INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Pet Virtual para Acompanhar e Motivar os Utilizadores num Sistema de Recomendações de Grupo Turístico

EDGAR MANUEL PILÃO CORDEIRO
Outubro de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

"*Pet*" Virtual para Acompanhar e Motivar os Utilizadores num Sistema de Recomendações de Grupo Turístico

Edgar Manuel Pilão Cordeiro

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Ana Almeida

Coorientador: Patrícia Alves

Júri:

Presidente:

Isabel Sampaio

Vogais:

Constantino Martins

Porto, 10 2023

Declaração de Integridade

Declaro ter conduzido este trabalho académico com integridade.

Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Portanto, o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

ISEP, Porto, 13 de outubro de 2023

Resumo

O conceito de *"gamification"* é a aplicação de mecanismos e caraterísticas referentes a jogos de modo a resolver problemas e facilitar a aprendizagem com o objetivo de despoletar ações e comportamentos em ambientes fora do contexto de jogos.

Dentro desta grande área, é possível destacar um subdomínio relativo à utilização de "*pets"* virtuais. Um "*pet"* virtual insere-se na categoria de companheiro humano artificial. Estes podem ser desenvolvidos para companheirismo ou por apenas diversão. A interação com "*pets"* virtuais pode ou não ser orientada a objetivos. Em alguns casos, estes requerem interações por parte do utilizador para o manter operacional, no entanto, existem outros casos em que estas interações apenas ocorrem do lado do *"pet"* virtual.

O principal objetivo desta dissertação resume-se ao estudo dos métodos de *"gamification"*, nomeadamente a aplicação de "*pets*" virtuais num sistema de recomendação para grupos de turismo sensível ao contexto, especialmente como método de apresentação de mensagens relativas ao contexto das atividades turísticas a realizar e como assistente virtual na aplicação desenvolvida. Do estudo efetuado, foram analisados os conceitos de sistema de recomendação e de "*pet*" virtual, sempre tendo em conta o mercado turístico.

Deste estudo foram retiradas ilações que foram aplicadas a uma solução que ajuda os turistas que utilizem a aplicação a ter uma experiência mais personalizada, na medida em que terão um assistente que sugere novos pontos de interesse a visitar perto de si e, ao mesmo tempo apresenta-lhes informações importantes no que toca ao contexto das atividades a realizar em tempo real.

Os resultados obtidos neste trabalho foram bastante satisfatórios tendo em conta os objetivos propostos no início do projeto, porém, é possível admitir que um trabalho futuro deve ser direcionado a estas funcionalidades, de modo a enriquecer, alargar e aperfeiçoar a implementação e renderização do "*pet*" virtual.

Palavras-chave: Pet Virtual, *"Gamification"*, Turismo, Sistemas de Recomendação para Grupos

Abstract

The concept of gamification is the application of mechanisms and characteristics related to games to solve problems and facilitate learning to trigger actions and behaviors in environments outside the context of games.

Within this large area, it is possible to highlight a subdomain related to the use of virtual pets. A virtual pet falls into the category of artificial human companion. They can be developed for the purpose of companionship or just for fun. Interaction with virtual pets may or may not be goal oriented. In some cases, these require user interactions to keep them operational, however there are other cases in which these interactions only occur on the side of the virtual pet.

The main objective of this dissertation is the study of gamification methods, namely the application of virtual "pets" in a context-sensitive recommendation system for tourism groups, especially as a method of presenting messages related to the context of tourist activities. to be carried out and as a virtual assistant in the developed application. From the study carried out, the concepts of recommendation system and virtual pet were analyzed, always considering the tourist market.

From this study, conclusions were drawn and applied to a solution that aims to help tourists who use the application to have a more personalized experience, as they will have an assistant who will suggest new points of interest to visit close to them and at the same time presents them with important information regarding the context of the activities to be carried out in real time.

The results obtained in this work were quite satisfactory taking into account the objectives proposed at the beginning of the project, however, it is possible to admit that future work should be directed to these functionalities, in order to enrich, extend and improve the implementation and rendering of the virtual pet.

Keywords: Virtual Pet, Gamification, Tourism, Group Recommender Systems

Agradecimentos

Gostaria, em primeiro lugar, de agradecer à minha família pelo imenso apoio ao longo deste percurso académico e por me proporcionarem todas as condições para o meu sucesso pessoal e profissional.

Agradecer também aos meus amigos mais chegados por me motivarem e ajudarem a ultrapassar todas as noites mal dormidas na busca de sucesso neste percurso académico.

Enaltecer o apoio da minha namorada que, durante todo este percurso, aturou-me e lidou com todos os meus problemas e preocupações sem nunca se importar.

Às professoras Ana Almeida e Patrícia Alves pelo apoio prestado e por me guiarem neste projeto de forma a atingir as metas designadas no início desta jornada.

Por fim, gostaria de agradecer a todos os meus colegas de curso que me acompanharam durante as aulas do curso e que me permitiram manter o foco e atitude positiva de forma a terminar este percurso com os objetivos delineados atingidos.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

SR	Sistema de Recomendação
SRG	Sistema de Recomendação para Grupos
DNC	Modelo de Desenvolvimento de um Novo Conceito
QFD	Qualify Function Deployment
REST	Representational State Transfer
IC	Índice de Consistência
IR	Índice de Aleatoriedade
RC	Razão de Consistência
GECAD	Research Group on Intelligent Engineering and Computing For Advanced Innovation And Development

1 Introdução

1.1 Contexto

A personalização é cada vez mais reconhecida como um fator-chave na eficácia dos sistemas de recomendação (SR), portanto, quanto mais estes souberem sobre os seus utilizadores, melhores recomendações podem fazer. O uso de aspetos psicológicos, hábitos e contexto do utilizador para gerar recomendações é uma tendência crescente nos SR que apresenta melhores resultados do que as abordagens comuns e também ajuda com problemas de "*cold-start*" para fornecer uma abordagem mais personalizada. No entanto, esta abordagem pode ter muitas limitações, seja devido à configuração demorada, intervenção excessiva ou à necessidade de inúmeras ações do utilizador para criar o perfil [67].

A aplicação de jogos e elementos de jogos em contextos não relacionados a jogos ("*gamification*"), como na educação, saúde, negócios e turismo, é um conceito interessante que o SR pode alavancar. A "*gamification"* melhora a motivação do utilizador, mostrando excelentes resultados como o método de próxima geração para "*marketing*" e aquisição de clientes. Portanto, os utilizadores podem ficar mais motivados a usar o SR ao adicionar componentes do jogo ao sistema, como, por exemplo, um "*pet*" virtual que o acompanha [67].

Neste trabalho foi desenvolvido um "*pet*" virtual com o intuito de ser integrado no protótipo de um Sistema de Recomendação para Grupos (SRG) de turismo denominado GrouPlanner do GECAD.

O GECAD é uma Unidade de Investigação que tem como missão o desenvolvimento da investigação científica e inovação para a incorporação da Inteligência em Sistemas Complexos de Engenharia e Computação. Os Sistemas Inteligentes de Energia são a referência Internacional e Nacional de Excelência do GECAD, nomeadamente em áreas como "*Smart Grids"*, Mercados de Eletricidade, Recursos Distribuídos, "*Demand Response"*, Eficiência Energética e Edifícios Inteligentes [38].

O projeto GrouPlanner é uma aplicação Android para dispositivos móveis que visa a realização de estudos, investigação e experimentação no âmbito dos Sistemas de

Recomendação de Grupos e destina-se a apoiar grupos de turistas chineses em todas as vertentes do planeamento de viagens e seleção de Pontos de Interesse (POI) a visitar no Norte de Portugal [37]. Para este projeto foi desenvolvido um sistema de recomendação para grupos de turismo baseado em micro serviços, sendo um deles, o micro serviço multiagente, que permite a comunicação direta com os agentes dos turistas usando "*endpoints*" REST, tirando proveito das habilidades sociais destes para interagirem entre si e ajudar a refinar as recomendações individuais e de grupo [61].

1.2 Motivação

No caso de um SR para Turismo, o *"pet"* virtual seria como um amigo do turista e pode desempenhar um papel importante no sistema ao acompanhar o turista, ajudando a decidir o roteiro com base no contexto do turista e motivando-o durante uma excursão, apresentando informações inteligentes, por exemplo *"push-notifications*", e propondo desafios personalizados conforme as intenções e interesses do turista [24].

Os "*pets*" virtuais demonstraram ter um impacto positivo no bem-estar dos seus utilizadores. Estudos demonstraram que interagir com "*pets*" virtuais pode reduzir os níveis de "*stress*" e ansiedade, aumentar a felicidade e melhorar o bem-estar emocional. Estes benefícios podem ser particularmente importantes para os turistas que enfrentam uma série de fatores de "*stress*", como o "*jet lag*", diferenças culturais, saudades de casa, medo do desconhecido ou falta de companhia e apoio [24].

Estes benefícios associados com as funcionalidades descritas anteriormente revelam um forte motivo para o uso de um "*pet*" virtual na aplicação GrouPlanner visto que pode auxiliar o turista a sentir empatia com o sistema [23].

1.3 Objetivos

O objetivo do "*pet"* virtual é acompanhar e motivar o turista durante o uso do aplicativo, como na configuração do perfil, criação de grupos de excursão, na apresentação das recomendações, etc., utilizando implicitamente os dados pessoais do turista e as informações de contexto.

Eram esperados os seguintes resultados:

- Contextualização sobre o estado da arte de:
  - Sistemas de Recomendação (para Grupos);
  - SR sensíveis ao contexto;
  - *"Gamification"* em geral e aplicada ao turismo;
  - Assistentes virtuais em geral e aplicados a SR.
- Desenvolvimento de um animal de estimação virtual, que será integrado no protótipo do SRG existente;

- Teste da integração do "*pet*" virtual no protótipo SRG usando cenários de casos de uso reais;
- Análise dos resultados e redação da dissertação.

1.4 Abordagem

Os SRG são um tópico importante e complexo no campo dos SR [1]. Os membros do grupo têm gostos diferentes e é difícil encontrar uma solução que satisfaça a todos. É muito importante garantir que nenhum dos membros do grupo fique insatisfeito, pois este sentimento pode espalhar-se no grupo através do "*ripple effect of emotional contagion*" [2]. Por isso, um SRG que pode fornecer recomendações contextualizadas pode ser a solução perfeita.

Os SRG encontrados na literatura são intrusivos na forma como apresentam as suas recomendações e não focam nas motivações pessoais dos turistas e nos seus interesses, levando a ignorar os outros membros do grupo [18].

A personalização é uma componente chave do sucesso do SR no turismo [3]. Quanto mais informações soubermos sobre o turista, melhores sugestões poderemos fazer. Informações como demografia turística, traços de personalidade, aspetos socioculturais, hábitos e preferências podem ser fatores importantes na eficácia do sistema. Foi demonstrado que a personalidade melhora as recomendações do grupo e também pode ajudar com problemas de "arranque a frio" [4].

A "*gamification*" pode ser o veículo perfeito. Foi demonstrado que a "*gamification*" aumenta o envolvimento e a motivação do utilizador em coisas como a aprendizagem e o trabalho [5]. O SRG pode se tornar mais emocionante com a adição de componentes do jogo. A tecnologia de "*gamification*" também pode ser usada para introduzir avatares nos dispositivos móveis de modo a personificar o sistema. Um avatar é como um acompanhante de um viajante, acompanhando-o durante todo o processo, ajudando a definir o itinerário do grupo ao qual o viajante pertence, fornecendo informações inteligentes e motivando o viajante durante o passeio, podendo desempenhar um papel importante no sistema, sugerindo desafios personalizados segundo as intenções e interesses do turista. Representar o viajante por meio de um avatar claramente ajuda-o a desenvolver empatia pelo sistema [6].

Os jogos de realidade aumentada baseados em localização têm grande potencial e podem ser uma maneira mais inteligente dos turistas visitarem o país.

Como primeiro passo, é necessário explorar o mercado atual e as aplicações existentes que permitem este tipo de "*gamification*" e personificação do utilizador. A seguir, uma análise dos vários motores de desenho e animação utilizados na construção do "*pet*" virtual e das várias potencialidades da sua idealização na plataforma integrada.

No contexto da sua aplicação, deve-se planear e estruturar como este novo serviço de personificação do utilizador será implementado, bem como de que forma irá armazenar diversas animações para o seu "*pet*" virtual.

A comunicação deste novo sistema e a sua integração em aplicações já desenvolvidas também precisam de ser estruturadas e consideradas para posterior implementação. Durante este processo, houve um contacto constante com os professores do GECAD para entender o nível de sucesso esperado ao nível da solução e definir metas de usabilidade para o uso bem-sucedido da aplicação pelos turistas.

1.5 Metodologia

A metodologia adotada para o desenvolvimento da solução foi baseada no modelo de cascata com retroalimentação. Neste modelo, as atividades do processo de desenvolvimento são estruturadas numa cascata de forma linear e sequencial, que permite a revisão de fases anteriores e também correções que sejam necessárias noutras fases do processo.

Neste projeto também foram adotadas metodologias ágeis com "*Scrum"*, baseada em sprints, ciclos de produção de um projeto, garantindo revisão e aperfeiçoamento constantes para o resultado ser sempre o melhor possível.

A solução foi desenvolvida atendendo às melhores práticas de programação, nomeadamente ao uso de micro serviços, uma vez que permitem uma melhor modularidade, escalabilidade e podem ser implementados de forma independente, cada um com a sua própria base de dados. Isso significa que pode(m) ser usada(s) a(s) linguagem(s) de programação mais adequada(s) para cada serviço, haverá um melhor isolamento de falhas, uma entrega contínua e componentes espalhados por vários servidores. A comunicação entre os micro serviços será assíncrona e por meio do protocolo REST.

1.6 Estrutura do Documento

Este documento está estruturado em diferentes capítulos, cada um deles com informações diferentes:

- **Introdução**: O primeiro capítulo apresenta algumas informações sobre o problema a resolver, a interpretação do mesmo, o contexto, os objetivos e motivação deste projeto.
- **Análise de Valor**: Neste capítulo, é discutida a análise de valor da ideia com o potencial de alavancar o projeto para o mercado e desenvolver um negócio real em torno do mesmo.
- **Estado da Arte**: Neste capítulo, é fornecida uma breve introdução à aplicação de turismo em que este projeto se baseia, assim como ao conceito de *"gamification"* e à aplicação de *"pets"* virtuais num contexto turístico. São abordados mecanismos de contexto relativos a diferentes atividades turísticas, assim como são discutidas as

diferentes bibliotecas e aplicações utilizadas para a apresentação e desenvolvimento de modelos 3D em aplicações Android.

- **Design**: Neste capítulo é discutida a abordagem arquitetónica do sistema desenvolvido, nomeadamente a visão estrutural e funcionalidades implementadas.
- **Experiências e Avaliação**: Neste capítulo são analisadas as experiências realizadas em relação ao modelo estabelecido e os tipos de avaliação considerados, bem como o desempenho apresentado considerando as métricas de conduta (dos tipos de avaliação) adotadas.
- **Conclusão**: Neste capítulo, é feita uma conclusão sobre o trabalho escrito, bem como a discussão das próximas etapas e os resultados obtidos ao longo do trabalho.

2 Estado da Arte

Neste capítulo, são analisadas as definições e as caraterísticas-chave de alguns dos principais componentes do projeto desenvolvido. É também realizada uma profunda investigação às ferramentas disponíveis no estado atual e aos conceitos inerentes ao projeto para ter uma visão mais concreta e objetiva dos assuntos abordados e também quais as ferramentas que melhor se enquadram no caminho para os objetivos propostos. O final de cada subcapítulo tem um pequeno parágrafo onde se conclui sobre a pesquisa feita e quais as ferramentas escolhidas do leque pesquisado e a respetiva justificação.

2.1 Turismo

O conceito de Turismo, segundo a Organização Mundial de Turismo/Nações Unidas, define-se como "o conjunto de atividades *que as pessoas realizam durante as suas viagens e permanência em lugares distintos dos que vivem, por um período de tempo inferior a um ano consecutivo, com fins de lazer, negócios e outros.*" [8]. Mathieson e Wall [9], vieram completar esta definição realçando as complexidades da atividade turística, referindo que *"o turismo é o movimento temporário de pessoas para destinos fora dos seus locais habituais de trabalho e residência, as atividades desenvolvidas durante a permanência nesses destinos e as facilidades criadas para satisfazer as suas necessidades*" [9].

2.1.1 Análise do Mercado Turístico

A indústria de Turismo está em rápido crescimento, contribuindo significativamente para a economia global. Segundo a Organização Mundial do Turismo, as deslocações de turistas internacionais aumentaram de 25 milhões em 1950 para 1,5 bilhões em 2019 [39]. Este crescimento aumentou o interesse na pesquisa do mercado turístico, nomeadamente nas áreas da segmentação do mercado e comportamento do consumidor.

A segmentação do mercado passa por um processo de divisão do mercado em grupos de consumidores de menor dimensão e com necessidades/características semelhantes. No mercado turístico, a segmentação é utilizada para atingir grupos específicos de turistas. Pesquisas recentes demonstram que a segmentação do mercado é crucial para o "*marketing*" eficaz e o desenvolvimento de produtos [10]. Nestes mesmos estudos é explorada a utilização de novas tecnologias, como, por exemplo, aplicações eletrónicas e *"big data"*, para melhorar a segmentação do mercado [11].

O comportamento do consumidor é um aspeto essencial do mercado turístico, visto que determina a necessidade de diferentes tipos de produtos e serviços turísticos. A investigação efetuada nesta área explorou, maioritariamente, os fatores que influenciam as decisões tomadas por um turista consoante a sua viagem, como a motivação, o valor percebido e a satisfação. Por exemplo, estudos demonstram que as atitudes e preocupações ambientais são cada vez mais importantes na tomada de decisões por parte do turista [12]. Outras pesquisas exploram o papel das emoções nas experiências turísticas e o seu impacto na satisfação e lealdade do turista [13].

O mercado turístico é uma indústria complexa e dinâmica que requer um processo de investigação contínua de modo a entender os seus vários componentes. Segmentação do mercado, comportamento do consumidor, competitividade do destino e sustentabilidade são áreas essenciais do mercado turístico, dado que contribuem para o sucesso e para a sustentabilidade do setor. No desenvolvimento deste projeto, segmentação do mercado e comportamento do consumidor foram aspetos importantes no desenvolvimento do mesmo, visto que a aplicação elaborada tira partido destas novas tendências do mundo turístico enquanto fornece planeamento turístico detalhado e contextual segundo o perfil do turista assim como dos diferentes membros que compõem o grupo turístico.

2.1.2 Análise das Dificuldades de um Turista no Planeamento Turístico

Planear uma viagem pode ser um processo complexo, e os turistas podem encontrar diversas dificuldades ao longo do processo de planeamento. Após alguma pesquisa, foram identificadas as principais dificuldades do turista no planeamento de uma viagem que serão discutidas posteriormente.

Um dos maiores desafios que os turistas enfrentam no planeamento de uma viagem é a sobrecarga de informação. Com tantos recursos disponíveis, desde guias até "*sites"* de viagens *"online"*, pode tornar-se difícil filtrar a quantidade de informação adquirida e encontrar fontes relevantes e de confiança. Pesquisas recentes revelam que a sobrecarga de informação pode levar a confusão, ansiedade e fadiga no processo de decisão, influenciando negativamente o processo de planeamento turístico [14].

Outra dificuldade que os turistas encontram ao planear uma viagem são as restrições monetárias. As despesas da viagem podem aumentar rapidamente e os turistas podem precisar de priorizar certas atividades com base no seu orçamento. Os turistas preocupados com o

orçamento são mais propensos a usar recursos "*online*" para planear a sua viagem, o que revela que as restrições monetárias desempenham um papel significativo no processo de planeamento [15].

As restrições de tempo também podem apresentar desafios para os turistas. Os turistas podem ter um período de férias limitado ou um itinerário de viagem fixo, o que resulta numa maior dificuldade na inclusão de todas as atividades planeadas. Os turistas com tempo limitado são mais propensos a usar as redes sociais e sites de avaliação para planear a sua viagem, visto que esses recursos fornecem informações rápidas e de fácil acesso [16].

O planeamento de uma viagem pode ser um processo complexo, mas existem maneiras de mitigar as dificuldades e garantir uma viagem tranquila e agradável. Os turistas podem beneficiar do uso de recursos fidedignos e relevantes para evitar uma sobrecarga de informação e dificuldades na priorização de atividades com base no seu orçamento e restrições de tempo. É neste contexto que se insere a aplicação desenvolvida pelo GECAD, visto que permite mitigar estas dificuldades e permite uma organização atempada e em tempo real no planeamento da viagem com sugestões contextuais de cada atividade.

2.2 Sistemas de Recomendação

Nos últimos anos, a indústria turística passou por mudanças significativas, com o crescimento das agências de viagens "*online*" e das plataformas de avaliação de viagens, permitindo o acesso a uma ampla gama de informação sobre viagens. No entanto, a abundância de informação pode ser esmagadora e os turistas geralmente exigem recomendações personalizadas que atendam às suas preferências. Para resolver este problema, os sistemas de recomendação tornaram-se uma solução popular na indústria do turismo, revelando bons resultados na ajuda aos turistas na identificação de opções de viagem satisfatórias com base nos seus interesses e preferências [68].

Os sistemas de recomendação são um tipo de sistema de filtragem de informação que permite prever a preferência que um utilizador daria a um determinado assunto. Estes sistemas tornaram-se omnipresentes nos últimos anos sendo aplicados em diversas áreas, como filmes, música, notícias, livros, artigos de pesquisa, consultas de pesquisa e produtos em geral [69]. O desenvolvimento de sistemas de recomendação foi impulsionado devido a uma maior disponibilidade de dados, a avanços tecnológicos na área de *"Machine Learning"* e à crescente necessidade de personalizar a experiência do utilizador [17].

Existem vários tipos de sistemas de recomendação, nomeadamente os de filtragem colaborativa, os baseados em conteúdo e os sistemas híbridos [17]:

- Os **sistemas de recomendação baseados em conteúdo** recomendam produtos aos utilizadores com base nas suas preferências anteriores por produtos semelhantes. Este tipo de sistema compara os atributos dos produtos que um utilizador gostou no passado com os produtos da base de dados do sistema e

recomenda os produtos mais semelhantes. Esta abordagem é baseada na ideia de que as pessoas gostam de produtos semelhantes aos que gostaram no passado.

- Os **sistemas de recomendação de filtragem colaborativa**, por outro lado, fazem recomendações com base nas preferências de outros utilizadores. Este tipo de sistema identifica utilizadores com preferências semelhantes e recomenda produtos que estes gostaram. Existem duas abordagens para a filtragem colaborativa: uma baseada em memória e outra baseada em modelo. A filtragem colaborativa baseada em memória usa toda a matriz utilizador-produto para calcular as semelhanças entre utilizadores e produtos. A filtragem colaborativa baseada em modelo usa algoritmos de *"Machine Learning"* para encontrar fatores semelhantes que explicam as relações entre utilizadores e produtos.
- Os **sistemas de recomendação híbridos** combinam as abordagens dos sistemas descritos acima. Estes sistemas usam tanto o conteúdo dos produtos quanto as preferências de outros utilizadores para fazer recomendações. A ideia por trás dos sistemas híbridos é alavancar os pontos fortes das abordagens de filtragem colaborativa e baseada em conteúdo e superar as suas respetivas limitações.

Os sistemas de recomendação são usados para sugerir produtos personalizados aos utilizadores com base nas suas preferências e comportamentos. Os sistemas de recomendação para grupos, por outro lado, sugerem produtos relevantes para um grupo de utilizadores. Estes sistemas visam fornecer recomendações que satisfaçam as preferências de vários utilizadores, procurando ao mesmo tempo, promover a coesão do grupo e evitar conflitos.

