explicada e foi relatimente 'fácil' entender o Personality Insights. A única dificuldade que aconteceu neste tópico foi que o Personality Insights não faz a leitura de textos na língua portuguesa, o que me fez ter que escolher um entre dois caminhos: fazer um aplicativo que só compatível com perfis do Facebook de usuários que postam em inglês ou acoplar uma API de tradução linguística. A primeira opção traz mais desafios de testes, a segunda em relação a implementação. Acabei optando pela segunda, utlizando uma API do próprio Watson chamada Language Translator.
3) O ideal teria sido utilizar um webcrawler para ler um website de críticas relacionadas a viagens, enviando as críticas positivas para o Personality Insights. Porém, por questões de complexidade e o tempo disponível na disciplina de Projeto Final, isto não seria factível

neste momento. Então decidi realizar a “população da base de dados” de forma “manual”, adaptando o código similar que já estava pronto (para a geração do perfil do usuário). Desta forma, criei um aplicativo de ‘apoio’, já apresentado neste documento, feito para ser utilizado por administradores, para popular o banco de dados. Neste aplicativo, o administrador preenche os campos com dados da sugestão turística e faz login em um perfil de teste do Facebook criado para esta sugestão com postagens retiradas do site de críticas TripAdvisor.

4) Nesta etapa houve um grande desafio por conta do Firebase. Para fazer a comparação dos perfis, é necessário pegar todos indicadores de personalidade, tanto das sugestões turísticas quanto do usuário no banco de dados. O Firestore funciona como uma árvore, onde a raiz é uma *‘collection’* de ‘*documents’*, e cada *‘document’* contem campos com valores ou outra *‘collection’*. O problema é que para fazer a leitura de cada *‘document’*, é necessário abrir uma thread assíncrona e não é possível fazer a leitura de um *‘document’* ao mesmo tempo que outro *‘document’* mesmo que ele esteja dentro de uma *‘collection’* sua. Como a estrutura dos dados estava respeitando o diagrama de classes projetado, ele possuía quase 95 *‘documents’*, ou seja, para o usuário e cada sugestão turística, o Firebase abria uma thread assíncrona para cada ‘document’. Foi perdido muito tempo tentando sincronizar essas threads, até que decidi que o melhor a ser feito seria alterar o que já tinha sido feito, e cheguei à seguinte solução:
    - Não utilizar a base de dados do perfil de personalidade do usuário no Firebase, passando a utilizar apenas os dados locais.
    - Retirar todas as *‘subcollections’* (*‘collections’* dentro de *‘documents’*) das sugestões turísticas e passá-las para o *‘document’* principal.

   Esta solução deixou os dados ‘mais desorganizados’, diferente do que foi modelado, e custou muito retrabalho. Mas passamos a trabalhar com apenas uma thread assíncrona e isto não apenas facilitou a programação e também diminuiu o tempo de resposta dos servirços do Firebase e os riscos de bugs.
5) Esta parte é a que o usuário escolhe a cidade e os filtros por categoria e depois ao clicar no botão para efetuar a busca, o software apresenta uma tableView com as sugestões, que quando clicadas redirecionam o usuário para uma janela onde ele pode ver a sugestão mais detalhadamente.

   O primeiro problema grave encontrado foi que o Firestore não possui alguns operadores básicos de banco de dados, como “and”, “or”, “in”, “contains” e “like”. Vale lembrar que este serviço ainda está em fase beta de teste. Um dos administradores

registrou em uma “issue” criada no GitHub do Firebase que estes operadores estão sendo estudados para possivelmente serem implementados em futuros updates.

Desta forma foi feito o filtro na busca pela cidade, mas não foi possível fazer pelos filtros de categoria direto na query. Então esses filtros foram aplicados no resultado da busca, já no próprio aplicativo, o que não é ideal, já que vai exigir mais processamento por parte do dispositivo do usuário.

Um outro desafio da implementação foi que, durante o projeto houveram diversas atualizações nas ferramentas utilizadas. A primeira foi a atualização do Swift 4 para o 4.2, que causou várias incompatibilidades entre o código que já estava escrito e as bibliotecas do Watson e do Facebook, resultando em ter que reescrever completamente o programa, que nesta época estava por volta de 50% pronto. Mas no final foi muito positivo, já que me fez revisar todo o código, fazendo melhorias.