Os sistemas de recomendação para grupos são investigados em vários campos, incluindo no "*e-commerce*", nas redes sociais e no turismo. No "*e-commerce*", os sistemas de recomendação para grupos são utilizados para sugerir produtos a grupos de utilizadores no caso de uma compra conjunta. Nas redes sociais, os sistemas de recomendação para grupos são utilizados para sugerir conteúdos a um grupo de utilizadores com base nos seus interesses comuns. No turismo, os sistemas de recomendação para grupos são usados para sugerir pacotes de viagem para grupos de utilizadores, procurando dar soluções que satisfaçam turistas com preferências diferentes [18].

Existem diferentes abordagens para o desenvolvimento de sistemas de recomendação para grupos. Uma abordagem é agregar as preferências individuais dos membros do grupo para gerar uma preferência de grupo [18]. Esta abordagem assume que a preferência do grupo pode ser inferida a partir das preferências individuais dos membros que o compõem. Outra abordagem é usar modelos de grupo que representam as preferências do grupo todo. Os modelos de grupo podem ser baseados em técnicas estatísticas ou na teoria da escolha social [18].

Vários estudos mostram que os sistemas de recomendação para grupos podem melhorar a satisfação do utilizador e a coesão do grupo. No entanto, os SRG também enfrentam

alguns desafios, nomeadamente lidar com preferências em conflito e garantir justiça e diversidade nas recomendações [18].

2.2.1 Sistemas de Recomendação aplicados ao Turismo

A aplicação de sistemas de recomendação na indústria do turismo está bastante generalizada. Seguidamente são enumerados alguns exemplos recentes desta tecnologia aplicada ao turismo em plataformas comerciais "*online"*:

- O **TripAdvisor** é um site de viagens que usa um sistema de recomendação para sugerir atrações turísticas aos seus utilizadores. O sistema toma em consideração o histórico de buscas, localização e avaliações do utilizador para sugerir os melhores pontos turísticos de uma determinada cidade ou região [40].
- O **Foursquare** é uma rede social que usa um sistema de recomendação para sugerir atrações turísticas populares com base na localização do utilizador e no histórico de "*check-in*". O sistema também considera as avaliações de outros utilizadores para fornecer recomendações personalizadas [41].
- O **Google Maps** é um serviço de mapeamento que usa um sistema de recomendação para sugerir atrações turísticas com base na localização e no histórico de pesquisa do utilizador. O sistema toma em consideração as avaliações, críticas e popularidade para sugerir as melhores atrações turísticas de uma determinada área [42].
- A **Expedia** é um site de viagens que usa um sistema de recomendação para sugerir atrações turísticas aos seus utilizadores. O sistema toma em consideração o histórico de pesquisas, localização e avaliações do utilizador para sugerir os melhores pontos turísticos de uma determinada cidade ou região [43].
- O **Airbnb** é uma plataforma de aluguer de imóveis que usa um sistema de recomendação para sugerir atrações turísticas aos seus utilizadores. O sistema toma em consideração o histórico de pesquisas, localização e avaliações do utilizador para sugerir os melhores pontos turísticos de uma determinada cidade ou região [44].

Os sistemas de recomendação, para além de serem aplicados a plataformas comerciais como as denominadas acima, têm emergido como aplicações completas e singulares aplicadas ao turismo com ênfase na personalização:

- **INTRIGUE** é um protótipo dum servidor de informações turísticas que apresenta informações sobre a área ao redor da cidade de Turim, Itália, em computadores e dispositivos moveis. Este sistema recomenda destinos e roteiros turísticos tendo em conta as preferências de grupos turísticos heterogéneos (como famílias com crianças ou idosos) e explica as recomendações atendendo às necessidades dos membros do grupo. Além disso, o sistema disponibiliza uma agenda interativa para agendamento do

passeio. Os serviços oferecidos pela INTRIGUE contam com modelagem do utilizador e técnicas de hipermédia adaptativa [49].

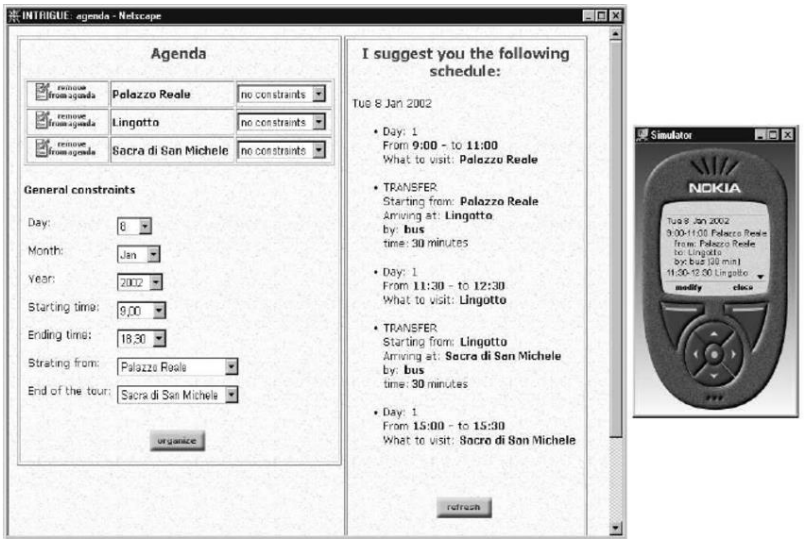


Figura 1 - Sistema de Recomendação INTRIGUE, retirado de [49]

- **Turist@** é um sistema com um "*design*" modular baseado em agentes que permite modelar diferentes tipos de atividades de maneira flexível e permite a implementação de um "*front-end*" com reconhecimento de localização no dispositivo móvel do utilizador. Um cuidado especial foi dado ao mecanismo de recomendação, implementado por meio de um Agente de Recomendação especializado. Este incorpora uma mistura de estratégias de recomendação baseadas em conteúdo e colaborativas, evitando assim as desvantagens de cada método individual e consegue realizar recomendações em cenários heterogéneos. As recomendações consideram os perfis dos utilizadores, implicitamente atualizados após a análise das ações do mesmo (por exemplo, consultas, avaliações) [50].

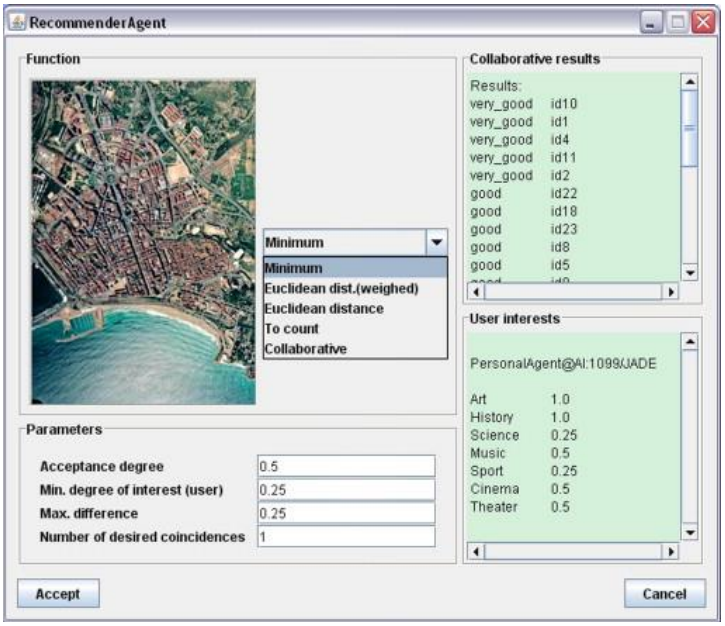


Figura 2 - Sistema de Recomendação Turist@, retirado de [50]

- Um sistema desenvolvido por Nguyen e Ricci consiste num SRG integrado num "*chat*" para dispositivos moveis que facilita a participação dos elementos de um grupo no processo de escolha. Esta aplicação é parecida ao WhatsApp com a adição de funcionalidades extra que permite aos utilizadores avaliar o POI visitado anteriormente e definir o seu estado de espírito. Os utilizadores podem classificar o POI recomendado, como o melhor ou se simplesmente gostaram ou não. Esta avaliação permite ao sistema inferir as restrições dos utilizadores com base nos atributos do POI classificado, e atualizar gradualmente as informações sobre um POI recomendado com explicações adicionais, com base nessas restrições. [51].

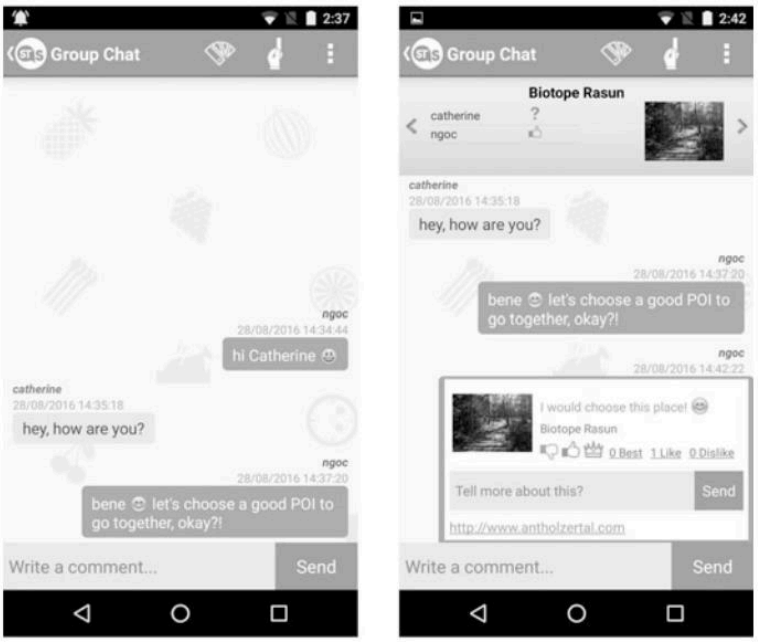


Figura 3 - Sistema de Recomendação STSGroup, retirado de [51]

A utilização de sistemas de recomendação individuais e para grupos tornou-se cada vez mais popular na indústria do turismo, fornecendo aos turistas recomendações personalizadas que atendem aos seus interesses, preferências e restrições. No entanto, após alguma análise, existem poucos sistemas que têm em conta o planeamento da viagem baseado na personalidade e preferências do utilizador, assim como a adaptação do mesmo em tempo real segundo o contexto das atividades. É neste campo que a aplicação desenvolvida vem resolver o problema, nomeadamente com a implementação do *"pet"* virtual para personalizar e adaptar a experiência do utilizador no planeamento da sua viagem.

2.3 "Gamification"

A *"gamification"* é um termo usado para descrever o processo de aplicação de elementos de jogos a contextos não relacionados a jogos, como educação, "*marketing*" e até mesmo o turismo. O uso da *"gamification"* no turismo ganhou uma atenção significativa nos últimos anos, visto que oferece uma maneira inovadora de cativar os turistas, aprimorar as experiências e aumentar os níveis de satisfação [19].

A *"gamification"* baseia-se no princípio de que os seres humanos são naturalmente motivados por desafios, feedback e reconhecimento social. O uso da *"gamification"* tornou-se cada vez mais popular em vários setores, incluindo na educação, no "*marketing*" e na saúde [70].

A psicologia da *"gamification"* é baseada nos princípios de motivação e dedicação. Segundo a Teoria da Autodeterminação, as pessoas são motivadas por três necessidades psicológicas básicas: autonomia, competência e relacionamento. A autonomia refere-se à necessidade de se sentir no controlo das suas ações, a competência refere-se à necessidade de se sentir eficaz e capaz e o relacionamento refere-se à necessidade de se sentir conectado aos outros [19].

A *"gamification"* tira proveito dessas necessidades psicológicas, fornecendo aos utilizadores uma sensação de controlo, progresso e interação social. Ao incorporar elementos como recompensas e tabelas de classificação, a *"gamification"* explora a motivação intrínseca das pessoas e promove o envolvimento com o conteúdo [19].

Embora a *"gamification"* seja eficaz na promoção da motivação e da dedicação, esta não é uma panaceia. A eficácia da *"gamification"* depende de vários fatores, como a motivação do utilizador, o "*design*" do sistema ludificado e o contexto em que é usado [19].

Além disso, *a "gamification"* pode não ser adequada para todos os contextos. Esta pode não ser eficaz na promoção de mudanças de comportamento a longo prazo, visto que os utilizadores podem tornar-se dependentes da mecânica do jogo em vez de captar o comportamento. Esta pode também não ser motivadora para certas idades ou personalidades, uma vez que há pessoas que não gostam de jogos no telemóvel [20].

2.3.1 *"Gamification"* aplicada ao Turismo

O uso da *"gamification"* no turismo oferece muitos benefícios, incluindo maior dedicação, motivação e satisfação dos turistas [19]. A *"gamification"* pode ser aplicada de várias maneiras para aprimorar a experiência do turista, incluindo o uso de aplicações móveis, redes sociais e realidade aumentada.

Um exemplo de *"gamification"* no turismo é o uso de aplicações móveis que incorporam elementos de jogos, como sistemas de pontos e recompensas, para motivar e envolver os turistas. Por exemplo, a aplicação móvel "Omio" oferece recompensas aos turistas que reservarem vários modos de transporte, como voos e comboios, através da aplicação. Isso incentiva os turistas a usar o aplicativo e reservar mais experiências de viagem através dele, contribuindo para uma maior dedicação e fidelidade [45].

Outro exemplo de *"gamification"* no turismo é o uso das redes sociais para criar desafios e incentivar os turistas a compartilhar as suas experiências. Por exemplo, a campanha "Iceland Naturally" criou um desafio nas redes sociais que incentivou os turistas a tirar fotos de si mesmos em vários locais da Islândia e a compartilhá-las nas redes sociais usando uma hashtag específica. Este desafio incentivou os turistas a explorar mais locais e a partilhar as suas experiências com outras pessoas, contribuindo para um maior envolvimento e conhecimento do destino [46].

A realidade aumentada é outro exemplo de *"gamification"* no turismo, ao permitir que os turistas conectem com o ambiente de uma maneira nova e empolgante. Por exemplo, o aplicativo móvel "Sightsy" permite que os turistas usem os seus "*smartphones*" para detetar pontos de referência e locais históricos, para fornecer informações, vídeos e elementos interativos, como questionários e jogos. Isto melhora a experiência do turista, fornecendo uma maneira mais imersiva e interativa de aprender sobre o destino [47].

O uso da *"gamification"* no turismo tem-se revelado eficaz para melhorar a experiência do turista e aumentar os níveis de satisfação. O uso da *"gamification"* no turismo pode levar ao aumento dos níveis de dedicação e satisfação, bem como ao aumento da lealdade e da vontade de recomendar o destino a outras pessoas [21] [22].

2.4 "Pets" Virtuais

"Pets" virtuais são simulações digitais de animais da vida real idealizados para serem cuidados e interagirem com os seus donos. Embora os *"pets"* virtuais não tenham uma presença física, estes tornaram-se cada vez mais populares nos últimos anos, principalmente na forma de aplicações e jogos para dispositivos móveis [71].

Um jogo chamado "Neko Atsume" foi criado e lançado em 2014 para dispositivos móveis pela Hit-Point. O objetivo do jogo é atrair o máximo de gatos possível e terminar o livro dos gatos, uma lista de todos os gatos que frequentam o jardim do jogador. Para enfeitar o jardim e atrair mais gatos, cada um com aparência e personalidade distintas, os jogadores também podem comprar comida e brinquedos. Os jogadores podem tirar fotos dos gatos e aprender detalhes sobre cada um, como o nome, raça e comida preferida, assim que chegarem ao jardim [72].

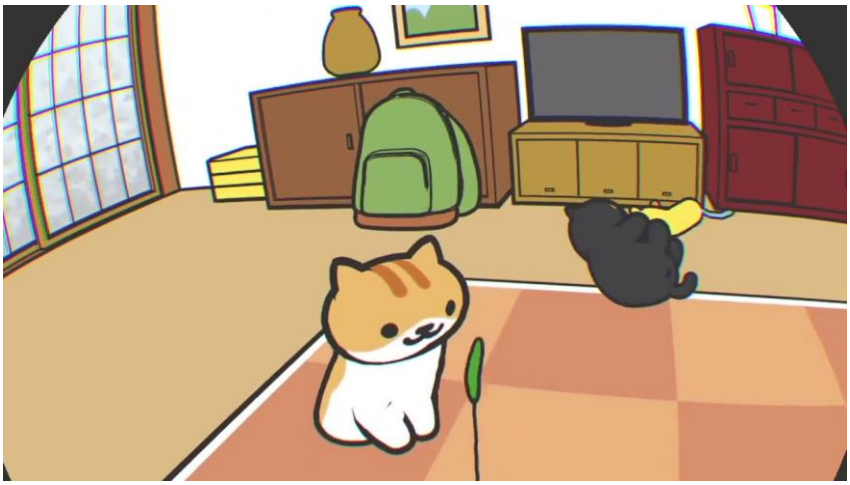


Figura 4 – Aplicação Neko Atsume, retirado de [75]

"MetaPals" é um jogo de animais de estimação único que funciona como uma extensão do Chrome que permite interagir e criar animais de estimação virtuais diretamente no ecrã do

computador. Os “MetaPals” podem ser criados com personalidades distintas, para que cada um dedique tempo para aprender, desenvolver e exibir comportamentos positivos ou negativos, tal como um animal de estimação real faria [73].

Figura 5 - Aplicacão MetaPals, retirado de [76]

Os *“pets”* virtuais podem ter um impacto positivo no comportamento humano, incentivando a empatia e a responsabilidade. Um estudo, conduzido por investigadores da Universidade da Flórida Central, descobriu que as crianças que brincavam com *“pets”* virtuais apresentavam níveis mais elevados de empatia e responsabilidade em comparação com aquelas que não brincavam [23].

Os *“Pets”* virtuais também ajudam a reduzir os níveis de “*stress*”. Um estudo publicado no *“International Journal of Human-Computer Interaction”* evidenciou que brincar com um *“pet”* virtual pode ter um efeito tranquilizante nos utilizadores, principalmente em situações em que se sentem stressados ou ansiosos [24].

“Pets” virtuais também podem melhorar as habilidades sociais, especialmente para crianças com autismo. Um estudo publicado no *“Journal of Autism and Developmental Disorders”* descobriu que crianças com autismo que brincavam com *“pets”* virtuais apresentaram habilidades sociais melhoradas e níveis reduzidos de ansiedade social [25].

Para indivíduos que não podem cuidar de um animal de estimação na vida real, os *“Pets”* virtuais podem fornecer companhia e apoio emocional. Um estudo publicado no “*Journal of Gerontological Nursing*” descobriu que *“pets”* virtuais podem ajudar a reduzir a solidão e a depressão entre adultos mais velhos [26].

Finalmente, descobriu-se que *“pets”* virtuais melhoram a função cognitiva em adultos mais velhos. Um estudo publicado no “*Journal of Gerontological Nursing*” descobriu que brincar

com um *"pet"* virtual pode melhorar a função cognitiva, principalmente nas áreas de atenção e memória [26].

Os *"pets"* virtuais podem ter uma série de efeitos positivos no comportamento humano e no bem-estar, desde o incentivo à empatia e responsabilidade, até à redução do "*stress*" e melhoria da função cognitiva. Estas descobertas sugerem que estes têm o potencial de melhorar a vida e estabilidade emocional dos seus utilizadores de várias maneiras.

2.4.1 "Pets" Virtuais aplicados ao Turismo

Os *"pets"* virtuais podem fornecer uma fonte de entretenimento e companheirismo para os turistas, especialmente aqueles que viajam sozinhos ou que não podem viajar com os seus animais de estimação.

Os *"pets"* virtuais demonstraram ter um impacto positivo no bem-estar dos seus utilizadores. Estudos demonstraram que interagir com *"pets"* virtuais pode reduzir os níveis de "*stress*" e ansiedade, aumentar a felicidade e melhorar o bem-estar emocional. Estes benefícios podem ser particularmente importantes para os turistas que enfrentam uma série de fatores de "*stress*", como o *"jet lag"*, diferenças culturais, saudades de casa, medo do desconhecido ou falta de companhia e apoio [24].

Os *"pets"* virtuais também podem ser usados para promover o turismo sustentável. A utilização de *"pets"* virtuais num contexto turístico pode ajudar a promover a consciência ambiental e incentivar práticas de turismo sustentável. Um estudo descobriu que os turistas que interagiram com um *"pet"* virtual associado a iniciativas ambientais eram mais propensos a envolver-se nas práticas de turismo sustentável, como reduzir o uso de água e energia [27].

Os *"pets"* virtuais também são usados no contexto do bem-estar animal no turismo. Em alguns casos, os *"pets"* virtuais têm sido usados como substitutos de animais reais, de modo a reduzir o impacto do turismo na vida selvagem. Por exemplo, *"pets"* virtuais são utilizados em jardins zoológicos e aquários como uma forma de proporcionar aos visitantes uma experiência animal prática, sem os impactos negativos que podem resultar do contacto direto com animais [28].

Os *"pets"* virtuais têm uma variedade de aplicações na indústria do turismo, desde o entretenimento e companhia até à promoção do turismo sustentável e bem-estar animal. Estudos científicos demonstraram que *"pets"* virtuais podem ter um impacto positivo no bem-estar dos seus utilizadores e podem ser uma ferramenta útil para promover práticas de turismo ambientalmente responsáveis e éticas. Tendo em conta a pesquisa sobre a utilização desta tecnologia aplicada ao turismo, é notório as grandes vantagens que esta traz ao utilizador, nomeadamente na redução de níveis de "*stress*" e compromisso com a aplicação. Deste modo, revela-se que esta tecnologia aplicada à recomendação contextual das atividades de um turista no seu planeamento da viagem pode traduzir uma maior efetividade na passagem da mensagem e numa maior atenção a estes alertas por parte do utilizador.

2.4.2 Aplicações de Modulação 3D

Como foi anunciado anteriormente, os "*pets*" virtuais tem uma série de efeitos positivos no comportamento humano e podem ser uma ferramenta útil para promover práticas de turismo ambientalmente responsáveis e éticas. Para serem usados com sucesso na aplicação Grouplanner, convém que o seu desenvolvimento seja pertinente e com um bom grau de detalhe. Para isso, convém estudar as aplicações de modulação 3D existentes no mercado e escolher qual se adequa mais ao desenho dos "*pets*" virtuais a serem renderizados.

Os mecanismos de modelagem 3D são ferramentas de "*software*" que permitem aos utilizadores criar modelos digitais 3D, animações e efeitos visuais. Alguns mecanismos populares de modulação 3D são enumerados abaixo:

- **Blender** é um mecanismo de modulação 3D gratuito e de código aberto, popular entre iniciantes e profissionais. É fácil de usar e possui um poderoso conjunto de recursos que permitem aos utilizadores criar modelos 3D complexos, animações e efeitos visuais. O Blender também possui uma comunidade ativa de utilizadores que criam e compartilham tutoriais e complementos, tornando-o uma ferramenta muito versátil [52].

Figura 6 - Aplicação de Modelação 3D Blender, retirado de [77]

- **Autodesk Maya** é um mecanismo de modulação 3D popular nas indústrias de filmes e jogos. Possui um conjunto abrangente de ferramentas para criar modelos 3D de alta qualidade, animações e efeitos visuais. O "*Maya*" também possui um forte suporte a "*scripts*" e "*API*", tornando-o um favorito para criar ferramentas personalizadas [53].

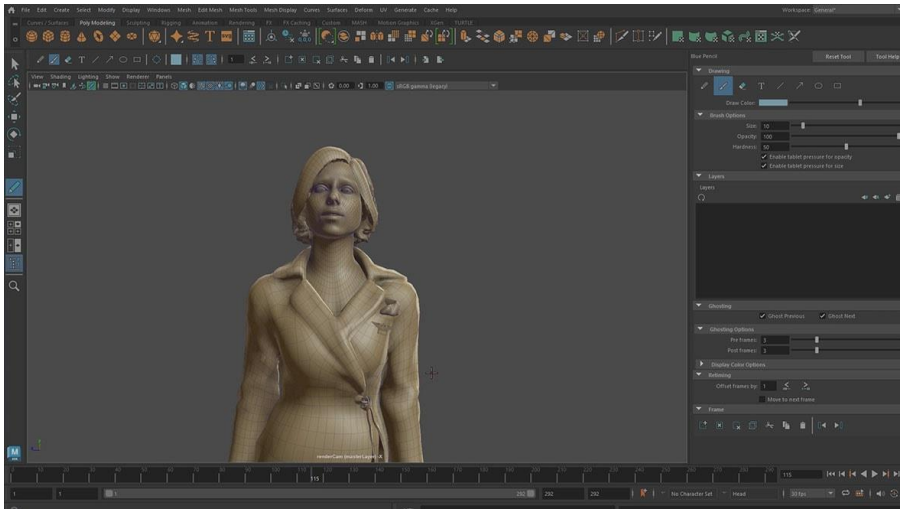


Figura 7 - Aplicação de Modelação 3D Autodesk Maya, retirado de [78]

- **3ds Max** é outro mecanismo de modulação 3D popular da "*Autodesk*" usado em arquitetura e "*design*" de produtos. Possui uma ampla gama de ferramentas para criar modelos e animações 3D complexos, além de um poderoso sistema de partículas para criar efeitos realistas [54].

Figura 8 - Aplicação de Modelação 3D 3ds Max, retirado de [79]

- **Cinema 4D** é um mecanismo de modulação 3D popular entre os "*designers*" de gráficos. Possui uma "*interface*" amigável e um conjunto abrangente de ferramentas para criar modelos 3D, animações e efeitos visuais. O Cinema 4D também possui um poderoso mecanismo de renderização que permite aos utilizadores criar renderizações finais de alta qualidade [55].

Figura 9 - Aplicação de Modelação 3D Cinema 4D, retirado de [80]

- **SketchUp** é um mecanismo de modulação 3D popular entre arquitetos, "*designers*" de interiores e "*designers*" de produtos. Possui uma "*interface*" amigável e uma ampla gama de ferramentas para criar modelos 3D de forma rápida e fácil. O "*SketchUp*" também possui uma grande biblioteca de "*plugins*" e extensões, tornando-o uma ferramenta muito versátil [56].

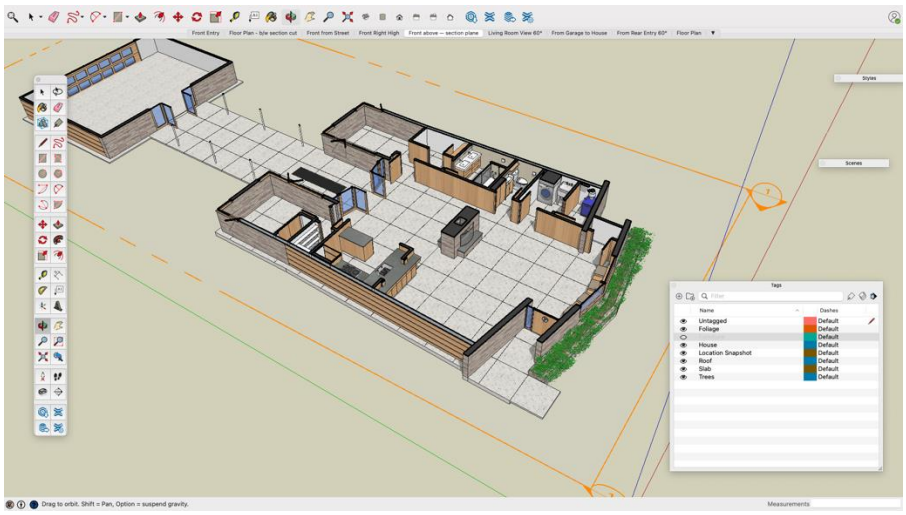


Figura 10 - Aplicação de Modelação 3D SketchUp, retirado de [81]

No geral, cada mecanismo de modulação 3D tem os seus pontos fortes e fracos, e a escolha de qual usar depende das necessidades e preferências específicas do utilizador. Tendo em conta a funcionalidade a ser desenvolvida, optou-se pelo Blender, visto que para além de ser uma

escolha popular devido à sua natureza de código aberto, comunidade ativa e conjunto abrangente de recursos, parece ser a ferramenta com menor grau de complexidade e dificuldade tendo em conta que a funcionalidade a ser desenvolvida não possuía um papel preponderante na aplicação final.

2.4.3 Bibliotecas de Renderização 3D em Android

Depois do estudo das aplicações de modelação, é necessário falar de bibliotecas de renderização em Android. O estudo destas bibliotecas é muito importante para o trabalho desenvolvido visto que se relaciona com a renderização do "*pet*" virtual na aplicação e convém que a biblioteca escolhida seja adequada ao grau de detalhe da modelação do mesmo. Estas bibliotecas são associadas ao sistema Android, visto que a aplicação Grouplanner, onde o "*pet*" virtual vai ser implementado, foi desenvolvida exclusivamente para dispositivos deste tipo.

Existem várias bibliotecas de renderização de ficheiros 3D disponíveis para Android, cada uma com o seu próprio conjunto de pontos fortes e fracos. A seguir são enumerados alguns exemplos destas bibliotecas:

- **Sceneform** é uma biblioteca de renderização de ficheiros 3D para Android projetada para ser fácil de usar e integrar ao "*ARCore*". Esta fornece muitos recursos e é adequada para criar aplicações de realidade aumentada que requerem gráficos de alta qualidade. No entanto, é limitada a um conjunto específico de formatos de ficheiro [57].
- **Assimp** é uma biblioteca de carregamento de ficheiros 3D para Android que suporta uma ampla variedade de formatos de ficheiro, tornando-a ideal para a construção de aplicações 3D que requerem compatibilidade com vários formatos de ficheiro. No entanto, não é um mecanismo de renderização 3D completo e requer integração com um mecanismo de renderização separado, o que pode adicionar complexidade ao processo de desenvolvimento [58].
- **Rajawali** é um mecanismo de renderização 3D para Android projetado para ser fácil de usar e leve. Fornece muitos recursos, sendo adequado para criar aplicativos 3D que requerem gráficos de alta qualidade. No entanto, pode não ser tão poderoso quanto outros mecanismos e pode ter suporte limitado [59].
- **Min3d** inclui um componente de carregamento de ficheiros 3D que permite carregar modelos 3D de uma variedade de formatos de ficheiro. Este foi projetado para ser leve e fácil de usar, tornando-o ideal para criar aplicativos 3D simples que requerem gráficos básicos. No entanto, pode não suportar todos os formatos de ficheiro suportados por outras bibliotecas e pode não ser tão poderoso quanto outros mecanismos de renderização [60].

Geralmente, a escolha da biblioteca de renderização de arquivos 3D para Android depende dos requisitos específicos da aplicação. Para aplicações simples que requerem gráficos básicos e suporte para um conjunto limitado de formatos de ficheiro, o min3d pode ser suficiente. No

presente caso, esta biblioteca foi a escolhida, uma vez que a funcionalidade desenvolvida não possuía um papel preponderante na aplicação final e o nível de renderização necessário não é elevado, ou seja, as renderizações a efetuar são simples e básicas e, portanto, convém utilizar uma biblioteca leve para não comprometer o desempenho do sistema desnecessariamente.

3 Análise De Valor

Neste capítulo, é definida e elaborada uma análise de valor para o projeto desenvolvido. Nesta análise, são considerados os principais elementos do modelo de Desenvolvimento de um Novo Conceito (DNC), é idealizada uma proposta de valor para o cliente assim como um modelo de negócio tendo em conta as caraterísticas do projeto.

3.1 Modelo de Desenvolvimento de um Novo Conceito

De acordo com o estudo desenvolvido por Koen, foi definido um novo DNC que consiste numa *"framework"*, bem como um conjunto de termos para tentar clarificar o conceito de *"fuzzy front-end"* [7].

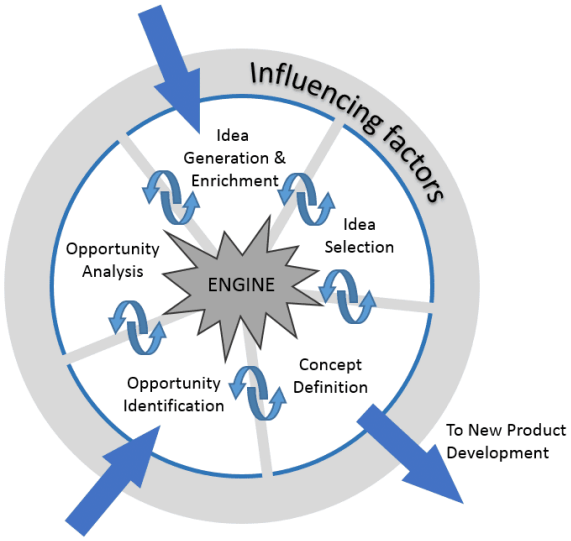


Figura 11 - Modelo de Desenvolvimento de um Novo Conceito, retirado de [82]

O modelo exemplificado acima identifica cinco elementos distintos na abordagem para desenvolver um novo conceito, organizados num esquema circular para representar a teoria de que as ideias devem fluir e iterar entre todos os cinco elementos [7].

3.1.1 Identificação da Oportunidade

O mercado está em constante mutação providenciando novas e diferentes oportunidades todos os dias para serem exploradas. Para identificá-las, uma análise de mercado deve ser planeada e realizada para identificar as necessidades do mesmo e fornecer novas e exclusivas ferramentas para satisfazer as lacunas encontradas.

O Turismo é uma indústria em rápido crescimento que contribui significativamente para a economia global. As deslocações de turistas internacionais aumentaram de 25 milhões em 1950 para 1,5 bilhões em 2019 [39]. Muitos destes turistas, previamente às suas deslocações, fazem um planeamento da sua viagem, nomeadamente aos locais que querem visitar. A execução deste planeamento pode determinar se uma viagem tem sucesso ou não de um ponto de vista de satisfação turística. Este problema agrava-se quando se trata de um grupo de turistas que planeiam viajar em conjunto, pois adiciona uma maior complexidade e "*stress*" ao problema. Algumas aplicações já tiram proveito deste problema, tentando recomendar locais de visita tendo em conta as classificações feitas por antigos viajantes. No entanto, não existe um sistema centralizado que faça o planeamento da viagem tendo em conta as caraterísticas pessoais do grupo de viajantes e que faça alterações às recomendações tendo em conta o contexto em que as atividades se inserem.

Com esta análise é possível identificar a oportunidade de recomendar o planeamento da viagem para um grupo de turistas e a possível adaptação do mesmo em tempo real, tudo de um modo centralizado sem precisar de grandes ações ou perdas de tempo por parte do turista.

3.1.2 Análise da Oportunidade

Nesta secção analisamos a oportunidade, de modo a entendê-la num ambiente de negócio para podermos compreender melhor como o mercado reagirá para aproveitar a solução criada e assim torná-la num projeto rentável.

Para o caso de estudo presente, será necessário um sistema que permita fazer recomendações de locais turísticos ao utilizador e fazer o seu planeamento tendo em conta o contexto das atividades e do ambiente geral, como, por exemplo, atividade climática e horários de fecho dos locais a visitar. Os sistemas de recomendação de grupo podem alcançar sucesso neste assunto, uma vez que permitem recomendar locais de visita tendo em conta o perfil pessoal do grupo de turistas e, ao mesmo tempo, em coordenação com sistemas de contexto, adaptar este planeamento tendo em conta aspetos climáticos ou temporais.

Os *"pets"* virtuais, embora não tenham uma presença física, tornaram-se cada vez mais populares nos últimos anos, principalmente na forma de aplicações e jogos para dispositivos móveis. Estes demonstraram ter um impacto positivo no bem-estar dos seus utilizadores, que ao interagir com os mesmos demonstram que podem reduzir os níveis de "*stress*" e ansiedade, assim como aumentar o nível de sucesso da passagem de informação. Assim, podemos concluir que existe um mercado para a combinação de sistemas de recomendação para grupos sensíveis ao contexto com os *"pets"* virtuais, de modo a ajudar o utilizador a receber informações contextuais da atividade planeada em tempo real, assim como sugestões de alteração do percurso de um modo mais calmo e menos stressante para o utilizador.

3.1.3 Idealização e Enriquecimento

A idealização e enriquecimento de ideias pode ser relevante de maneira a projetar novos objetivos relacionados com o foco do projeto, criando melhorias ou alterações à ideia original do projeto para proporcionar uma melhor satisfação ao cliente.

Existe uma variedade de ferramentas e plataformas que encaixam no projeto atual. A ideia é usar ferramentas *"open-source"* que sejam boas não só a oferecer algoritmos, mas também nos ambientes de desenvolvimento que se encaixem no planeamento do projeto.

Atualmente existem diversos estudos relacionados com o planeamento turístico que utilizam sistemas com outras metodologias de sistemas de recomendação. Neste projeto inovamos esta abordagem introduzindo os *"pets"* virtuais e as alterações de plano contextuais.

Para esta etapa, após discutir o planeamento do projeto com as pessoas envolvidas, foram estabelecidos os seguintes objetivos:

- Obter uma recomendação para o planeamento dos locais a visitar
- Perceber que "*pets*" virtuais terão mais sucesso de um ponto de vista do utilizador

- Alertar o utilizador através do *"pet"* virtual para mudanças no plano turístico tendo em conta aspetos contextuais da atividade
- Desenvolver uma plataforma amigável para o utilizador de forma a ser menos stressante e fácil de passar informação acerca do planeamento da viagem
- Aplicar o mesmo processo para outras funcionalidades da aplicação desenvolvida

3.1.4 Seleção da Ideia

A seguir à idealização de ideias, deve-se selecionar e priorizar as ideias a serem focadas no desenvolvimento da solução considerando que essa escolha deve respeitar uma relação entre necessidade, benefício e custo.

Após considerar todas as ideias, para esta segunda fase, visto que na primeira fase já foi desenvolvida uma aplicação de recomendação de atrações turísticas, foi definido que a alteração dos pontos de interesse turísticos sugeridos consoante alterações contextuais de última hora e a passagem da informação em tempo real para o utilizador por meio de *"pets*" virtuais será o objetivo a ser perseguido.

Estas ideias foram abordadas considerando o risco tecnológico, os benefícios para o cliente e o custo de desenvolvimento implicado. As outras ideias serão abordadas como trabalho futuro a ser feito.

3.1.5 Definição do Conceito

A etapa final do Novo Modelo de Desenvolvimento de um Novo Conceitos é a definição do conceito. Nesta etapa é feita uma avaliação completa do projeto quanto às necessidades e benefícios do cliente, os mecanismos e metodologias aplicadas, os investimentos necessários, a análise do segmento de mercado contra concorrentes e o risco adjacente ao projeto.

O planeamento turístico é uma área bastante sensível, visto que depende de vários fatores que podem influenciar positivamente ou negativamente a experiência, nomeadamente devido ao número de turistas envolvidos, a personalidade e motivações dos mesmos e as alterações de última hora a que estão sujeitos devido a comportamentos contextuais como alterações climáticas ou horários dos locais turísticos.

Este projeto irá focar-se no alerta destas alterações de última hora ao utilizador através de *"pets"* virtuais e recomendações de locais alternativos que não prejudiquem o plano turístico do mesmo. Desta forma, será possível assistir o utilizador com o seu planeamento turístico por meio de sugestões eficazes e precisas que permitam aumentar o nível de satisfação do mesmo na sua atividade e reduzir todos os fatores negativos associados, como o "*stress*", falta de tempo e falta de organização.

3.2 Proposta de Valor

O método de Proposta de Valor desenvolvido pelo Dr. Alexander Osterwalder é um modelo para analisar a entrada de um produto no segmento de mercado. Este modelo especifica a relação entre dois segmentos importantes: o cliente e a proposta de valor. Este modelo é geralmente aplicado quando um novo produto é oferecido ao mercado do zero [29].

Em relação aos dois segmentos do modelo, podemos identificar alguns submodelos importantes. No segmento de clientes temos:

- ***"Gains"*** – os benefícios que o cliente espera e precisa, o que entusiasmará o cliente e as coisas que aumentam a probabilidade de adotar uma proposta de valor [29].
- **"Pains"** – as experiências negativas, emoções e riscos que o cliente sentira no processo de completar a tarefa [29].
- ***"Customer jobs"*** – as tarefas funcionais, emocionais e sociais que os clientes estão a tentar executar, problemas que estão a tentar resolver e desejos que querem satisfazer [29].

No segmento de proposta de valor, temos:

- ***"Gain creators"*** – como o produto ou serviço gera ganhos para o cliente e como ele oferece valor ao cliente [29].
- ***"Pain relievers"*** – uma descrição de como o produto ou serviço alivia as dores do cliente [29].
- ***"Products and services"*** – *os produtos e serviços que criam ganho, aliviam a dor e que sustentam a criação de valor para o cliente* [29].

Na Figura abaixo, podemos ver a Proposta de Valor aplicada à ideia de negócio que fundamenta o projeto desenvolvido.

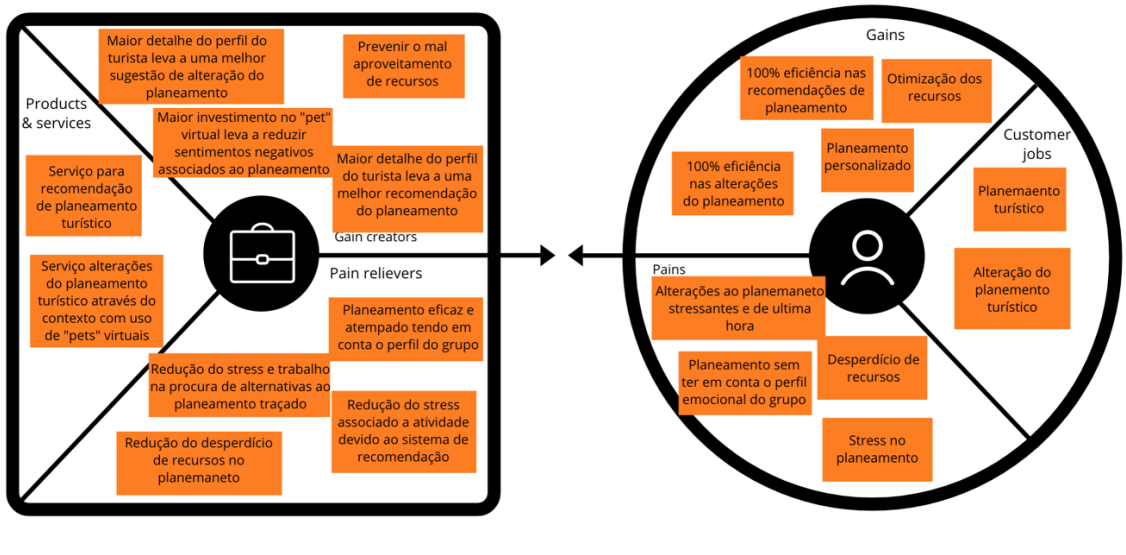


Figura 12 - Proposta de Valor

Resumindo, o objetivo desta ideia de negócio é desenvolver um sistema de recomendação de planeamento turístico em grupo com capacidade de alertar, através de um *"pet"* virtual, para alterações em tempo real do mesmo de acordo com comportamentos contextuais relativos à atividade.

Este sistema irá oferecer ao cliente uma maneira mais simples e eficaz de planear as suas viagens, tendo em conta a sua personalidade, com a possibilidade de alterações de última hora ao planeamento, tendo em conta o contexto das atividades através de um *"pet"* virtual, contribuindo para uma diminuição do "*stress*" associado à atividade por parte do turista e um melhor aproveitamento dos seus recursos para a atividade turística em si.

Com a implementação do *"pet"* virtual para a passagem da mensagem, pretende-se que o cliente fique com uma dedicação e cativação maior para com a plataforma e, ao mesmo tempo diminua certos sentimentos negativos associados as alterações de planos imprevisíveis.

Concluindo, há realmente valor na proposta apresentada visto que para além de conseguir recomendar o planeamento de uma viagem tendo em conta o perfil do turista, levando a um plano mais detalhado e que encaixa nas ideias do mesmo, terá também capacidade de o alterar de uma forma menos stressante e impactante para o turista, tendo em conta o contexto da atividade, para não prejudicar a viagem do mesmo.

3.3 Valor Percecionado

O Valor Percecionado define o que o cliente percebe como benefícios e sacrifícios de um produto comercial [30]. Este valor pode ser medido pela utilidade de um produto para o cliente e pela correlação dos benefícios percebidos com os sacrifícios obrigatórios. Esta definição é

reconhecida como um dos indicadores mais importantes para ganhar vantagem competitiva no mercado [31].

A definição deste conceito pode variar de produto para produto. No entanto, o valor percecionado carrega duas medidas importantes que podem ser usadas para entender o seu verdadeiro significado: comportamental e utilitária. De um ponto de vista utilitário, o valor percecionado pode ser categorizado nas observações de utilidade, transação e aquisição de um produto para o cliente. Do ponto de vista comportamental, a definição do conceito de valor percecionado vem da utilidade de um produto para o cliente em relação aos seus sacrifícios e benefícios.

O serviço prestado com este projeto é baseado num sistema de recomendação de planeamento turístico em grupo com capacidade de alertar, através de um *"pet*" virtual, para alterações em tempo real do mesmo de acordo com comportamentos contextuais relativos às atividades. Tendo em conta o conceito de valor percecionado, podemos entender que a utilidade deste produto é enorme, no sentido de que estes tipos de serviços são raros no mercado. O cliente perceberá que pequenos sacrifícios serão feitos principalmente em relação às despesas financeiras na aquisição e instalação do serviço, embora os benefícios sejam imensos, pois menos recursos serão gastos e o planeamento será mais preciso tendo em conta o perfil do grupo turístico e os fatores contextuais relativos às atividades turísticas.

3.4 Valor para o Cliente

O valor para o cliente advém dos benefícios que emergem do relacionamento da associação pessoal com a oferta do produto comercial. Este conceito traduz-se no aumento de benefícios, redução de sacrifícios ou uma combinação entre ambas as ações.

Como percebemos anteriormente, o valor de um produto decorre da proporção de benefícios e sacrifícios para um cliente durante e após a compra de um produto [32]. Esta avaliação de valor varia de cliente para cliente. A ideia de perceber o valor de um produto é tão frágil que pode variar desde o primeiro contacto do cliente com um produto até a sua utilização. O cliente constrói esta ideia de valor antes de comprar um produto, mas esta pode variar a longo prazo devido a múltiplos fatores, como preço ou usabilidade.

Na tabela abaixo, estão resumidos os benefícios e sacrifícios de um cliente relativamente a este conceito de produto numa perspetiva longitudinal.

Tabela 1 - Valor do Produto de um Ponto de Vista Longitudinal

	Benefícios	**Sacrifícios**
Antes da Compra	• Eficiência • Confiabilidade • Melhores recomendações turísticas	• Preço

Transação		• Custos da Aquisição
Após a Compra	• Suporte • Customização	• Familiarização • Instalação
Utilização	• Recomendação de pontos de interesse turísticos tendo em conta o perfil pessoal do grupo de turistas • Alteração das recomendações em tempo real tendo em conta o contexto das atividades • Otimização de recursos • Diminuição de perdas de tempo e do stress	• Atualizações • Disponibilizar dados pessoais para o sistema

3.5 Modelo de Negócio

Para que o conceito de uma ideia se transforme um negócio, é necessário que o valor proposto ao cliente seja entregue para atingir os objetivos do projeto. É imprescindível que o negócio esteja estruturado para atingir esse objetivo. Para começar, o cliente deve ser claramente identificado (o público-alvo), o problema a ser resolvido deve ser destacado, a solução ou soluções para contornar o problema devem ser projetadas e a estrutura do negócio deve ser definida para obter lucro.