Outra atualização foi a do Swift 5 e Xcode 10.2. Nesta a biblioteca do Facebook começou a apresentar diversos bugs e o aplicativo quebrava nas chamadas de função de login. Pesquisando as causas, cheguei a conclusão de que era uma incompatibilidade da versão atual do Xcode e do Swift com a biblioteca que estava utilizando do Facebook por ela estar em versão beta. O Facebook possui duas APIs que estão sendo utilizadas neste projeto, a “facebook-swift-sdk” e a “facebook-objc-sdk”, onde a primeira está em versão beta. Porém, toda documentação e tutoriais são baseados na primeira API, enquanto na segunda não. Este problema foi resolvido reescrevendo o código do login com a segunda biblioteca. Isto causou um pouco de retrabalho, mas o impacto maior foi no tempo que levou para descobrir a causa.

8. Conclusão

Este projeto me deu a oportunidade de conhecer uma linguagem de programação nova, Swift, que diferente das que estou acostumado, sofre frequentes atualizações que interferem no código. Além disso lidei com diversas APIs, algumas em versão beta e algumas com fraca documentação. Isto me me fez ter uma pequena experiência de como a programação e o desenvolvimento de software pode se tornar desafiador no ambito profissional.

Outro desafio ainda a ser superado será encontrar uma solução para a fonte melhor para ser interpretada pelo Watson Personality Insights. Muitos usuários fazem poucas postagens, e nesses casos o número de palavras pode ser insuficiente para fazer um perfil de personalidade, acarretando erro ou perfis de personalidade com baixa acurácia. Além disso, é importante ressaltar que na tradução das postagens do Facebook pelo Watson Language Translator pode existir erros. Estes erros vão ser considerados no Watson Personality Insights, diminuindo a qualidade do perfil de personalidade.

Apesar de existirem diversos aplicativos no mercado para encontrar atrações e organizar roteiros de viagens, não encontrei nenhum com a proposta de apresentar os resultados das pesquisas de forma personalizada. O programa desenvolvido entrega o essencial para cumprir com o seu objetivo, que é fornecer resultados personalizados para as pesquisas feitas pelo usuário sobre atrações em viagens e poder guardar estes estes resultados para futuras consultas.

Porém para chamar a atenção do usuários, ele precisa ter mais funcionalidades, cobrindo os principais recursos de outros aplicativos de viagens e descobrindo novas necessidades dos viajantes. No caso de uma continuidade deste projeto, acho que seria essencial uma pesquisa com pessoas que costumam viajar com frequencia e profissionais da área de turismo, com entrevistas e formulários, para descobrir as maiores necessidades.

8.1. Oportunidades para Trabalhos Futuros

A área do tursimo é bastante vasta e cheia de possibilidades, então sempre aparecerão novas ideias e funcionalidades que podem ser adicionadas ao aplicativo. Além disso, algumas soluções poderiam ter sido feitas de formas diferentes, cada uma com seus prós e contras. A seguir temos alguns exemplos de opotunidades de melhorias.

- **Webcrawler**

A adição de sugestões turísticas ao banco de dados poderíam ser feitas através de webcrawers ao invés de um aplicativo de 'apoio'. Desta forma, o admistrador usaria um aplicativo para Windows, MacOS, Linux ou web para preencher os dados sobre uma sugestão turística a ser cadastrada. Um dos campos sería um link para a página do TripAdvisor referente à tal sugestão. Ao submeter o cadastro, a aplicação iria utilizar um webcrawler para ler a página do link enviado, coletar as a análises e críticas positivas e adicionar a um vetor.

Ou seja, ao invés do programa buscar as críticas em um perfil do Facebook, ele iria buscar direto do site TripAdvisor. Depois o programa passaría pelos mesmos processos, de tradução e análise do Personality Insights para a geração do perfil de personalidade e de persistência dos dados no Firebase.

Esta solução sería ideal, uma vez que tornaria o processo de adicionar sugestões turísticas ao Firebase menos manual e mais automatico.

- **Melhoria no cálculo do Match**

O Personality Insights é uma ferramenta poderosa e que fornece uma variedade muito grande de indicadores de personalidade. Estes indicadores possuem todos o mesmo peso no cáculo do 'match'. Porém alguns destes indicadores poderíam ter mais peso para certas sugestões turísticas e dessa forma o aplicativo forneceria um resultado mais preciso.

Uma boa oportunidade de melhoria seria primeiramente uma pesquisa de como esses indicadores infliuenciam nas escolhas e vontades de uma pessoas durante uma viagem. E com este resultado, estudar qual é a melhor forma possível de distribuir o peso desta influencia no calculo do 'match'.