A análise de valor de um produto é extremamente importante, visto que define o porquê de o mercado priorizar o produto em questão em oposição ao que os concorrentes oferecem. O Modelo de Negócio é uma ótima forma de descrever de forma simples e objetiva o plano de negócios de modo a demonstrar a proposta de valor de um produto. Na figura abaixo vê-se este modelo aplicado ao desenvolvimento do produto atual.

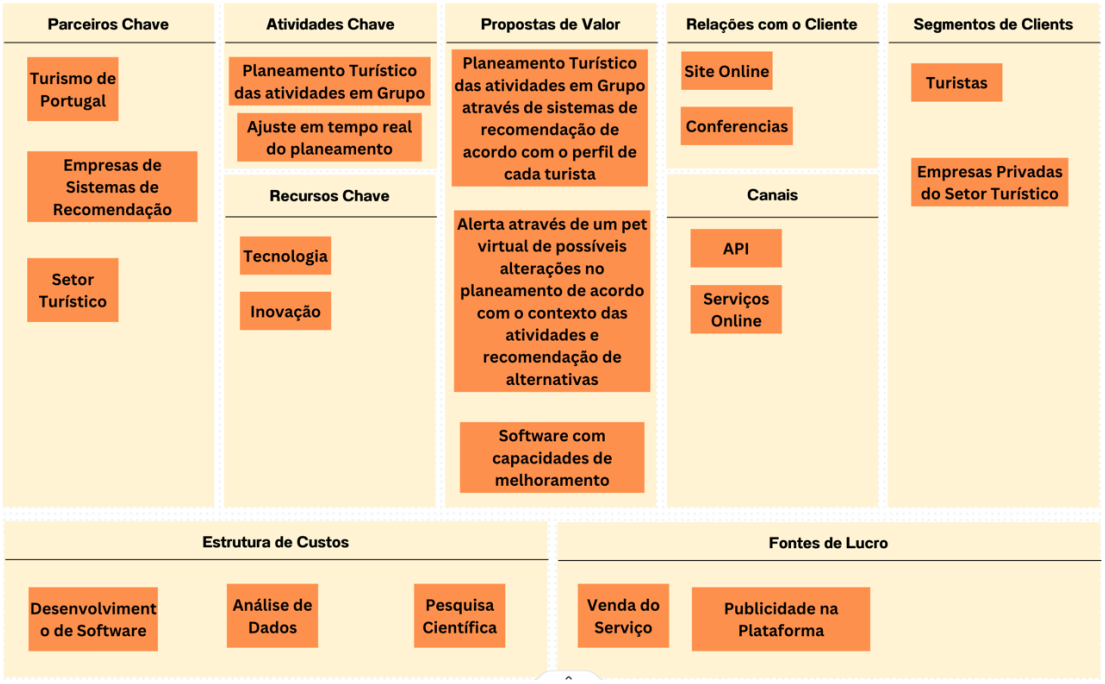


Figura 13 - Modelo de Negócio

Este projeto irá focar-se no alerta destas alterações de última hora ao utilizador por meio de "*pets*" virtuais e recomendações de locais alternativos que não prejudiquem o plano turístico do mesmo. Desta forma, será possível assistir o utilizador com o seu planeamento turístico através de sugestões eficazes e precisas que permitam aumentar o nível de satisfação do mesmo na sua atividade e reduzir todos os fatores negativos associados como o "*stress*", falta de tempo e falta de organização.

Este *"software*" será baseado em sistemas de recomendação de grupo contextuais, que serão selecionados e desenhados através de um estudo de campo clínico, que vão auxiliar a que o processo de recomendação seja o mais preciso possível. Estes sistemas terão por base a personalidade e preferências do turista ou grupo de turistas que será fornecido por técnicas de *"gamification"* do sistema. Quanto aos alertas contextuais e alterações de recomendações em tempo real, estes serão entregues por *"pets"* virtuais que serão desenhados e testados previamente, não só para a definição da sua modulação 3D, mas também para a idealização da biblioteca Android de renderização dos mesmos.

Em termos de custos, este plano de negócios envolve despesas relacionadas com infraestruturas, aquisição de "*software*" e pagamento aos engenheiros que irão supervisionar o desenvolvimento e manutenção do projeto. A possível aquisição e colheita dos dados do perfil dos turistas para o sistema de recomendações também foi considerada.

Este sistema será entregue ao público-alvo através de uma aplicação móvel a ser instalada nos sistemas dos clientes ou por meio de um sistema "*online*" de fácil acesso. As principais receitas deste serviço serão baseadas na venda do mesmo e nos direitos de publicidade que o sistema irá possuir de modo a atrair o cliente para outros negócios paralelos.

3.6 Análise de Valor do Produto

Nesta secção, será focada a análise do valor do produto deste projeto. Primeiramente, prestemos atenção à identificação e análise funcional do produto. Isso significa que as funções mais importantes do produto serão identificadas por meio de uma análise minuciosa. Uma função pode ser definida como o uso mais exigido de uma parte de um produto, bem como o valor estimado que este fornece. Após esta identificação, será feita uma análise dos mesmos, bem como a sua avaliação e comparação [33].

Um dos métodos utilizados nesta análise é o "*Qualify Function Deployment"* (QFD). O QFD é uma ferramenta de qualidade para desenhar um produto segundo os requisitos do cliente, envolvendo no processo todos os membros da organização do produto [34]. Para completar o objetivo deste método, cada função pedida pelo cliente é segmentada e analisada, identificando no caminho as formas de atingir esta segmentação [35].

Para iniciar este método, é imperativo questionar os interessados para identificar os requisitos necessários para o produto e emparelhá-los com uma variável de peso que traduza a importância de cada um deles. Foi determinada a variável de peso para cada requisito quanto à sua importância para o cliente, complexidade e esforço de desenvolvimento, lembrando que a soma de todos esses pesos deve ser 100. Na tabela abaixo podemos observar os requisitos e respetivos pesos do produto do projeto [33].

Tabela 2 - Requerimentos do Cliente e Respetivos Pesos

Requerimentos de Desenvolvimento	**Peso**
Planeamento da Viagem	20
Recolha dos dados do perfil do turista	20
Alteração das recomendações tendo em conta o contexto	20
Renderização de um "*pet*" virtual para passar mensagens de contexto	20
Aplicar a funcionalidade do "*pet*" virtual a outros conceitos	10
Desenvolvimento de uma interface *"user-friendly"*	10

O próximo passo é identificar os requisitos de qualidade que representam como os requisitos do cliente serão atendidos do ponto de vista técnico. Estes requisitos são:

- Sistema de recolha de dados do perfil do turista
- Sistemas de Recomendação para Grupos Sensíveis ao Contexto
- Modelação e renderização de um "*pet*" virtual
- Desenvolvimento de uma Interface *"user-friendly"*

Para a fase de análise funcional, será utilizado o método "*House of Quality*". Este método procura comparar os requisitos de qualidade com os requisitos do cliente, mencionados anteriormente, e avaliar as relações entre eles [36].

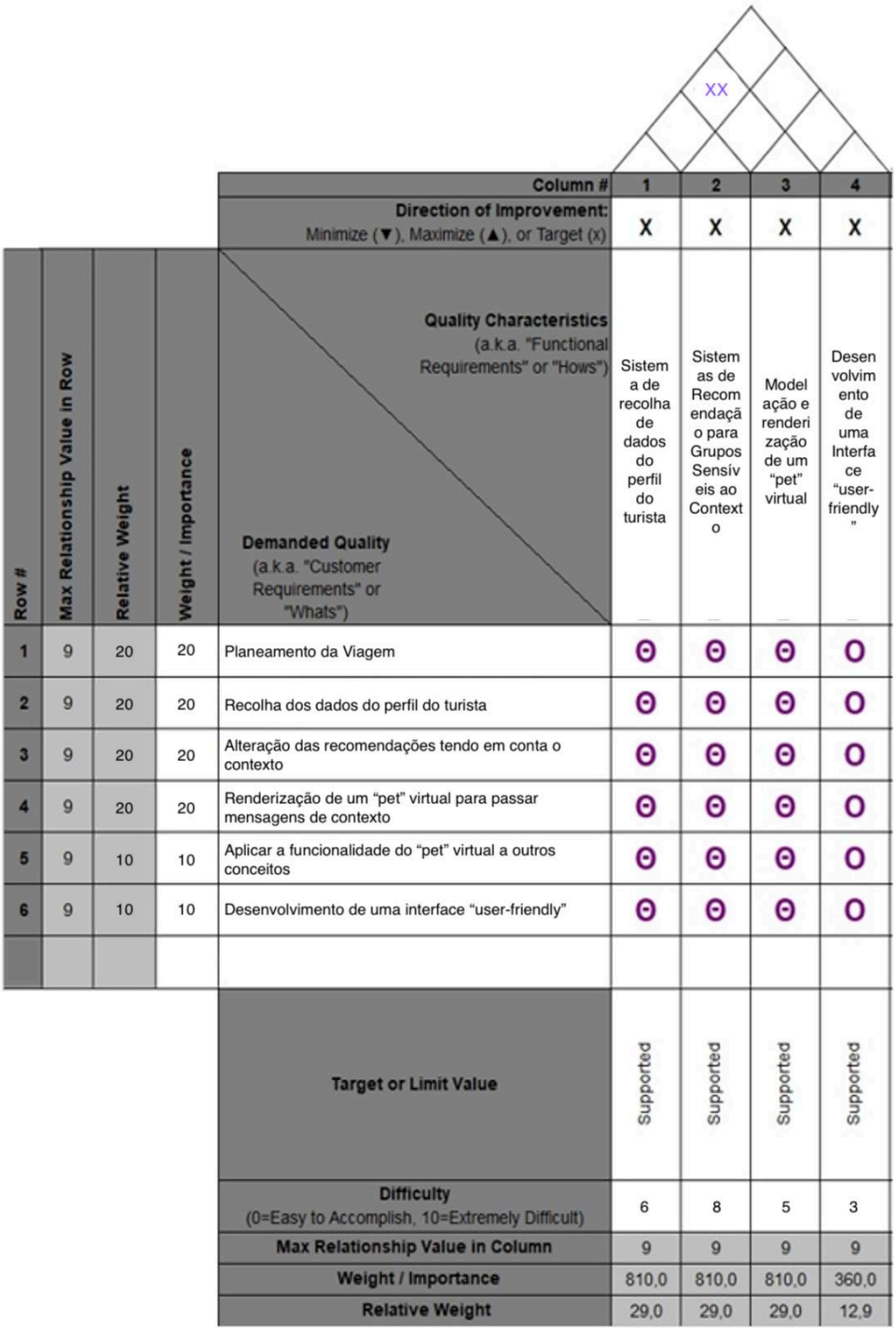


Figura 14 - "*House of Quality*"

Após analisar o modelo acima, podemos entender que todos os requisitos identificados estão fortemente relacionados às tarefas de sistemas de recomendação e implementação do "*pet*" virtual. Por esta razão, o foco da experimentação e desenvolvimento estará centrado nestes assuntos.

Para continuar a análise do produto vamos concentrar-nos agora na identificação de alternativas criativas. Apesar de todos os requisitos identificados pelo cliente, há um que se destaca que é a idealização do "*pet*" virtual. Isto é extremamente importante, pois a modelação do "*pet*" virtual é importante para o sucesso da passagem da mensagem ao cliente e na empatia criada pelo mesmo com o "*pet*". Os modelos de "*pet*" virtuais escolhidos são animais, humanos e monumentos.

Para selecionar um destes modelos como o mais eficiente será utilizado o "*Analytic Hierarchy Process*" (AHP). Na figura abaixo podemos visualizar em forma de árvore de decisão as diferentes alternativas e critérios a serem avaliados no processo de decisão.

Figura 15 - Árvore de Decisão para o Método de AHP

Cada modelo será analisado relativamente a três aspetos principais: facilidade de modulação, facilidade de animação e empatia com o utilizador. A facilidade de modulação prende-se com o grau de facilidade do processo de idealização e modulação 3D do "*pet*" virtual. A facilidade de animação é o grau de complexidade do processo de "*rigging*" e animação do "*pet*" virtual. A empatia com o utilizador compreende-se com o nível de afinidade com que o utilizador terá com cada módulo de "*pet*" virtual.

Após a identificação destes fatores, é importante uma comparação entre eles. É utilizada uma escala numérica que mostra diferentes graus de importância, exemplificando

quantas vezes uma alternativa é mais importante em relação a outra num determinado critério [36].

Tabela 3 - Escala Numérica de Comparação

Nível de Importância	**Definição**
1	Igual Importância
3	Fraca Importância
5	Grande Importância
7	Enorme Importância
9	Absoluta Importância
Outros Valores	Valores Intermédios

Considerando a escala acima, os diferentes fatores foram analisados e atribuídos com um valor desta escala para definir a sua importância.

Tabela 4 - Matriz de Comparação entre Diferentes Critérios

	Facilidade de Modulação	**Facilidade de Animação**	**Empatia com o Utilizador**
Facilidade de Modulação	1	$\frac{1}{3}$	$\frac{1}{5}$
Facilidade de Animação	3	1	$\frac{1}{4}$
Empatia com o Utilizador	5	4	1
Total	9	$\frac{16}{3}$	$\frac{29}{20}$

Após o processo de atribuição dos pesos de importância concluímos que a empatia com o utilizador é a mais importante seguida da facilidade de animação e da facilidade de modulação. A razão por trás disto é manter o foco na criação de um "*pet*" virtual que atinja mais facilmente o objetivo de criar laços com o utilizador para passar a mensagem com maior sucesso.

Para continuar o processo AHP, é necessário normalizar cada valor dos dados dividindo cada valor pela soma da respetiva coluna.

Tabela 5 - Matriz Normalizada de Comparação entre Diferentes Critérios

	Facilidade de Modulação	**Facilidade de Animação**	**Empatia com o Utilizador**
Facilidade de Modulação	$\frac{1}{9}$	$\frac{1}{16}$	$\frac{4}{29}$
Facilidade de Animação	$\frac{1}{3}$	$\frac{3}{16}$	$\frac{5}{29}$
Empatia com o Utilizador	$\frac{5}{9}$	$\frac{3}{4}$	$\frac{20}{29}$

A seguir determina-se o vetor de prioridade para identificar a ordem de importância de cada fator. O processo de cálculo é feito com o valor da média aritmética de cada linha da matriz.

Tabela 6 - Matriz Normalizada de Comparação entre Diferentes Critérios e Respetivos Vetores de Prioridade

	Facilidade de Modulação	**Facilidade de Animação**	**Empatia com o Utilizador**
Facilidade de Modulação	$\frac{1}{9}$	$\frac{1}{16}$	$\frac{4}{29}$
Facilidade de Animação	$\frac{1}{3}$	$\frac{3}{16}$	$\frac{5}{29}$
Empatia com o Utilizador	$\frac{5}{9}$	$\frac{3}{4}$	$\frac{20}{29}$
Vetor de Prioridade	0,10	0,23	0,67

A tabela acima confirma que a empatia com o utilizador é o fator-chave. Em seguida, é necessário verificar a consistência das prioridades sobre grandes amostras de julgamentos completamente aleatórios [36]. Para isso, vai-se calcular a razão de consistência (RC), que deve ser menor que 0,1 para ter prioridades consistentes e confiáveis.

$$RC = \frac{IC}{IR}$$

Equação 1 - Cálculo do RC

O IC é o índice de consistência e IR é o índice de aleatoriedade. Os valores de IC são estabelecidos pelo Laboratório Nacional de Oak Ridge, nos EUA. A tabela abaixo atribui cada valor de IR ao número de critérios [36].

Tabela 7 - Valores de IR definidos pelo Laboratório Nacional de Oak Ridge

1	0

2	0
3	0,58
4	0,9
5	1,12
6	1,24
7	1,32
8	1,41
9	1,45
10	1,49
11	1,51
12	1,48
13	1,56
14	1,57
15	1,59

Após analisar a tabela acima, determinamos que o IR é 0,58. O valor de IC é calculado pela equação apresentada a seguir. "λmax "é o autovalor da matriz e "n" é o número de critérios avaliados.

$$IC = \frac{\lambda\, max - n}{n - 1}$$

Equação 2 - Cálculo do IC

Com base na matriz de prioridade anterior e no vetor de prioridade, é possível calcular o "λmax ", conforme mostrado em baixo.

$$Matriz\ de\ Prioridade\ \times\ Vetor\ de\ Prioridade\ \cong \lambda_{\max} \times Vetor\ de\ Prioridade\ \rightarrow$$

$$\rightarrow \begin{bmatrix} 1 & 0{,}33 & 0{,}20 \\ 3 & 1 & 0{,}25 \\ 5 & 4 & 1 \end{bmatrix} \times \begin{bmatrix} 0{,}10 \\ 0{,}23 \\ 0{,}67 \end{bmatrix} \cong \lambda_{\max} \times \begin{bmatrix} 0{,}10 \\ 0{,}23 \\ 0{,}67 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0{,}31 \\ 0{,}70 \\ 2.10 \end{bmatrix} \cong \lambda_{\max} \times \begin{bmatrix} 0{,}10 \\ 0{,}23 \\ 0{,}67 \end{bmatrix} \leftrightarrow$$

$$\leftrightarrow \lambda_{\max} \cong 3{,}1$$

Substituindo o valor calculado na Equação 2 temos:

$$IC = \frac{\lambda\, max - n}{n - 1} \rightarrow CI = \frac{3{,}1 - 3}{3 - 1} \rightarrow CI = 0{,}05$$

A seguir, substituímos o valor calculado na Equação 1:

$$RC = \frac{IC}{IR} \rightarrow CR = \frac{0{,}05}{0{,}58} \rightarrow CR = 0{,}09$$

Como o valor de RC é menor que 0,1, podemos confirmar que os valores da prioridade de cada fator são consistentes e confiáveis.

Seguindo os procedimentos do método AHP, agora é importante construir uma matriz de comparação para cada critério, considerando cada alternativa [36]. O processo de normalização dos dados e obtenção dos respetivos vetores de prioridade é o mesmo que foi seguido acima.

Tabela 8 – Matriz Inicial, Matriz Normalizada e Vetor de Prioridade para o Critério de Empatia com o Utilizador

Matriz Inicial			
	Animal	**Monumento**	**Humano**
Animal	1	4	2
Monumento	$\frac{1}{4}$	1	$\frac{1}{3}$
Humano	$\frac{1}{2}$	3	1
Total	$\frac{14}{8}$	8	$\frac{10}{3}$
Matriz Normalizada			
	Animal	**Monumento**	**Humano**
Animal	$\frac{8}{14}$	$\frac{1}{2}$	$\frac{3}{5}$
Monumento	$\frac{1}{7}$	$\frac{1}{8}$	$\frac{1}{10}$
Humano	$\frac{2}{7}$	$\frac{3}{8}$	$\frac{3}{10}$
Vetor de Prioridade			
	0,56	0,12	0,32

Tabela 9 – Matriz Inicial, Matriz Normalizada e Vetor de Prioridade para o Critério de Facilidade de Animação

Matriz Inicial			
	Animal	**Monumento**	**Humano**
Animal	1	$\frac{1}{2}$	2

Monumento	2	1	3
Humano	$\frac{1}{2}$	$\frac{1}{3}$	1
Total	$\frac{7}{2}$	$\frac{11}{6}$	6
Matriz Normalizada			
	Animal	**Monumento**	**Humano**
Animal	$\frac{2}{7}$	$\frac{3}{11}$	$\frac{2}{6}$
Monumento	$\frac{4}{7}$	$\frac{6}{11}$	$\frac{3}{6}$
Humano	$\frac{1}{7}$	$\frac{2}{11}$	$\frac{1}{6}$
Vetor de Prioridade			
	0,30	0,54	0,16

Tabela 10 – Matriz Inicial, Matriz Normalizada e Vetor de Prioridade para o Critério de Facilidade de Modulação

Matriz Inicial			
	Animal	**Monumento**	**Humano**
Animal	1	$\frac{1}{2}$	2
Monumento	2	1	3
Humano	$\frac{1}{2}$	$\frac{1}{3}$	1
Total	$\frac{7}{2}$	$\frac{11}{6}$	6
Matriz Normalizada			
	Animal	**Monumento**	**Humano**

Animal	$\frac{2}{7}$	$\frac{3}{11}$	$\frac{2}{6}$
Monumento	$\frac{4}{7}$	$\frac{6}{11}$	$\frac{3}{6}$
Humano	$\frac{1}{7}$	$\frac{2}{11}$	$\frac{1}{6}$
Vetor de Prioridade			
	0,30	0,54	0,16

Estes valores serão importantes para calcular o vetor de prioridade, multiplicando a matriz de valores de cada alternativa pela matriz de valores de cada fator.

$$\begin{bmatrix} 0{,}30 & 0{,}30 & 0{,}56 \\ 0{,}54 & 0{,}54 & 0{,}12 \\ 0{,}16 & 0{,}16 & 0{,}32 \end{bmatrix} \times \begin{bmatrix} 0{,}10 \\ 0{,}23 \\ 0{,}67 \end{bmatrix} = \begin{bmatrix} 0{,}47 \\ 0{,}26 \\ 0{,}28 \end{bmatrix}$$

Ao processar os resultados, conclui-se que o animal é a melhor alternativa.

4 Arquitetura e Design

Este capítulo contém informação sobre o sistema de um ponto estrutural, a análise das funcionalidades desenvolvidas e a arquitetura do sistema.

A seguir, a abordagem para resolver o problema é apresentada. Ainda neste capítulo, é possível visualizar a implementação da solução proposta detalhada sobre cada procedimento efetuado para resolver o problema.

4.1 Modelo de Domínio

No início do projeto, foi apresentado nas reuniões com a coorientadora do GECAD um modelo com os principais conceitos de negócio já implementados na aplicação do GrouPlanner. Na figura seguinte, encontram-se representados esses mesmos conceitos.

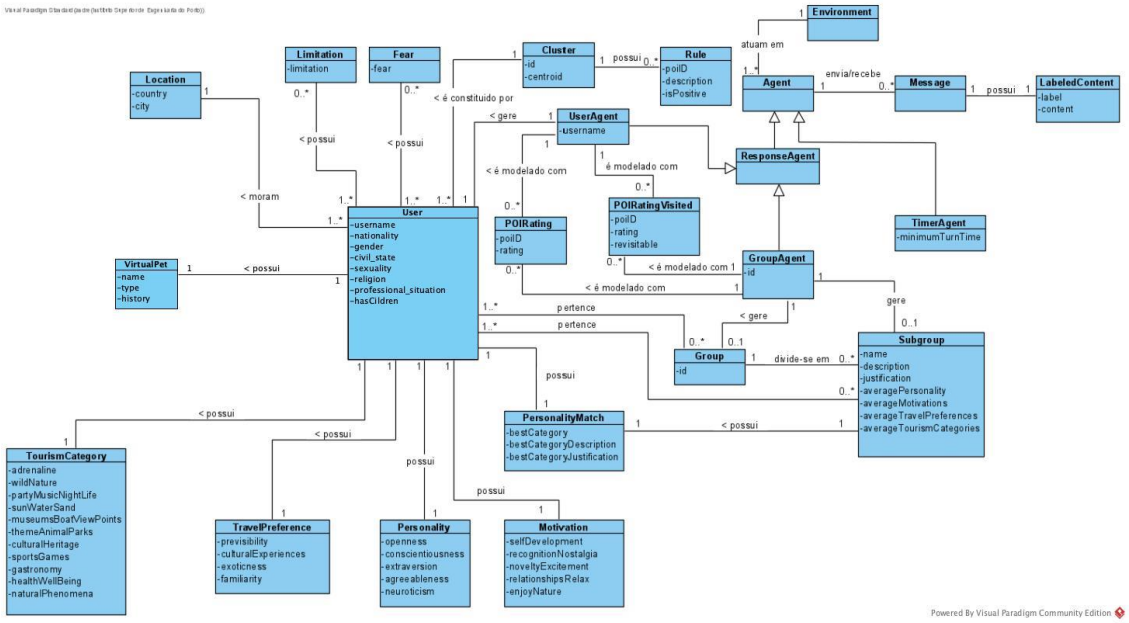


Figura 16 - Modelo de Domínio do GrouPlanner

Analisando a figura, percebemos que para esta dissertação, para além dos conceitos já documentados e desenvolvidos na aplicação GrouPlanner, foi adicionado mais um conceito. O conceito de **Virtual Pet** foi adicionado para representar o "*pet*" virtual que cada utilizador possui. Este mantém uma relação de um para um com o **User**. O **Virtual Pet** possui três atributos:

- **Name** – atributo que guarda o nome do "*pet*" virtual atribuído pelo utilizador que o detém.
- **Type** - atributo que guarda o tipo de "*pet*" virtual escolhido pelo utilizador que o detém de forma a identificar o "*pet*" virtual a renderizar.
- **History** – atributo que guarda um JSON em forma de String com todas as interações entre o "*pet*" virtual e o utilizador. Este JSON guarda cada entrada com a data da interação, mensagem apresentada ao utilizador e a ação que o utilizador tomou.

Desta forma, o **Virtual Pet** é modulado com toda a informação necessária para a renderização do "*pet*" virtual na aplicação, assim como com a possibilidade de armazenamento de toda a informação das interações entre ele e o utilizador que o possui.

Na tabela seguinte encontra-se um glossário do domínio do problema [74].

Tabela 11 - Glossário do Domínio do Problema, adaptado de [74]

Termo	Descrição
Agent	Entidade responsável pela gestão de informação e execução de tarefas que lhe são atribuídas.
Cluster	Aglomerado de utilizadores que possuem personalidades semelhantes em pelo menos 80%, usando a distância Euclidiana normalizada.
Environment	Ambiente onde os Agents atuam.
Fear	Medo/fobia associada a utilizadores.
Group	Grupo de excursão de utilizadores.
GroupAgent	Tipo de ResponseAgent que é responsável pela gestão de informação e execução de tarefas de um Group ou Subgroup.
LabeledContent	Rótulo associado a Messages durante o 35 processo de comunicação.
Limitation	Limitação física associada a utilizadores (e.g, surdez, cegueira, limitação motora, problemas cardíacos).
Location	Residência associada a utilizadores.
Message	Mensagem utilizada no processo de comunicação interna e externa dos Agents.
Motivation	Informação relativa às motivações que levam um dado User a viajar.
Personality	Personalidade associada a um utilizador de acordo com 5 dimensões do modelo Big Five (Abertura à experiência, Conscienciosidade, Extraversão, Agradabilidade, Neuroticismo).
PersonalityMatch	Informação relativa às características de um dado User ou Subgroup. Por exemplo, a melhor categoria de turismo para um dado utilizador com base nos seus dados.
POIRating	Feedback relativo a um POI que ainda não foi visitado por um dado User.
POIRatingVisited	Feedback relativo a um POI que já foi visitado por um dado User.
ResponseAgent	Tipo de Agent que é capaz de comunicar para fora do seu Environment.
Rule	Indicação de um Cluster para priorizar ou dificultar a recomendação de um determinado POI. As regras são geradas com base no feedback fornecido pelos utilizadores de um dado Cluster e pelas suas respetivas características pessoais.
Subgroup	Subgrupo de um determinado Group que agrupou Users com uma Personalidade semelhante.

TimerAgent	Agent adicionado ao Environment no momento da sua criação para prevenir que este termine a sua execução.
TourismCategory	Informação relativa às categorias de turismo preferenciais de um dado User.
TravelPreference	Informação relativa às preferências de viagem 36 de um dado User.
User	Utilizador da aplicação.
UserAgent	Tipo de ResponseAgent que é responsável pela gestão de informação e execução de tarefas de um User.
Virtual Pet (Acrescentado)	"*Pet*" Virtual associado ao utilizador que guarda toda a informação necessária para a renderização do "*pet*" virtual na aplicação assim como com a possibilidade de armazenamento de toda a informação das interações entre ele e o utilizador que o possui

4.2 Requisitos Não Funcionais

Requisitos não funcionais descrevem as características que o sistema deve ter, ao invés de especificar comportamentos específicos. Por outras palavras, enquanto os requisitos funcionais descrevem o que o sistema deve fazer (as suas funções), os requisitos não funcionais descrevem como o sistema realiza essas funções [66].

É importante realçar que os requisitos não funcionais muitas vezes se sobrepõem e, em projetos reais, estes podem ser categorizados de maneira diferente com base nas necessidades e prioridades do projeto. Estes desempenham um papel crucial no "*design*" e na arquitetura do sistema e a sua negligência pode resultar em falhas no sistema ou em expectativas não cumpridas, mesmo que o sistema atenda a todos os seus requisitos funcionais [66].

Neste projeto e na sua implementação foram aplicados e analisados quatro requisitos:

- Requisitos de Usabilidade
- Requisitos de Confiabilidade
- Requisitos de Escalabilidade
- Requisitos de Portabilidade

4.2.1 Requisitos de Usabilidade

Requisitos de usabilidade referem-se às características de um sistema que o tornam fácil de usar, eficaz e agradável de um ponto de vista do utilizador. Estes são parte dos requisitos

não funcionais de um sistema e são essenciais para garantir uma boa experiência do utilizador [66].

Abaixo estão algumas dimensões e exemplos típicos de requisitos de usabilidade:

- **Facilidade de Aprendizagem**: O sistema deve ser fácil para os novos utilizadores compreenderem e começarem a usar [66].
- **Acessibilidade**: O sistema deve ser acessível a todos os utilizadores, incluindo aqueles com deficiências [66].
- **Consistência e Padronização**: As interações e interfaces do sistema devem ser consistentes, permitindo que os utilizadores apliquem o que aprenderam [66].

Ao projetar ou avaliar um projeto, é importante não apenas considerar estes requisitos de usabilidade na teoria, mas também envolver os utilizadores finais no processo, conduzindo testes de usabilidade de forma a melhorar o sistema segundo a opinião dos mesmos [66].

4.2.2 Requisitos de Confiabilidade

Requisitos de confiabilidade referem-se à capacidade de um sistema, "*software*" ou componente de executar a sua função sem falhas e sob determinadas condições num período especificado. Por outras palavras, estes referem-se à capacidade do "*software*" de operar de forma consistente e correta em condições normais e antecipadas [66].

Aqui estão algumas dimensões e características associadas aos requisitos de confiabilidade:

- **Disponibilidade**: Refere-se à proporção de tempo que o sistema está a funcionar e disponível para uso [66].
- **Tolerância a falhas**: Capacidade do sistema de manter a operação mesmo na presença de falhas [66].
- **Recuperabilidade**: Capacidade do sistema de recuperar perante falhas e restaurar os seus níveis de desempenho e funcionalidade [66].
- **Resistência a erros**: Capacidade de um sistema de resistir a erros introduzidos, seja por falhas humanas ou problemas externos [66].

Ao definir e priorizar os requisitos de confiabilidade, é possível projetar sistemas que atendem às expectativas dos utilizadores em termos de desempenho, segurança e robustez [66].

4.2.3 Requisitos de Escalabilidade

Requisitos de escalabilidade referem-se às características que uma aplicação deve possuir para lidar com o aumento de carga sem degradar o seu desempenho ou a experiência

do utilizador. Estes requisitos são essenciais para assegurar que uma aplicação possa crescer e atender a um número maior de utilizadores [66].

Dentro do contexto de aplicações de software, aqui estão algumas dimensões e aspetos chave dos requisitos de escalabilidade:

- **Desacoplamento**: Componentes ou serviços da aplicação são independentes entre si, permitindo que cada um deles escale separadamente [66].
- **Performance e Otimização**: Assegurar que a aplicação tenha tempos de resposta rápidos mesmo sob carga pesada [66].
- **Elasticidade**: Capacidade da aplicação de expandir e contrair recursos automaticamente conforme a carga [66].

Ao desenvolver aplicações de "*software*" escaláveis, é fundamental projetar a arquitetura e a infraestrutura para atender a estes requisitos desde o início. Isto permite que o "*software*" cresça e se adapte com sucesso à medida que a carga aumenta [66].

4.2.4 Requisitos de Portabilidade

Requisitos de portabilidade são aqueles que garantem que um sistema consegue operar em diferentes ambientes ou de ser transferido de um ambiente para outro com esforço mínimo [66]. E podem abranger:

- **Adaptabilidade**: A capacidade do "*software*" se adaptar a diferentes ambientes sem a necessidade de mudanças [66].
- **Coexistência**: A habilidade do "*software*" coexistir com outras aplicações no mesmo ambiente, sem interferir no seu funcionamento [66].
- **Substituibilidade**: A capacidade do "*software*" ser substituído por outro software similar num ambiente específico, sem problemas [66].

Ao especificar requisitos de portabilidade, as organizações podem garantir que o seu "*software*" seja flexível, abrangente e capaz de atender a diferentes públicos e ambientes. Isto é especialmente importante para produtos de "*software*" que procuram alcançar um público global ou que podem ser utilizados em ambientes tecnológicos diversificados [66].

4.3 Requisitos Funcionais

Um requisito funcional é uma descrição da funcionalidade que o "*software*" tem para oferecer. Descreve um sistema dentro do projeto ou componente do mesmo. Um requisito não é nada mais é do que os *"inputs"* para o sistema, o processamento dos mesmos e os *"outputs*" [65]. Na figura abaixo podemos ver um diagrama de caso de uso, ilustrando-os.

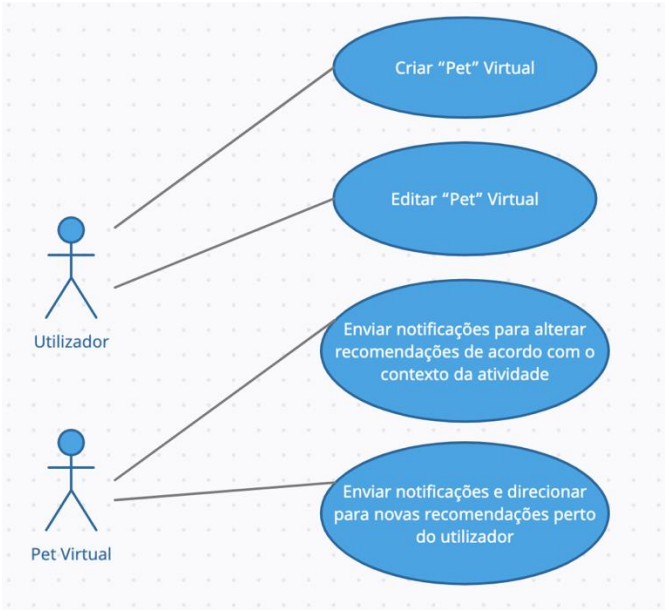


Figura 17 - Diagrama de Casos de Uso

Após a análise do diagrama acima, podemos identificar dois tipos de atores. O "*Pet*" Virtual sendo que todos os casos de uso a que é associado são automatizados e não necessitam de nenhuma ação por parte do utilizador, e o Utilizador, cujos casos de uso precisam da sua intervenção.

Na tabela seguinte, descreve-se detalhadamente cada um dos casos de uso referidos no diagrama acima.

Tabela 12 - Descrição dos Casos de Uso

Caso de Uso	Descrição
UC1 - Criar o "*pet*" virtual	O Utilizador deve submeter, no formulário de criação de conta, os campos de tipo e nome do "*pet*" virtual para serem utilizados na renderização do "*pet*" virtual, permitindo a personalização do mesmo por parte do utilizador da aplicação.
UC2 - Editar o "*pet*" virtual	O Utilizador deve submeter, no formulário de edição do "*pet*" virtual, os campos de tipo e nome do "*pet*" virtual para editar as seleções efetuadas previamente. O Utilizador deve também ter acesso ao registo de interações do "*pet*" virtual.
UC3 - Enviar notificações para alterar recomendações de acordo com o contexto da atividade	O "p*et*" Virtual deve sugerir novas recomendações ao utilizador para substituir outras na sua lista de atividades recomendadas individuais ou de grupo que não sejam possíveis visitar devido a entraves

	no contexto das mesmas (por exemplo, más condições atmosféricas).
UC4 - Enviar notificações e direcionar para novas recomendações perto do utilizador	O "*pet*" Virtual deve sugerir e direcionar o utilizador a pontos de interesse perto dele que se encaixem no seu perfil preferencial de atividades.

4.4 Funcionalidades do Sistema

Nesta secção são contextualizadas algumas das principais funcionalidades já implementadas no sistema GrouPlanner visto que é importante perceber estes processos para perceber as funcionalidades implementadas neste trabalho.

Antes de referir essas funcionalidades, existem alguns conceitos, nomeadamente micro serviços, que devem ser explicados de forma compreender cada funcionalidade [62]:

- **Micro serviço de Multiagentes –** micro serviço responsável por gerir e disponibilizar a criação e interação de agentes com os utilizadores da aplicação GrouPlanner. Estes agentes representam os turistas na aplicação e possuem várias características, nomeadamente a possibilidade de interagirem socialmente entre eles, auxiliando as funcionalidades de recomendações.
- **Micro serviço do Motor de Recomendações –** micro serviço responsável por receber pedidos de agentes para, através do perfil e personalidade do utilizador ou utilizadores em questão, sugerir pontos de interesses que melhor se adequam aos parâmetros de entrada, usando uma base de dados de pontos de interesse do norte de Portugal.
- **Micro serviço de Gestão de Utilizadores –** micro serviço responsável por guardar e gerir toda a informação de perfil de cada utilizador criado na aplicação Grouplanner.

Existem duas funcionalidades importantes no sistema que necessitam de ser referidas para contextualizar as funcionalidades descritas nas secções seguintes: a funcionalidade de recomendações individuais e a funcionalidade de recomendações de grupo.

No que toca às recomendações individuais, uma mensagem é enviada diretamente ao agente turístico do micro serviço multiagente quando um turista solicita recomendações de pontos de interesse a visitar. Após obter acesso aos detalhes do turista, o agente turístico compila o perfil do turista e solicita ao micro serviço do motor de recomendações os dez pontos de interesse mais relevantes que correspondem melhor à personalidade do visitante. O micro serviço do motor de recomendações fornece ao agente turístico uma lista de pontos de interesse sugeridos. Depois disso, o agente usa as suas habilidades sociais para perguntar aos outros agentes se há melhores recomendações baseadas no seu conhecimento atual. O utilizador que iniciou o pedido recebe então as recomendações finais [62].

Figura 18 – GrouPlanner: Ecrã com Exemplo de uma Lista de Recomendações Individuais

Quando um turista cria um grupo de excursão, tornando-se líder do grupo, as informações do grupo são enviadas ao micro serviço multiagente, que cria imediatamente um agente com os dados do grupo, responsável pelo mesmo [62].

Quando o líder do grupo solicita recomendações para o grupo, é enviado um pedido que divide o grupo em subgrupos com personalidades semelhantes, ou seja, um grupo pode ser dividido em dois ou mais subgrupos, dependendo da personalidade dos turistas [62].

Os subgrupos são enviados ao micro serviço do motor de recomendações seguindo o processo descrito na funcionalidade de recomendações individuais. Após distribuir os dez pontos de interesse mais compatíveis para cada subgrupo, o serviço responde ao agente com uma lista de pontos de interesse sugeridos. Após isso, o agente efetua a mesma comunicação com os outros agentes da mesma forma que foi descrito nas recomendações individuais. A lista final de pontos de interesse é enviada pelo agente ao GrouPlanner, que cria os subgrupos propostos e atribui os respetivos turistas. Cada subgrupo pode então consultar a respetiva lista de POI recomendados [62].

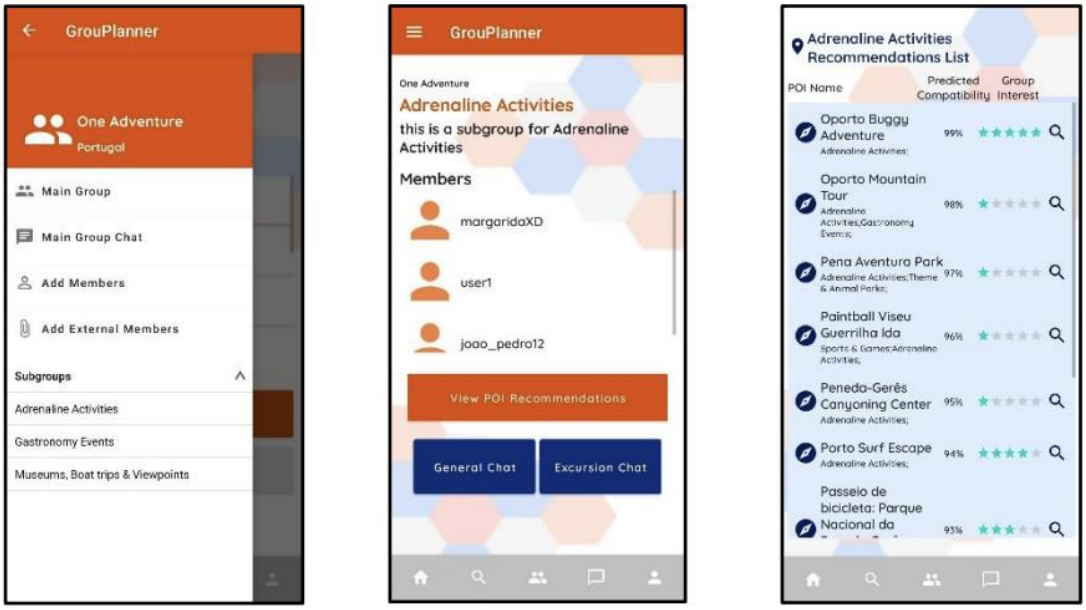


Figura 19 - Ecrãs da Funcionalidade de Solicitação de Novas Recomendações de Grupo

4.5 Arquitetura do Sistema

Nesta secção, é explicada a arquitetura do sistema GrouPlanner existente e que alterações foram efetuadas no mesmo para o desenvolvimento deste projeto, assim como as respetivas justificações para a abordagem adotada.

A aplicação GrouPlanner é um sistema de recomendação para grupos de turismo baseado em micro serviços, sendo um deles, o micro serviço multiagente, que permite a comunicação direta com os agentes dos turistas usando "*endpoints*" REST, tirando proveito das habilidades sociais destes para interagirem entre si e ajudar a refinar as recomendações individuais e de grupo [61].

Cada turista é representado por um agente, modelado com o perfil do turista (dados demográficos, personalidade, preferências e preocupações relacionadas à viagem), que é acessível por meio de um "*endpoint*" REST. O objetivo é expor os agentes turísticos como recursos através de uma REST API, de modo que as informações sobre os turistas possam ser compartilhadas de forma fácil e rápida. Os agentes podem partilhar os seus conhecimentos e dados de interação, interna e externamente, e colaborar para fornecer recomendações mais precisas e satisfatórias aos turistas, reduzindo os conflitos de grupo, sem a necessidade de um agente corretor [62].

.NET é uma das "*frameworks*" de programação orientada a objetos mais utilizadas para implementação de serviços web e aplicativos de plataforma cruzada e, portanto, foi usada para implementar o protótipo GrouPlanner. Quanto ao desenvolvimento do micro serviço multiagente, foi utilizada a "*framework*" ActressMAS.

O ActressMAS é uma "*framework*" "*open-source*" utilizada no desenvolvimento de sistemas multiagente que visa reduzir todas as complexidades inerentes às configurações e à aprendizagem das linguagens utilizadas pelos agentes que variam de tecnologia para tecnologia [63]. Além disso, a "*framework*" foi desenvolvida para ser compatível com .NET devido à escassez de "*frameworks*" compatíveis com este ambiente de desenvolvimento, necessário para o desenvolvimento dos micro serviços do protótipo [63].

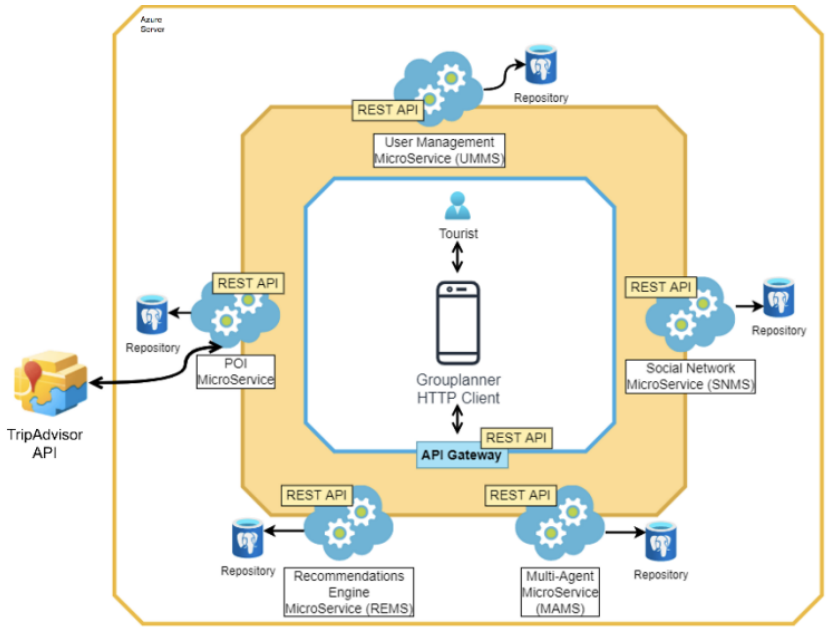


Figura 20 – Arquitetura em Micro Serviços do GrouPlanner, retirado de [62]

No que toca às funcionalidades desenvolvidas, foram efetuadas algumas alterações ao nível da arquitetura do sistema, maioritariamente ao nível da aplicação Android. Fora da aplicação Android e relativamente ao sistema de micro serviços, nem todos são relevantes para as novas iterações realizadas.

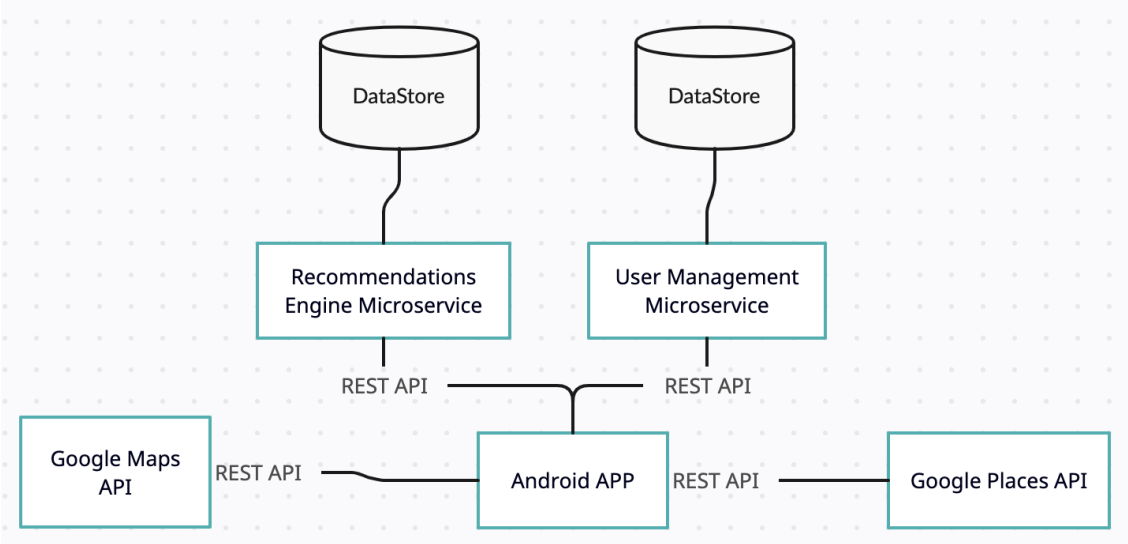


Figura 21 - Arquitetura Detalhada do Sistema Implementado para o "*Pet*" Virtual

Como se pode observar na figura 21, para a implementação destas funcionalidades apenas foram relevantes os micro serviços já existentes de gestão de utilizadores e do motor de recomendações.