- **Indicação de cidades**

Quando apresentadas ao aplicativo, algumas pessoas demonstraram interesse em uma funcionalidade que sugeriesse para elas uma cidade de destino. Para isto o ideal seria fazer um perfil de ‘personalidade da cidade’, que pode ser feito da mesma forma como é feito o perfil das ‘sugestões turísticas’. Outra forma sería fazer uma média do ‘match’ de todas as sugestões turísticas de cada cidade e usá-la como ‘match’ com a cidade.

Existem várias formas de como isso poderia ser apresentado para o usuário como, por exemplo, as duas a seguir.

  - O usuário escreve duas ou mais opções de destino. O aplicativo calcula o ‘match’ com cada uma dessas opções e apresenta o resultado como uma lista ordenada decrescentemente. A primeira cidade da lista seria a indicada.
  - O usuário pesquisa por um tema (que pode ser os mesmos dos filtros de sugestões já existentes) e o aplicativo apresenta em ordem decrescente por match cada cidade. A primeira cidade sería a que ele mais encontraría programas do tema pesquisado e parecidos com o seu perfil de personalidade.

- **Data de ida e volta**

Um dado que não é levado em consideração é a data de ida e volta da viagem. Este dado podería impactar de diversas formas os resultado das sugestões turísticas como, por exemplo, as duas a seguir:

  - Eventos normalmente acontecem em datas específicas, então não faz sentido apresentar um deles caso não aconteça durante o período de viagem do usuário. Desta forma, seria interessante guardar as datas dos eventos e comparar com a data da viagem do usuário para apresentar um evento como sugestão turística apenas se as datas forem compatíveis.
  - Existem alguns programas que são sasonais. Por exemplo, a Grouse Mountain em Vancouver funciona como estação de esquí no inverno e possui trilhas durante o verão.

- **Agenda de viagem**

Atualmente o aplicativo permite adicionar sugestões turísticas à uma lista de favoritos. Mas esta funcionalidade pode ser aperfeiçoada como uma agenda de viagem. Desta forma o usuário seria capaz de cadastrar uma viagem, com data de início e fim e visualiza-la na forma de calendário. Depois, ele poderia adicionar a sugestão turística em um certo dia e horário, formando a agenda.

9. Referências Bibliográficas

1 IBM. **Documentação do Personality Insights.** 2019. Disponível em: <https://cloud.ibm.com/docs/services/personality-insights>. Acesso em: 25 maio 2019.

2 OGATA, RICARDO; COUTINHO, KAÊ. **Sugestões de entretenimento usando computação cognitiva**. IBM, 2016. Disponível em: <https://www.ibm.com/developerworks/br/cloud/sugestao-entretenimento-computacao-cognitiva/index.html>. Acesso em: 3 maio 2019.

3 Apple. **sorted(by:).** Disponível em: <https://developer.apple.com/documentation/swift/array/2296815-sorted>. Acesso em: 20 junho 2019.

4 IBM Watson. **IBM Watson: How it Works.** 2014. Disponível em: <https://www.youtube.com/watch?v=_Xcmh1LQB9I&t=3s>. Acesso em: 20 setembro 2018.

5 IBM. **Documentação do Language Translator.** 2019. Disponível em: < https://cloud.ibm.com/docs/services/language-translator?topic=language-translator-gettingstarted>. Acesso em: 3 maio 2019.

6 IBM. **Personality Models.** 2017. Disponível em: <https://console.bluemix.net/docs/services/personality-insights/models.html#models>. Acesso em: 20 setembro 2018.

7 Facebook. **Graph API.** 2018. Disponível em: <https://developers.facebook.com/docs/graph-api/overview>. Acesso em: 25 maio 2019.

8 Firebase. **Cloud Firestore.** 2019. Disponível em: <https://firebase.google.com/docs/firestore>. Acesso em: 25 maio 2019.

9 Peterson, Steve; Christiani, Raimon. **Beyond bots and robots: Exploring the unrealized potential of cognitive computing in the travel industry.** Disponível em: <https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=GBE03776USEN&>. Acesso em: 20 setembro 2018.

10 Bringwater, Adrian. **Come Fly With AI, IBM Cloud Builds 'Chatbot' Virutal Travel Agent.** Forbes, 2016. Disponível em: <https://www.forbes.com/sites/adrianbridgwater/2016/11/22/come-fly-with-ai-ibm-cloud-builds-chatbot-virtual-travel-agent/#1088409b4813>. Acesso em: 20 setembro 2018.

11 Greengard, Samuel. **WayBlazer's Journey Leads to Cognitive Computing.** Baseline, 2015. Disponível em: <http://www.baselinemag.com/cloud-computing/wayblazers-journey-leads-to-cognitive-computing.html>. Acesso em: 24 setembro 2018.