No que toca ao micro serviço de gestão de utilizadores, a justificação é simples e prende-se com o facto de ser necessário guardar informações vitais para a renderização do "*pet*" virtual, nomeadamente o tipo, nome e registo de atividade do mesmo. Tendo em conta a arquitetura atual do sistema, só faz sentido guardar esta informação neste micro serviço uma vez que toda a informação do utilizador é guardada lá e assim, é reutilizado o serviço sem a necessidade da criação de um novo para o armazenamento desta informação.

Relativamente ao motor de recomendações, a justificação também é bastante simples. Uma vez que foi necessário criar recomendações e analisar o contexto das atividades já recomendadas, foi necessário reutilizar algumas funcionalidades deste micro serviço para obter os resultados pretendidos, em vez de criar funcionalidades do mesmo calibre na aplicação Android, o que resultaria numa sobrecarga da aplicação.

Em ambos os casos, a troca de informação é efetuada por meio de pedidos HTTP REST, sendo que em alguns casos foram modificados alguns "*endpoints*" já existentes e noutros a criação de novos para obter os resultados desejados.

No que toca a novos componentes na arquitetura, destacam-se as API da Google, Google Maps e Google Places. Ambos os componentes utilizam pedidos HTTP REST para a troca de informação direta com a aplicação Android. Ambas as API estão diretamente relacionadas com a funcionalidade de receber notificações do "*pet*" virtual e direcionar para novas recomendações de Pontos de Interesse (POI) perto dele. Como tal, estas chamadas são efetuadas com alguma regularidade para detetar pontos de interesse relevantes em torno do utilizador. Esta latência de chamadas foi a principal razão pela qual se efetuou uma ligação direta entre a aplicação Android e as API. Desta forma, existe apenas uma chamada realizada

num intervalo definido, enquanto se existisse um micro serviço intermediário, a velocidade e desempenho poderiam ser afetadas.

Relativamente à aplicação Android existente, esta adota uma estrutura convencional relativamente parecida aos modelos "*Model-View-Controller*".

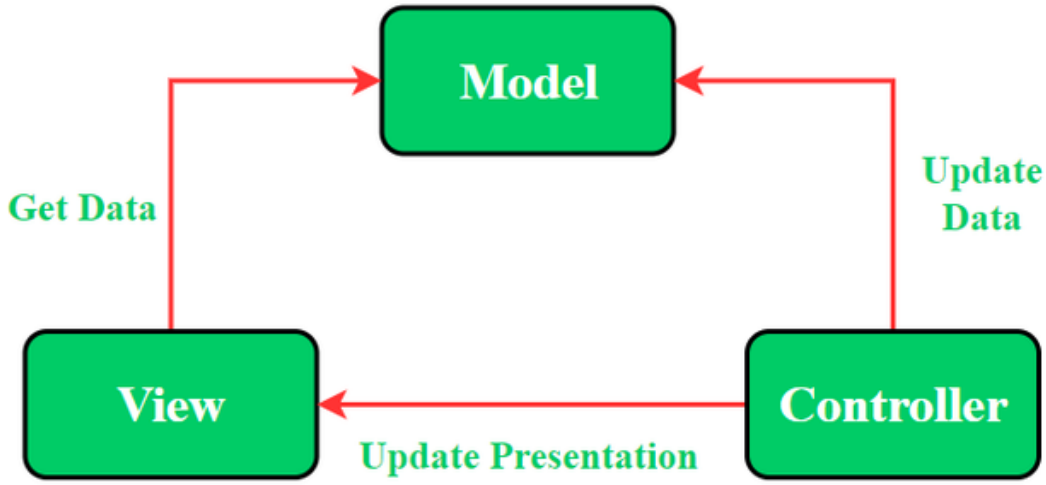


Figura 22 – Exemplo do Modelo "*Model-View-Controller*", retirado de [83]

Esta possui atividades e fragmentos que funcionam como "*controllers*" que detêm a lógica para manipular as "*views*" apresentadas ao utilizador. Estas atividades recolhem e atualizam os dados dos micro serviços que funcionam como o "*model*" da aplicação por meio de chamadas HTTP REST. Para efetuar estas chamadas aos micro serviços, é utilizada uma "interface" denominada Retrofit. A API Retrofit é composta pelo uso de outras API para garantir aquilo que ela promove de melhor, a conexão e consumo de serviços padrão RESTful [64]. Esta estratégia permitiu ter disponível implementações e controlos abstraídos para gerir a conexão, por meio de anotações nos métodos que assinam as chamadas nas "*interfaces*" escritas, ficando ao cargo do Retrofit o trabalho de lidar com as chamadas diretas às API que incorpora [64].

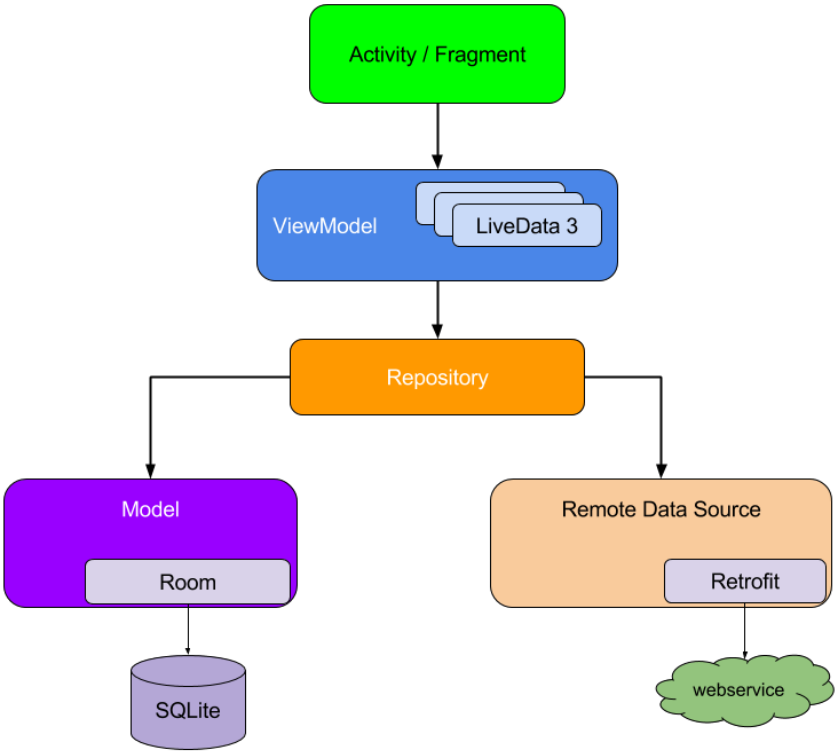


Figura 23 - Arquitetura da Aplicação Android, retirado de [84]

As novas funcionalidades tiram proveito desta arquitetura, já implementada com a adição de novos elementos. No entanto, a grande alteração nesta aplicação prende-se com a adição da biblioteca min3d. Esta biblioteca acompanha de perto a API OpenGL ES, o que a torna ideal para obter uma compreensão da mesma, enquanto fornece a conveniência de uma biblioteca de classes orientada a objetos. A figura seguinte apresenta uma vista detalhada da arquitetura de implementação dos novos componentes no "*software*" da aplicação Android do GrouPlanner relativos ao trabalho desenvolvido.

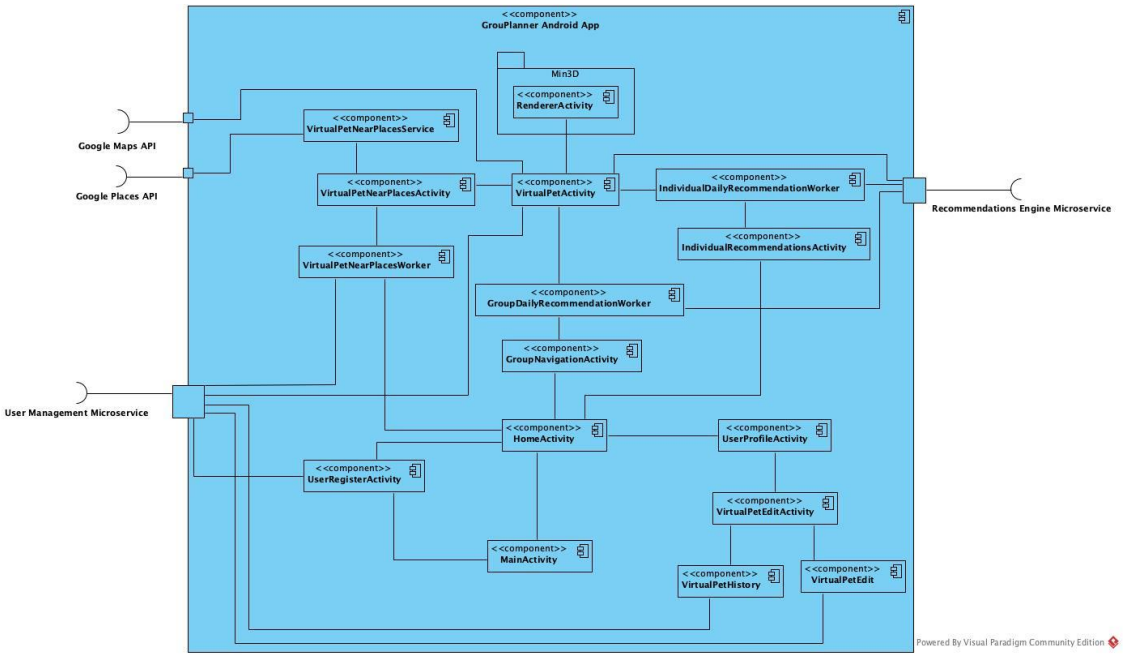


Figura 24 - Arquitetura do "*Pet*" Virtual na Aplicação Android

4.6 Design e Implementação

Nesta secção, para cada um dos requisitos denominados acima, é feita a análise do "*design*" escolhido, assim como detalhes específicos do código de implementação de cada um com as respetivas justificações para as decisões tomadas. Em termos de "*design*", mostra-se um diagrama de sequência assim como um diagrama de classes para explicar o desenho. Em termos de implementação, são exemplificados pedaços de código assim como a justificação e explicação dos mesmos para o objetivo da funcionalidade.

4.6.1 Criar "*Pet*" Virtual

Como já foi anunciado acima, esta funcionalidade procura guardar as configurações do "*pet*" virtual do utilizador no momento de registo de conta por parte do mesmo. Para isso, foram efetuadas algumas alterações ao processo de registo já implementado na aplicação Android do protótipo.

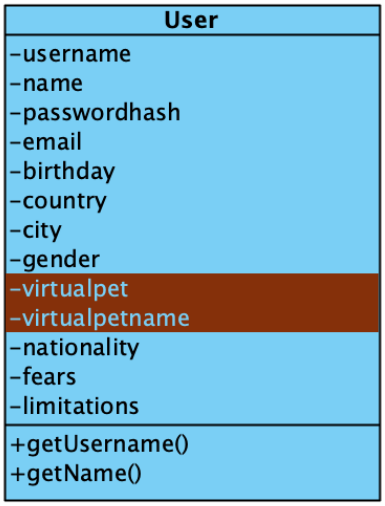


Figura 25 - Atributos Utilizados na Funcionalidade de Criação do "*Pet*" Virtual

Analisando a figura 25, dois novos atributos foram criados para a classe User:

- **Virtualpet** – atributo do tipo String para guardar o tipo de "*pet*" virtual a renderizar para o utilizador. O utilizador tem a opção de escolher entre um macaco e um urso.
- **Virtualpetname** – atributo do tipo String para guardar nome do "*pet*" virtual.

A criação destes atributos foi importante para o funcionamento do processo de renderização do "*pet*" virtual. O "*virtualpet*" é importante, pois é neste elemento que se define o modelo 3D a ser importado e renderizado para o utilizador na atividade do "*pet*" virtual. Estas alterações foram efetuadas não só na aplicação Android, mas também no repositório de utilizadores e respetivo micro serviço.

Na figura no Anexo A, é ilustrado o "*design*" da funcionalidade de criação de perfil do utilizador. Aquando da abertura da aplicação por parte do utilizador, a "*Main Activity*" despoleta as funções "*onCreate*" e "*onStart*" que criam a "*view*" de "*login*" e validam se o utilizador já se encontrava com a sessão iniciada para efetuar o "*login*" automático do utilizador. Após carregar no botão de registo, a "*User Registry Activity*" é inicializada construindo a "*view*" de registo para o utilizador.

Nesta atividade algumas alterações foram efetuadas para implementar a funcionalidade descrita no que toca ao método "*onCreate()*", assim como o documento XML.

```
<LinearLayout
    android:id="@+id/radioVirtualPetsButtons_Container"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_marginTop="0dp"
    android:orientation="vertical">

    <RadioGroup
        android:id="@+id/radioGroup_VirtualPet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/radioButton_Bear"
            android:layout_width="76dp"
            android:checked="true"
            android:layout_height="wrap_content"
            android:layout_marginStart="0dp"
            android:layout_marginLeft="40dp"
            android:buttonTint="#E14B00"
            android:textColor="@color/black"
            android:text="Bear" />

        <RadioButton
            android:id="@+id/radioButton_Monkey"
            android:layout_width="116dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="40dp"
            android:layout_marginLeft="40dp"
            android:buttonTint="#E14B00"
            android:text="Monkey"
            android:textColor="@color/black" />
    </RadioGroup>
```

Figura 26 - XML do atributo "*Virtual Pet Type*" da "*View*" de Registo

```
    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_centerInParent="true"
        android:layout_marginTop="20dp">

        <ImageView
            android:id="@+id/virtualPetImageView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@drawable/bear"/>

    </FrameLayout>
</LinearLayout>
```

Figura 27 – XML do "*Layout*" para Ilustrar o "*Pet*" Virtual Escolhido

```xml
<TextView
    android:id="@+id/txt_VirtualPetName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:fontFamily="@font/quicksand_medium"
    android:text="Virtual Pet Name *"
    android:textColor="#E14B00"
    android:textSize="18dp"
    app:layout_constraintBottom_toTopOf="@+id/dataInput_VirtualPetName"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.030"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/radioVirtualPetsButtons_Container"
    app:layout_constraintVertical_bias="1.0" />

<EditText
    android:id="@+id/dataInput_VirtualPetName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:backgroundTint="@android:color/darker_gray"
    android:ems="10"
    android:fontFamily="@font/quicksand_medium"
    android:inputType="textPersonName"
    android:textColor="#000000"
    android:textCursorDrawable="@null"
    android:textSize="14dp"
    app:layout_constraintBottom_toTopOf="@+id/txt_Name"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="@+id/txt_VirtualPetName"
    app:layout_constraintTop_toBottomOf="@+id/txt_VirtualPetName" />
```

Figura 28 - XML do atributo "*Virtual Pet Name*" da "*View*" de Registo

Analisando as figuras acima, para o caso do "*VirtualPetName*", foi criado um título do tipo "*TextView*" a acompanhar o componente "*EditText*" que resulta num campo editável do tipo String para receber o nome do "*pet*" virtual. No caso do "*VirtualPetType*", foi criado um "*LinearLayout*" para receber um "*RadioGroup*" e um "*FrameLayout*". O "*RadioGroup*" é constituído por dois tipos de "*RadioButton*": um para o modelo do macaco e outro para o modelo do urso. O "*FrameLayout*" é constituído por uma "*ImageView*" com o objetivo de ilustrar o "*pet*" virtual escolhido pelo utilizador.

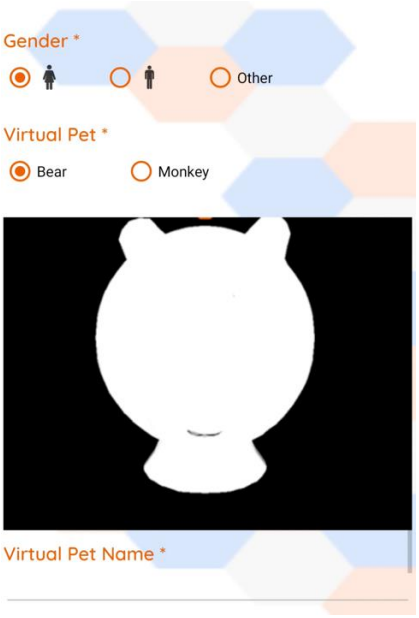


Figura 29 - Ecrã de Registo do Utilizador onde pode Escolher o "*Pet*" Virtual

Todos estes campos, com exceção do titulo dos campos, são mapeados na função "*onCreate()*" da atividade para depois extrair a informação inserida pelo o utilizador.

```
radioButton_Bear = findViewById(R.id.radioButton_Bear);
radioButton_Monkey = findViewById(R.id.radioButton_Monkey);
radioGroup_VirtualPet = findViewById(R.id.radioGroup_VirtualPet);
dataInput_VirtualPetName = findViewById(R.id.dataInput_VirtualPetName);
virtualPetImageView = findViewById(R.id.virtualPetImageView);
```

Figura 30 - Mapeamento dos Campos de Registo do "*Pet*" Virtual

A imagem do "*ImageView*" é alterada de acordo com o "*RadioButton*" selecionado pelo utilizador. Para isso uma função "*setOnCheckedChangeListener()*" foi criada para captar o evento de mudança de seleção do botão e alterar a imagem do "*ImageView*" para a correspondente ao botão selecionado.

```
radioGroup_VirtualPet.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
{
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        int virtualPetId = radioGroup_VirtualPet.getCheckedRadioButtonId();

        if (radioGroup_VirtualPet.findViewById(virtualPetId).equals(radioButton_Bear)) {
            virtualPetImageView.setBackgroundResource(R.drawable.bear);
        } else if (radioGroup_VirtualPet.findViewById(virtualPetId).equals(radioButton_Monkey)) {
            virtualPetImageView.setBackgroundResource(R.drawable.monkey);
        }
    }
});
```

Figura 31 - Código de Alteração da Imagem de Ilustração do "*Pet*" Virtual

Ao completar o registo, o evento "*onClick()*" é despoletado, construindo um novo objeto do tipo User onde os vários valores dos campos de registo são captados.

```
User user = new User(dataInput_Username.getText().toString(), dataInput_Name.getText().toString(), dataInput_Password.getText().toString(),
        dataInput_Email.getText().toString(), dataInput_BirthDate.getText().toString(), select_nationality.getSelectedItem().toString(),
        list_countries.getSelectedItem().toString(), list_cities.getSelectedItem().toString(), getGender(),fearsListSend,
        limitationsListSend, getVirtualPet(), dataInput_VirtualPetName.getText().toString());
```

Figura 32 - Construção do Objeto "*User*"

No que toca ao nome do "*pet*" virtual, este é captado diretamente do campo mapeado através do método "*getText()*". No que toca ao tipo de "*pet*" virtual, este é resgatado através do método "*getVirtualPet()*".

```
public String getVirtualPet() {
    int genderId = radioGroup_VirtualPet.getCheckedRadioButtonId();
    if (radioGroup_VirtualPet.findViewById(genderId).equals(radioButton_Bear)) {
        return "Bear";
    } else if (radioGroup_VirtualPet.findViewById(genderId).equals(radioButton_Monkey)) {
        return "Monkey";
    } else {
        return "0";
    }
}
```

Figura 33 - Método "*getVirtualPet()*" do Registo

Este método utiliza o campo do "*RadioGroup*" para identificar o "*id*" do "*RadioButton*" selecionado e depois compara com os dois botões previamente mapeados para definir qual deles foi selecionado pelo utilizador para retornar a String identificadora do tipo de "*pet*" virtual.

Após o objeto User ser criado, este é utilizado na função "*registerUser()*" para enviar a informação para o micro serviço de gestão de utilizadores para criar uma entidade na base de dados. Recebendo uma resposta de sucesso por parte do micro serviço, uma nova atividade é despoletada, a atividade "*User Account Validation*" onde o utilizador tem de introduzir o código de validação que recebeu no email proveniente do método "*onCreate()*" da mesma. Introduzindo o código correto, o evento "*onClick()*" é acionado para criar um objeto "*UserValidation*" que é enviado através do método "*validateUser()*" para o micro serviço para validar o código inserido. Em caso de sucesso, um novo "*token*" da Google é criado para o utilizador novo através da função "*generateTokenFromGoogleToken()*" para ser possível o "*auto login*" por parte da aplicação quando o utilizador revisitar a aplicação.

Após o sucesso da sequência de eventos, a atividade "*Home*" é acionada para mostrar ao utilizador o ecrã principal da aplicação.

No caso de erro de qualquer uma das chamadas ao servidor descritas anteriormente, uma mensagem de erro é mostrada ao utilizador, impedindo-o de avançar no processo de registo e validação.

4.6.2 Editar "*Pet*" Virtual

Esta funcionalidade pretende editar as preferências selecionadas pelo utilizador no momento do registo, assim como consultar todas as interações que o utilizador teve com o "*pet*" virtual. Para executar esta funcionalidade, algumas alterações foram efetuadas na funcionalidade de editar o perfil do utilizador já existente.

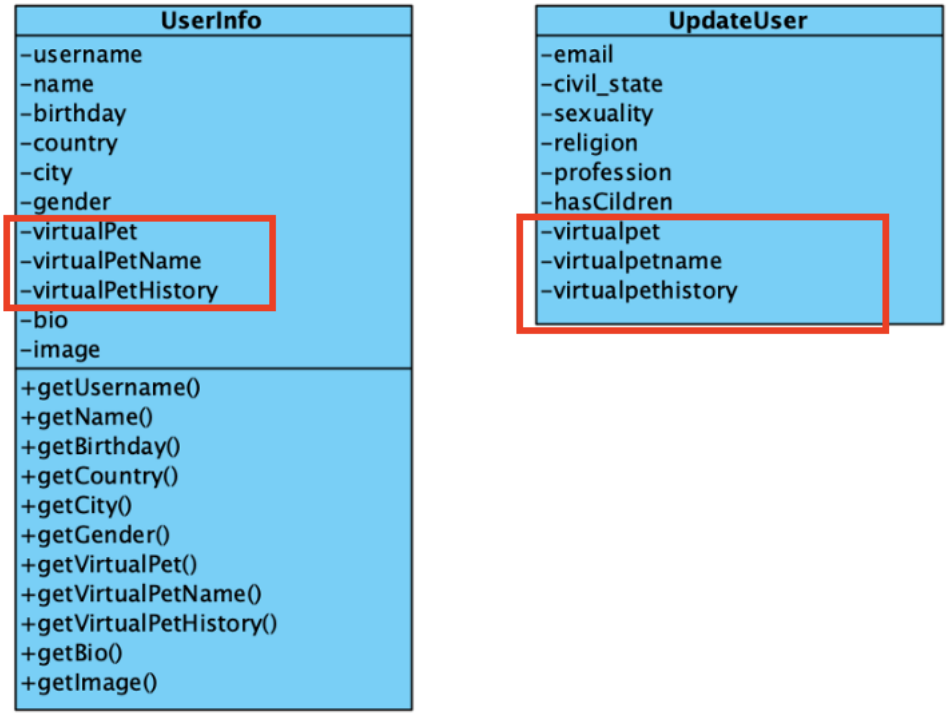


Figura 34 - Atributos Utilizados na Funcionalidade de Edição do "*Pet*" Virtual

Observando a figura 34, é percetível a adição de novos atributos nas classes apresentadas acima. No que toca a classe "*UserInfo*", utilizada para construir um objeto com a informação resgatada do micro serviço de gestão de utilizadores, temos a adição dos seguintes atributos:

- **virtualPet** – atributo do tipo String que recebe o tipo de "*pet*" virtual a ser renderizado. Possui também um método "*getVirtualPet()*" para obter a informação guardada no atributo.
- **virtualPetName** - atributo do tipo String que recebe o nome do "*pet*" virtual a ser renderizado. Possui também um método "*getVirtualPetName()*" para obter a informação guardada no atributo.
- **virtualPetHistory** - atributo do tipo String que recebe um JSON das interações entre o utilizador e o "*pet*" virtual com a respetiva data, informação e resposta dada pelo utilizador. Possui também um método "*getVirtualPetHistory()*" para obter a informação guardada no atributo.

No que toca à classe "*UpdateUser*", utilizada para atualizar a informação do perfil do utilizador no micro serviço de gestão de utilizadores, temos a adição dos seguintes atributos:

- **virtualPet** - atributo do tipo String que guarda o novo tipo de "*pet*" virtual a ser atualizado no micro serviço.
- **virtualPetName** - atributo do tipo String que guarda o novo nome de "*pet*" virtual a ser atualizado no micro serviço.

- **virtualPetHistory** - atributo do tipo String que guarda um JSON das interações entre o utilizador e o "*pet*" virtual com a respetiva data, informação e resposta dada pelo utilizador.

Anteriormente já foram enumeradas as razões por detrás da criação destes atributos na classe "User" sendo que, no caso do "*UserInfo*" e "*UpdateUser*", a razão mantém-se visto que é necessário alterar estas classes para garantir o fluxo de informação entre aplicação e micro serviço, de modo a ser possível implementar a funcionalidade.

No que toca a esta funcionalidade, vários processos são inerentes à mesma e por isso são analisados separada e detalhadamente.

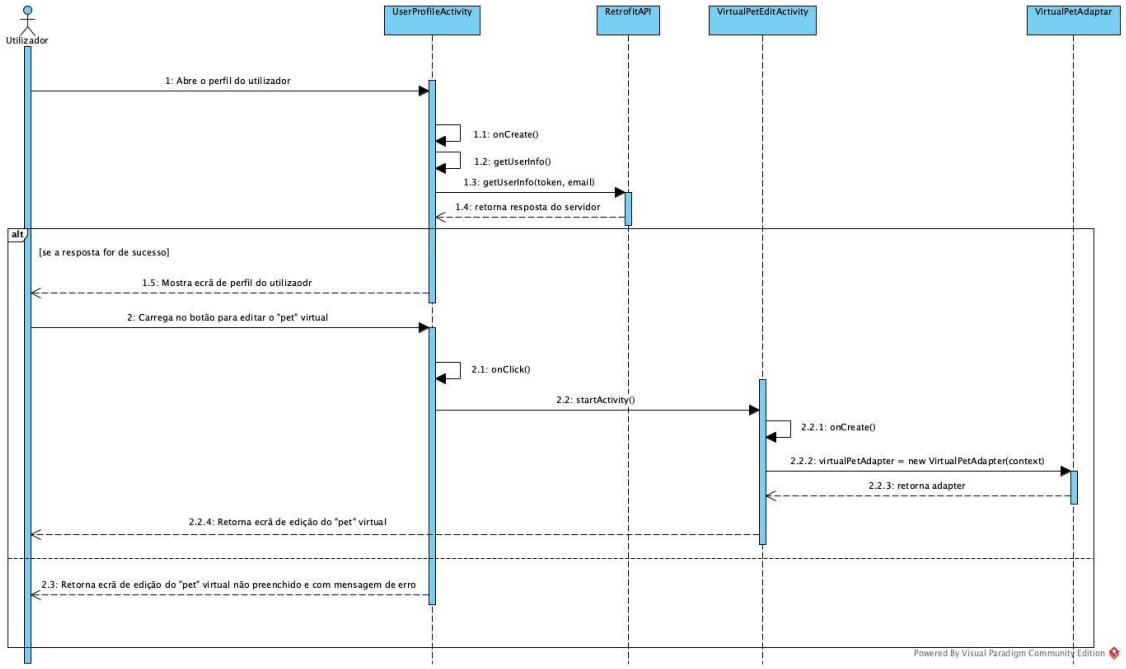


Figura 35 - Diagrama de Sequência para Mostrar Ecrã de Edição do "*Pet*" Virtual

Analisando a figura 33, quando o utilizador abre o perfil do mesmo, a atividade "*UserProfile*" é ativada e inicia o método "*onCreate()*" onde, para além de iniciar a "*view*" do perfil, também mapeia os campos da mesma para poder alterá-los e inicia os eventos para os botões presentes. No mesmo fluxo, o método "*getUserInfo()"* é acionado onde é chamado o método *"getUserInfo()"* da "*RetrofitAPI*" para construir um objeto da classe "*UserInfo*" com a informação recebida do micro serviço de gestão de utilizadores para preencher a "*view*".

Caso a chamada não tenha sucesso, a "*view*" é apresentada ao utilizador sem estar preenchida e com uma mensagem de erro a informar o erro no processo de busca da informação do utilizador. No caso de sucesso da resposta, o ecrã de edição de perfil é apresentado ao utilizador corretamente preenchido.

```xml
<LinearLayout
    android:layout_width="100dp"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/userPic"
        android:layout_width="94dp"
        android:layout_height="102dp"
        android:adjustViewBounds="true"
        android:src="@drawable/ic_user" />

    <ImageView
        android:id="@+id/virtualPetIcon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginTop="10dp"
        android:layout_marginLeft="20dp"
        android:src="@drawable/ic_virtual_pet" />
</LinearLayout>
```

Figura 36 - XML da "*View*" do Botão de Edição do "*Pet*" Virtual

O XML desta "*view*" foi alterado devido à funcionalidade a desenvolver. Uma "*ImageView*" com o "*id*" do "*virtualPetIcon*" e com a imagem do ícone do "*pet*" virtual foi criada verticalmente paralela à imagem de perfil do utilizador.

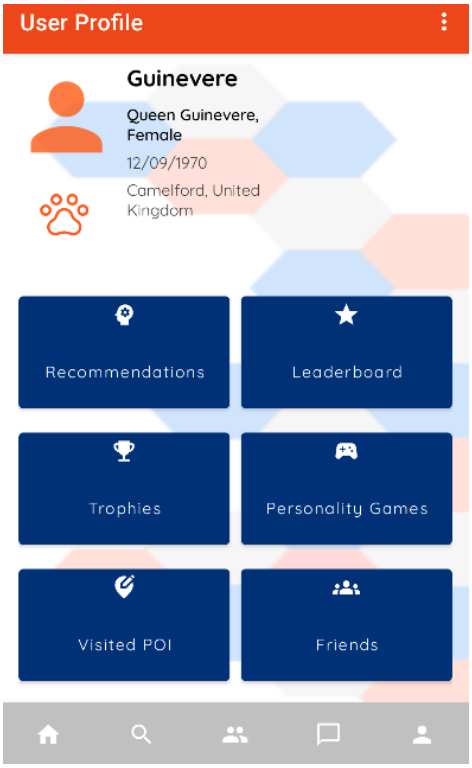


Figura 37 - Ecrã do Perfil do Utilizador com Ícone do "*Pet*" Virtual

```
virtualPetIcon = findViewById(R.id.virtualPetIcon);

virtualPetIcon.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Trigger virtual pet edit view
        startActivity(new Intent( packageContext: UserProfileActivity.this, VirtualPetEditActivity.class));
    }
});
```

Figura 38 - Código de Associação do Botão de Edição do "*Pet*" Virtual ao Evento de "*Click*"

Este ícone é mapeado no evento "*onCreate()*" da atividade "*UserProfile*" e associada a um evento do tipo click para despoletar a nova atividade para edição do "*pet*" virtual. Este evento ao ser despoletado pelo utilizador, inicia a atividade "*VirtualPetEdit*" e consequentemente o evento "*onCreate()*" do mesmo.

```
public class VirtualPetEditActivity extends AppCompatActivity {

    TabLayout tabLayout;
    ViewPager2 viewpager2;
    VirtualPetAdapter virtualPetAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_virtual_pet_edit);

        tabLayout = findViewById(R.id.tab_layout);
        viewpager2 = findViewById(R.id.view_pager);
        virtualPetAdapter = new VirtualPetAdapter( fragmentActivity: this);
        viewpager2.setAdapter(virtualPetAdapter);

        tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                viewpager2.setCurrentItem(tab.getPosition());
            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {}

            @Override
            public void onTabReselected(TabLayout.Tab tab) {}
        });

        viewpager2.registerOnPageChangeCallback(onPageSelected(position) -> {
                super.onPageSelected(position);
                tabLayout.getTabAt(position).select();
            });
    }
}
```

Figura 39 - Código da Atividade "*VirtualPetEdit*"

Neste método existem vários processos importantes. O primeiro prende-se com a inicialização da "*view*" do layout do ecrã de edição do "*pet*" virtual.

```
<RelativeLayout
    android:id="@+id/petLayout"
    android:layout_width="356dp"
    android:layout_height="515dp"
    android:background="@drawable/virtual_pet_layout"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" >

    <com.google.android.material.tabs.TabLayout
        android:layout_width="306dp"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="25dp"
        android:layout_height="wrap_content"
        android:id="@+id/tab_layout">

        <com.google.android.material.tabs.TabItem
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Edit Pet"/>

        <com.google.android.material.tabs.TabItem
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Pet History"/>

    </com.google.android.material.tabs.TabLayout>

    <androidx.viewpager2.widget.ViewPager2
        android:layout_width="306dp"
        android:layout_marginLeft="25dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/tab_layout"
        android:id="@+id/view_pager" />

</RelativeLayout>
```

Figura 40 - XML do “*Layout*” do Ecrã de Edição do "*Pet*" Virtual

O próximo é o mapeamento dos campos da “*view*” às variáveis da atividade e aqui deve-se salientar o “*tabLayout*” e o “*viewpager*”. O “*tabLayout*” relaciona-se com o elemento com mesmo nome do documento XML da figura acima, que possui dois “*TabItem*” (um para a página de edição do “*pet*” virtual e outra para o histórico do mesmo). A este elemento é associado um evento que capta a mudança de “*tab*” por parte do utilizador para iniciar o novo fragmento consoante a página a que este quer aceder. Para inicializar este fragmento, entra em ação a variável “*viewpager*”. Esta variável possui um “*adapter*” do tipo “*VirtualPetAdapter*” que, conforme o “*tab*” selecionado pelo utilizador, inicializa o fragmento do “*VirtualPetHistory*” ou então o “*VirtualPetEdit*”.

```java
public class VirtualPetAdapter extends FragmentStateAdapter {
    public VirtualPetAdapter(@NonNull FragmentActivity fragmentActivity) {
        super(fragmentActivity);
    }

    @NonNull
    @Override
    public Fragment createFragment(int position) {
        switch (position) {
            case 0:
                return new VirtualPetEdit();
            case 1:
                return new VirtualPetHistory();
            default:
                return new VirtualPetEdit();
        }
    }

    @Override
    public int getItemCount() { return 2; }
}
```

Figura 41 - Código do "*VirtualPetAdapter*"

No final da inicialização do fragmento, a "*view*" resultante é exposta no elemento XML denominado "*Viewpager2*".

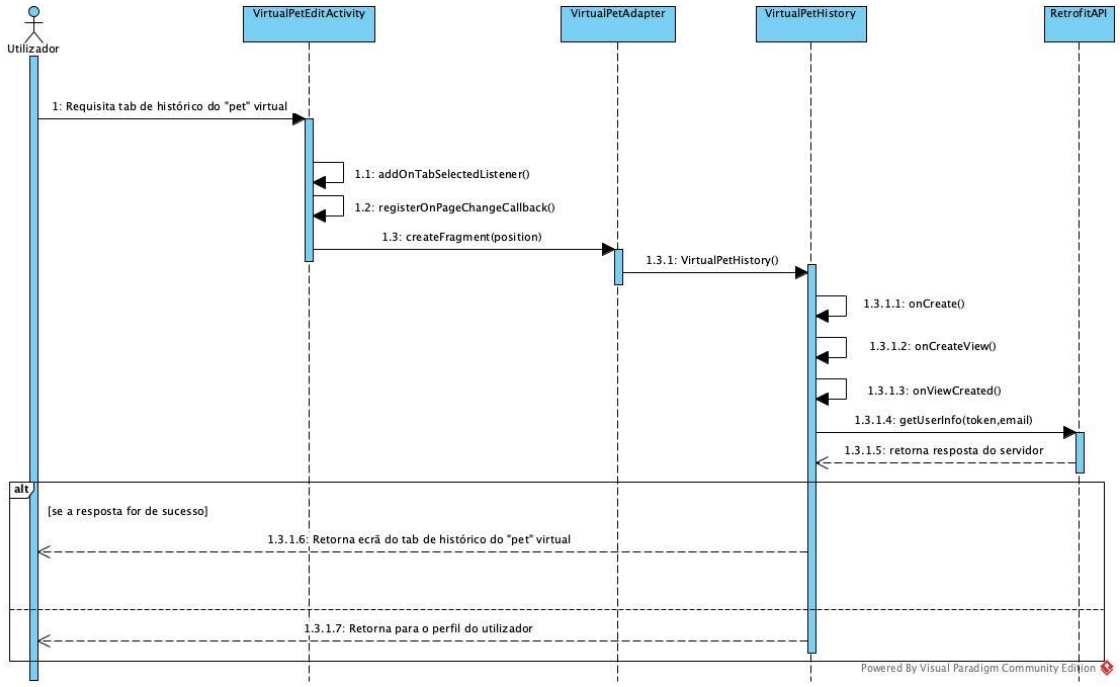


Figura 42 - Diagrama de Sequência da Seleção do "*Tab*" de Histórico do "*Pet*" Virtual

Relativamente ao "*tab*" do histórico do "*pet*" virtual ilustrado na figura acima, o processo inicial segue a ideologia explicada no que toca ao "*VirtualPetAdapter*". Após a inicialização do fragmento "*VirtualPetHistory*", várias funções são despoletadas. A função *"onCreate()"*, inicializa a classe "*Fragment*" do elemento. A função *"onCreateView()"*, inicializa a "*view*" deste fragmento.

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_virtual_pet_history, container,  attachToRoot: false);
}
```

Figura 43 - Código de Inicialização da "*View*" do Fragmento

No que respeita à "*view*", o documento XML possui elementos fulcrais, nomeadamente o "*ScrollView*", responsável pelo comportamento de "*scroll*" da tabela apresentada quando a informação não consegue caber no ecrã, e o "*TableLayout*", que resulta numa tabela que recebe a informação proveniente do histórico do "*pet*" virtual e apresenta ao utilizador três cabeçalhos distintos: data, mensagem e resposta.

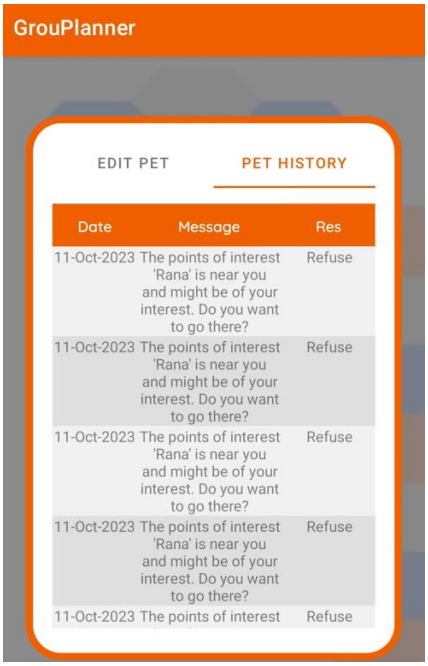


Figura 44 - Cabeçalhos da Tabela de Histórico de Interações do "*Pet*" Virtual

```xml
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="none"
    android:layout_weight="1"
    android:layout_marginBottom="30dp"
    android:layout_marginTop="15dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TableLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/white"
            android:id="@+id/table_VirtualPetHistory">
```

Figura 45 – "*ScrollView*" do XML do Histórico do "*Pet*" Virtual

```xml
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/white"
    android:id="@+id/table_VirtualPetHistory">

    <TableRow
        android:background="@color/dark_orange">

        <TextView
            android:layout_width="10dp"
            android:layout_height="match_parent"
            android:padding="10dp"
            android:fontFamily="@font/quicksand_medium"
            android:text="Date"
            android:textColor="@color/white"
            android:textSize="14dp"
            android:layout_weight="4"
            android:gravity="center"/>

        <TextView
            android:layout_width="50dp"
            android:layout_height="match_parent"
            android:padding="10dp"
            android:fontFamily="@font/quicksand_medium"
            android:text="Message"
            android:textColor="@color/white"
            android:textSize="14dp"
            android:layout_weight="4"
            android:gravity="center"/>

        <TextView
            android:layout_width="90dp"
            android:layout_height="match_parent"
            android:padding="10dp"
            android:fontFamily="@font/quicksand_medium"
            android:text="Res"
            android:textColor="@color/white"
            android:textSize="14dp"
            android:layout_weight="4"
            android:gravity="center"/>
```

Figura 46 – "*TableLayout*" do XML do Histórico do "*Pet*" Virtual

A função "*onViewCreated()*" é responsável pelo mapeamento dos campos da "*view*" criada pela função anterior e por preencher a "*view*" com a informação correta no que toca ao histórico do "*pet*" virtual.

```java
api.getUserInfo( auth: "Bearer " + userSession.getToken(), userSession.getUserEmail()).enqueue(new Callback<UserInfo>() {
    @Override
    public void onResponse(Call<UserInfo> call, Response<UserInfo> res) {
        if (res.code() == 200) {
            UserInfo dto = res.body();
            try {
                if (dto.virtualPetHistory != null) {
                    JSONObject virtualPetHistory = new JSONObject(dto.virtualPetHistory);
                    JSONArray interactions = virtualPetHistory.getJSONArray( name: "interactions");

                    for (int i = 0; i < interactions.length(); i++) {
                        JSONObject interaction = interactions.getJSONObject(i);
                        TableRow row = new TableRow(getActivity());
                        TextView date = new TextView(getActivity());
                        date.setText(interaction.getString( name: "date"));
                        date.setGravity(1);
                        TextView message = new TextView(getActivity());
                        message.setText(interaction.getString( name: "message"));
                        message.setMaxWidth(360);
                        message.setGravity(1);
                        TextView response = new TextView(getActivity());
                        response.setText(interaction.getString( name: "response"));
                        response.setGravity(1);
                        row.addView(date);
                        row.addView(message);
                        row.addView(response);

                        if ((i % 2) == 0) {
                            row.setBackgroundColor(0xEEEEEEEE);
                        } else {
                            row.setBackgroundColor(0xD3D3D3D3);
                        }

                        table_VirtualPetHistory.addView(row);
                    }
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
```

Figura 47 - Código para Preencher o Histórico do "*Pet*" Virtual

Após o sucesso da chamada da função *"getUserInfo()*", a resposta é desconstruída para retirar o JSON do histórico do "*pet*" virtual e porventura o Array de interações entre "*pet*" virtual e utilizador. Estas interações são iteradas e para cada uma é criada uma linha na tabela do XML ilustrado acima. Estas entidades têm um texto para cada um dos cabeçalhos e possuem uma cor de fundo alternada para distingui-las. No caso de a chamada não ter sucesso, o utilizador é redirecionado para o perfil do mesmo.

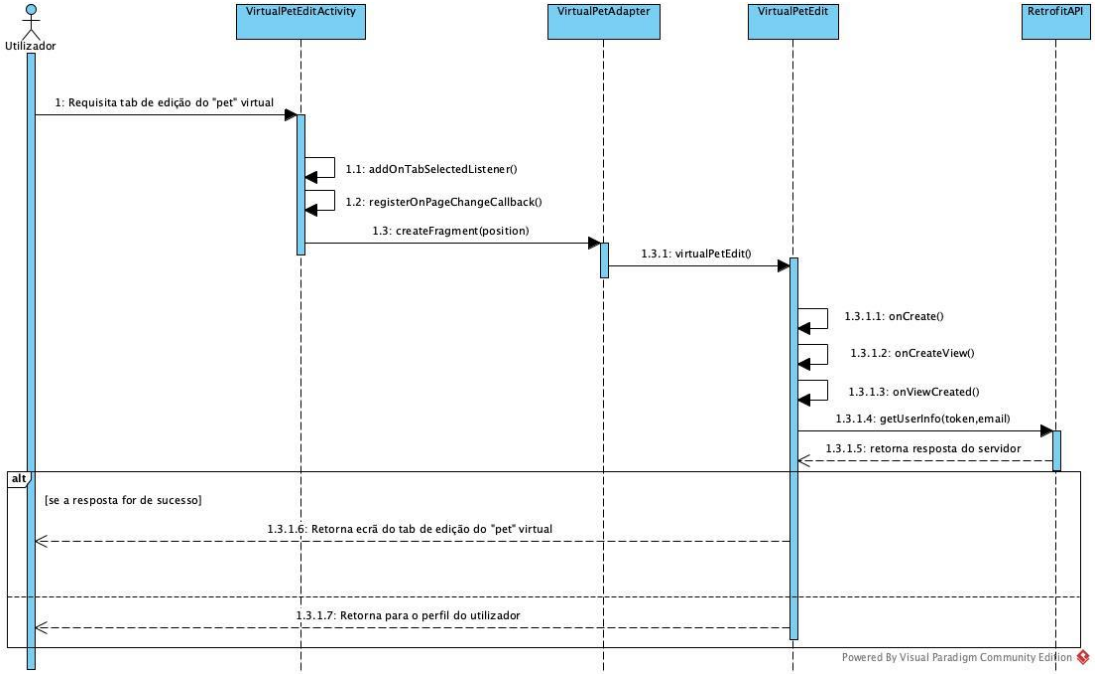


Figura 48 - Diagrama de Sequência da Seleção do "*Tab*" de Edição do "*Pet*" Virtual

Relativamente ao "*tab*" de edição do "*pet*" virtual ilustrado na figura 46, o processo é relativamente parecido ao anterior, com a exceção do XML da view, o método "*onCreateView()"* e *"onViewCreated()*".

O XML da "*view*" é semelhante à "*view*" da funcionalidade do registo do utilizador das figuras 26 e 27. Para o caso do nome do "*pet*" virtual, foi criado um título do tipo "*TextView*" a acompanhar o componente "*EditText*", que resulta num campo editável do tipo String para receber o nome do "*pet*" virtual. No caso do tipo de "*pet*" virtual, foi criado um "*LinearLayout*" para receber um "*RadioGroup*" e um "*FrameLayout*". O "*RadioGroup*" é constituído por dois tipos de "*RadioButton*": um para o modelo do macaco e outro para o modelo do urso. O "*FrameLayout*" é constituído por uma "*ImageView*" com o objetivo de ilustrar o "*pet*" virtual escolhido pelo utilizador.

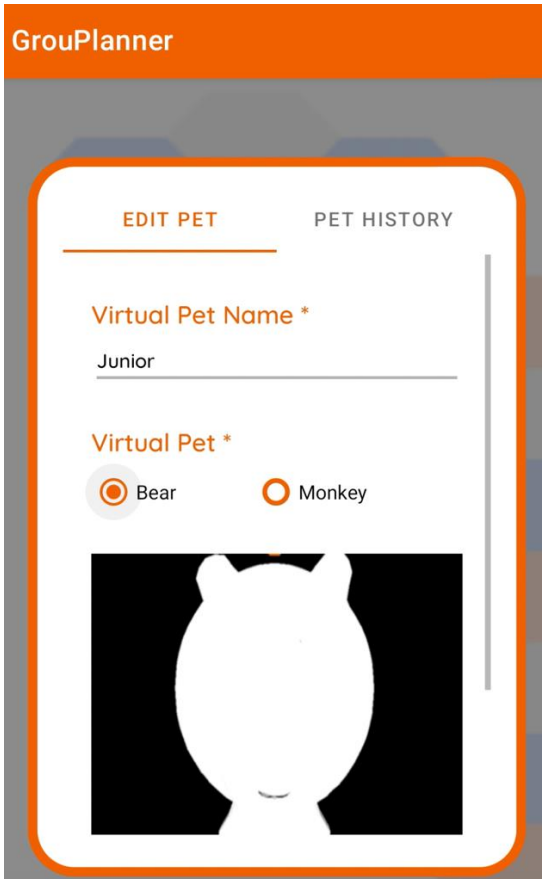


Figura 49 - Ecrã de Edição do "*Pet*" Virtual

```
<LinearLayout
    android:id="@+id/register_Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:gravity="center"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btn_EditVirtualPet"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:background="@color/dark_orange"
        android:backgroundTint="@color/dark_orange"
        android:fontFamily="@font/quicksand_bold"
        android:text="Save"
        android:textAllCaps="false"
        android:textColor="@color/white" />
</LinearLayout>
```

Figura 50 - XML do Botão para Guardar Alterações ao "*Pet*" Virtual

O método *"onCreateView()"*, para além de inicializar a "*view*" do respetivo fragmento, também inicializa os eventos do botão criado para esta "*view*", para guardar as alterações efetuadas pelo utilizador.

O método *"onViewCreated()"* fica encarregue de mapear os elementos da "*view*" no fragmento e também de preencher os campos da mesma com a informação guardada no perfil do utilizador relativamente ao "*pet*" virtual.

```
radioGroup_VirtualPet = getView().findViewById(R.id.radioGroup_VirtualPet);
dataInput_VirtualPetName = getView().findViewById(R.id.dataInput_VirtualPetName);
radioButton_Bear = getView().findViewById(R.id.radioButton_Bear);
radioButton_Monkey = getView().findViewById(R.id.radioButton_Monkey);

try {
    RetrofitAPI api = ApiUtils.getAPIService();
    SessionManager userSession = new SessionManager(getActivity());

    api.getUserInfo( auth: "Bearer " + userSession.getToken(), userSession.getUserEmail()).enqueue(new Callback<UserInfo>() {
        @Override
        public void onResponse(Call<UserInfo> call, Response<UserInfo> response) {
            if (response.code() == 200) {
                UserInfo dto = response.body();
                virtualPetHistory = dto.virtualPetHistory;
                dataInput_VirtualPetName.setText(dto.virtualPetName);
                if (dto.virtualPet != null && (dto.virtualPet.compareToIgnoreCase( str: "Bear") == 0)) {
                    radioButton_Bear.setChecked(true);
                } else {
                    radioButton_Monkey.setChecked(true);
                }
            }
        }

        @Override
        public void onFailure(Call<UserInfo> call, Throwable t) {
            getActivity().onBackPressed();
        }
    });
} catch (Exception e) {
    e.printStackTrace();
}
```

Figura 51 - Código para Receber Informação Guardada do "*Pet*" Virtual

O método "*getUserInfo()*" é acionado onde uma chamada HTTP REST é efetuada pelo método "*getUserInfo()*" da "*RetrofitAPI*" para construir um objeto da classe "*UserInfo*" com a informação recebida do micro serviço de gestão de utilizadores para preencher a "*view*". Deste objeto, os atributos nome e tipo do "*pet*" virtual são retirados para preencher o campo do tipo String com o nome do "*pet*" e acionar o respetivo "*radioButton*" referente ao tipo do "*pet*" guardado previamente. No caso de a resposta da chamada REST API falhar, o utilizador é redirecionado para o ecrã do perfil do utilizador.

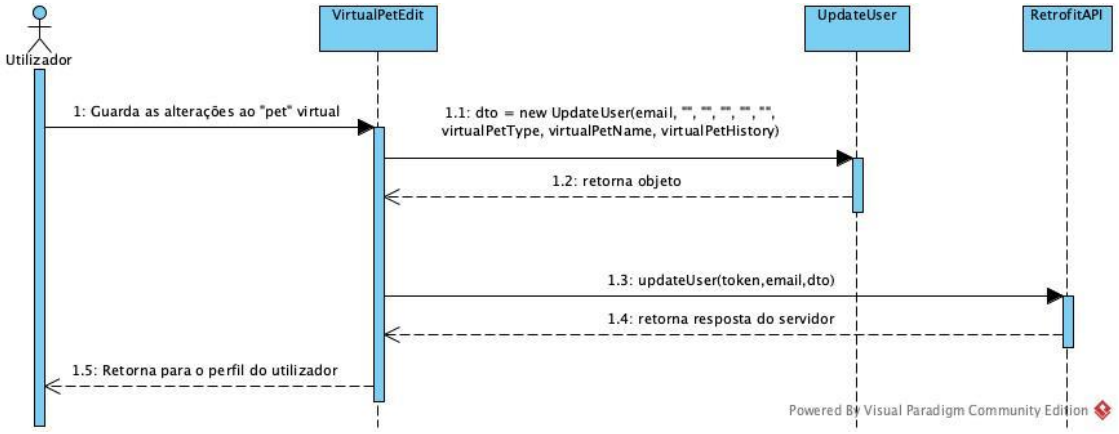


Figura 52 - Diagrama de Sequência da Acão de Guardar as Alterações ao "*Pet*" Virtual

Finalmente temos o processo de guardar as alterações efetuadas pelo utilizador ao "*pet*" virtual. Primeiramente, neste processo, o evento de "*Click*" do botão é acionado onde o nome do "*pet*" é obtido do campo "*VirtualPetName*" através do método "*getText()*" e o tipo do "*pet*" virtual é obtido do campo "*RadioGroup*", onde é identificado o "*id*" do "*RadioButton*" selecionado e depois comparado com os dois botões previamente mapeados, para definir qual deles foi selecionado pelo utilizador para retornar a String identificadora do tipo de "*pet*".

Estes dois campos são utilizados para preencher o objeto da classe "*UpdateUser*" utilizado na função "*updateUser()*" da classe "*RetrofitAPI*". Esta função despoleta uma chamada HTTP REST que atualiza a informação referente ao "*pet*" virtual no micro serviço de gestão de utilizadores. Tanto no caso de a chamada falhar, como no caso de sucesso da mesma, o utilizador é redirecionado para o seu perfil.

```
view.findViewById(R.id.btn_EditVirtualPet).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String virtualPetName = dataInput_VirtualPetName.getText().toString();
        String virtualPetType = "";
        int virtualPetId = radioGroup_VirtualPet.getCheckedRadioButtonId();

        if (radioGroup_VirtualPet.findViewById(virtualPetId).equals(radioButton_Bear)) {
            virtualPetType = "Bear";
        } else if (radioGroup_VirtualPet.findViewById(virtualPetId).equals(radioButton_Monkey)) {
            virtualPetType = "Monkey";
        }

        try {
            RetrofitAPI api = ApiUtils.getAPIService();
            SessionManager userSession = new SessionManager(getActivity());

            UpdateUser dto = new UpdateUser(userSession.getUserEmail(),  civil_state: "",  sexuality: "",  religion: "",  profession: "",
                    hasChildren: "", virtualPetType, virtualPetName, virtualPetHistory);

            api = ApiUtils.getAPIService();
            api.updateUser( token: "Bearer " + userSession.getToken(), dto).enqueue(new Callback<ResponseBody>() {
                @Override
                public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
                    if(response.code() == 200) {
                        getActivity().onBackPressed();
                    }
                }

                @Override
                public void onFailure(Call<ResponseBody> call, Throwable t) {
                    getActivity().onBackPressed();
                }
            });
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
```

Figura 53 - Código de Atualização da Informação referente ao "*Pet*" Virtual

4.6.3 Enviar Notificações para Alterar Recomendações de Acordo com o Contexto da Atividade

Esta funcionalidade visa sugerir novas recomendações ao utilizador para substituir outras na sua lista de atividades recomendadas que não sejam possíveis visitar devido a entraves no contexto das mesmas (por exemplo, más condições atmosféricas). Este requisito divide-se em dois domínios visto que a aplicação consegue sugerir recomendações de grupo e individuais, por isso também tem de conseguir alterar as recomendações dos dois tipos.

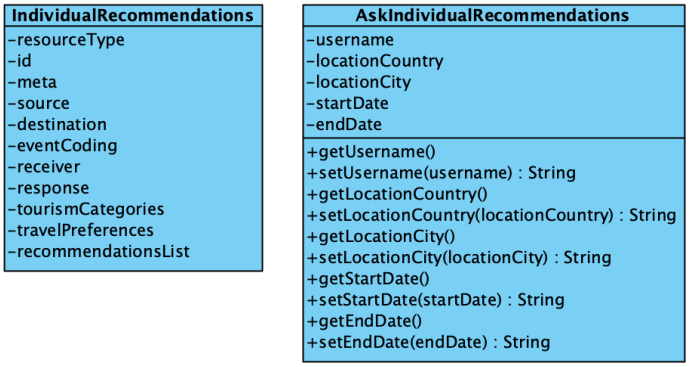


Figura 54 – Classes Utilizadas na Funcionalidade de Sugestão de Recomendações Individuais

Começando pelas recomendações individuais e analisando a figura 52, duas classes são reutilizadas neste processo da mesma forma que são utilizadas no processo de pedido de novas recomendações:

- **IndividualRecommendations** – esta classe é responsável por agregar toda a informação relativa às recomendações individuais do utilizador. É utilizada na chamada REST API ao respetivo micro serviço para associar a lista de recomendações ao utilizador em questão.
- **AskIndividualRecommendations** – esta classe é responsável por requisitar e agrupar toda a informação necessária para a chamada REST API de pedido de novas recomendações individuais para o utilizador.

Relativamente à funcionalidade, esta possui vários processos que a compõem, os quais se analisam detalhadamente.

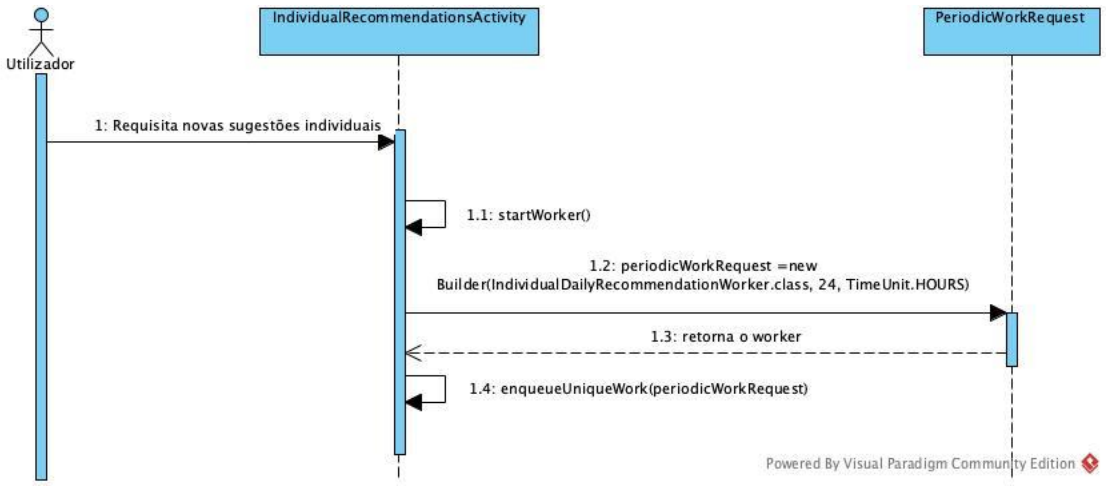


Figura 55 - Diagrama de Sequência de Inicialização do "*Worker*"

Após o utilizador requisitar novas recomendações individuais, no final do processo de as obter e associar ao utilizador, a função *"startWorker()"* da atividade "*IndividualRecommendations*" entra em ação.

```
private void startWorker() {

    // Transformar a start date em long
    long startDateLong = convertDateStringToMillis(manager.getStartDate());
    long endDateLong = convertDateStringToMillis(manager.getEndDate());

    // Obtem a data atual em milissegundos
    long currentDateInMillis = System.currentTimeMillis();

    // Calcula a diferença entre a data de início e a data atual em dias
    long diffInDays = TimeUnit.MILLISECONDS.toDays( duration: startDateLong - currentDateInMillis);

    PeriodicWorkRequest periodicWorkRequest = null;

    if (diffInDays <= 5) {
        // Se a diferença for de 5 dias ou menos, enfileire o trabalho para começar imediatamente
        periodicWorkRequest = new PeriodicWorkRequest.Builder(IndividualDailyRecommendationWorker.class,  repeatInterval: 24, TimeUnit.HOURS).build();
    } else {
        // Se a diferença for mais de 5 dias, enfileira o trabalho para começar quando a diferença for de 5 dias
        long delay = diffInDays - 5;
        periodicWorkRequest = new PeriodicWorkRequest.Builder(IndividualDailyRecommendationWorker.class,  repeatInterval: 24, TimeUnit.HOURS)
                .setInitialDelay(delay, TimeUnit.DAYS).build();
    }

    // Envia o WorkRequest para o WorkManager
    WorkManager.getInstance(this).enqueueUniquePeriodicWork( uniqueWorkName: "tag_" + manager.getUsername(),
            ExistingPeriodicWorkPolicy.REPLACE, periodicWorkRequest);

    // Salva o UUID do trabalho e a data de fim no SharedPreferences
    SharedPreferences sharedPreferences = getSharedPreferences( name: "com.example.grouplannerapp", Context.MODE_PRIVATE);
    sharedPreferences.edit().putString( s: "job_uuid_" + manager.getUsername(), periodicWorkRequest.getId().toString()).apply();
}
```

Figura 56 - Código da função "*startWorker()*"

Esta função começa por definir quantos dias faltam para o início da excursão definida pelo utilizador quando pediu novas recomendações individuais. Se a diferença for superior a cinco dias, um "*Worker*" periódico de vinte e quatro horas do tipo "*IndividualDailyRecommendation*" é construído com um atraso em dias igual à diferença calculada anteriormente. Isto significa que o "*Worker*" só irá ser inicializado após os dias calculados na diferença e será ativado em períodos recorrentes de vinte e quatro horas. Se a diferença for menor ou igual a cinco, um "*Worker*" periódico de vinte e quatro horas do tipo "*IndividualDailyRecommendation*" é construído sem atraso. Nesta mesma função, o "*id*" do "*Worker*" inicializado é guardado também para futuras ações.

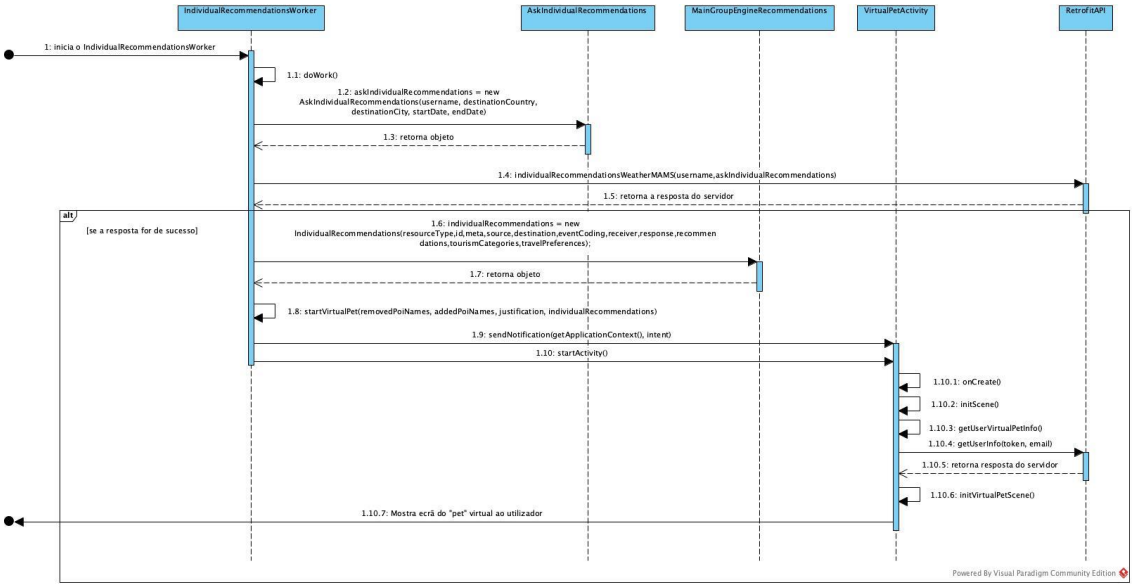


Figura 57 - Diagrama de Sequência de Ativação do "*Worker*"

Quando chega a altura de ativar o "*Worker*" em fila, a função "*doWork()*" é inicializada. Primeiramente, nesta função, a data atual do sistema em milissegundos é comparada com a data final da excursão. Caso a data do sistema seja posterior à data de fim da excursão, o "id" do "*Worker*" previamente guardado é utilizado na função "*cancelWorkById()*" para cancelar o "*Worker*" para não ser mais ativado.

```
//Verifica se a data atual passou da data de fim da excursão
long currentDateInMillis = System.currentTimeMillis();

SharedPreferences sharedPreferences = getApplicationContext().getSharedPreferences( s: "com.example.grouplannerapp", Context.MODE_PRIVATE);

long endDateLong = convertDateStringToMillis(manager.getEndDate());

if (currentDateInMillis >= endDateLong) {
    UUID workId = UUID.fromString(sharedPreferences.getString( s: "job_uuid_" + manager.getUsername(),  s1: ""));
    WorkManager.getInstance(getApplicationContext()).cancelWorkById(workId);
    return Result.success();
}
```

Figura 58 - Código para Cancelar "*Worker*"

Caso a data seja anterior ou igual à data da excursão, a função continua a sua cadeia de eventos. Primeiramente, uma chamada do tipo REST API é realizada ao micro serviço do motor de recomendações através da função "*individualRecommendationsWeatherMAMS()*". Esta função recebe como parâmetros o "*username*" e um objeto da classe "*AskIndividualRecommendations*". Este objeto é construído antes da chamada REST API e recebe como parâmetros para o seu construtor o "*username*", o país de destino, a cidade de destino, a data de início e a data de fim da excursão. Apenas se esta chamada for de sucesso é que o "*Worker*" continua o seu processamento.

```
String username = manager.getUsername();
api = ApiUtils.getMAMSService();

AskIndividualRecommendations askIndividualRecommendations = new AskIndividualRecommendations(username, manager.getDestinationCountry(),
        manager.getDestinationCity(), manager.getStartDate(), manager.getEndDate());
Call<WeatherIndividualRecommendations> call = api.individualRecommendationsWeatherMAMS(username,askIndividualRecommendations);
Response<WeatherIndividualRecommendations> response = call.execute();
```

Figura 59 - Código da Chamada REST API ao Motor de Recomendações

O resultado desta chamada é processado de onde são extraídas duas listas. A primeira é a lista de recomendações já associada ao utilizador pelo motor de recomendações, a outra é uma lista de alternativas as recomendações definidas na lista referida previamente. Cada POI da lista de alternativas possui um atributo que guarda o "*id*" do POI da lista de recomendações a ser substituído. A seguir, ambas as listas são iteradas e cada "*id*" de cada ponto de interesse da lista de recomendações é comparado com o referido anteriormente da lista de alternativas. No caso de coincidirem, quer dizer que esse ponto de interesse da lista de recomendações deve ser substituído pelo ponto da lista de alternativas. Para isso, um novo objeto do tipo "*RecommendationList*" é construído com as informações do ponto de interesse alternativo sendo inserido na lista de recomendações do utilizador na posição do ponto de interesse a ser removido. Neste mesmo processo, o nome do ponto de interesse removido e adicionado são guardados respetivamente no Array "*removedPoiNames*" e "*addedPoiNames*".

```
List<RecommendationList> recommendations = response.body().recommendationResult.getRecommendations();
List<RecommendationAlternative> alternatives = response.body().recommendationResult.getAlternatives();

List<String> addedPoiNames = new ArrayList<>();
List<String> removedPoiNames = new ArrayList<>();

for (int i = 0; i < recommendations.size(); i++) {
    for (RecommendationAlternative alternative : alternatives) {
        if (recommendations.get(i).getPoiId().equals(alternative.getPoiId())) {
            // Replace the item in recommendations with the alternative
            RecommendationList newRecommendation = new RecommendationList(
                    alternative.getAlternativePoiId(),
                    recommendations.get(i).getOrder(),
                    recommendations.get(i).getPredictedCompatibility(),
                    alternative.getAlternativeJustification()
            );

            removedPoiNames.add(alternative.getPoiName());
            addedPoiNames.add(alternative.getAlternativePoiName());

            recommendations.set(i, newRecommendation);
            break;  // No need to check the rest of the alternatives
        }
    }
}
```

Figura 60 - Código de Substituição de Recomendações Individuais

Após este processamento, um objeto do tipo "*IndividualRecommendations*" é construído com a informação proveniente da chamada "*individualRecommendationsWeatherMAMS()"* com a exceção da lista de recomendações substituída pela lista desenvolvida no bloco de código anterior. Finalmente, este objeto, os dois Arrays mencionados acima com os nomes dos pontos de interesse e a justificação para a mudança de pontos de interesse, são enviados como parâmetros para a função *"startVirtualPet()".* A justificação referida é uma String utilizada para explicar ao utilizador que estas mudanças são sugeridas devido às más condições atmosféricas.

```
// Create the IndividualRecommendations object with the updated recommendations list
IndividualRecommendations individualRecommendations = new IndividualRecommendations(
        response.body().resourceType,
        response.body().id,
        response.body().meta,
        response.body().source,
        response.body().destination,
        response.body().eventCoding,
        response.body().receiver,
        response.body().response,
        recommendations,  // the updated recommendations list
        response.body().tourismCategories,
        response.body().travelPreferences
);

startVirtualPet(removedPoiNames, addedPoiNames, justification: "because of the weather conditions.", individualRecommendations);
```

Figura 61 - Código de Inicialização da Função "*startVirtualPet()*"

Nesta função, uma String é construída a explicar quais os pontos de interesse a serem substituídos e por quais, através da iteração dos Arrays referidos anteriormente, e com a adição da justificação a explicar o porquê de serem substituídos. A seguir, a função "*startActivity()*" é chamada para inicializar a atividade de renderização do "*pet*" virtual ("*VirtualPetActivity*") em que recebe como parâmetro a String previamente construída, o objeto do tipo "*IndividualRecommendations*" referido anteriormente e dois Booleans que servem para identificar qual a funcionalidade que iniciou a atividade (neste caso estão os dois a falso porque se trata das recomendações individuais). Por último, a função "*sendNotification()*" da atividade "*VirtualPetActivity*" é também executada.

```
private void startVirtualPet(List<String> removedPoiNames,List<String> addedPoiNames, String justification, IndividualRecommendations responseBody) {
    Intent intent = new Intent(getApplicationContext(), VirtualPetActivity.class).setFlags(FLAG_ACTIVITY_NEW_TASK);
    VirtualPetActivity.sendNotification(getApplicationContext(), intent);

    String message = "The points of interest '";

    for (String poiNameRemoved : removedPoiNames) {
        message += poiNameRemoved + ",";
    }

    message = message.substring(0, message.length() -1);
    message += "' will be substituted by the points of interest '";

    for (String poiNameAdded : addedPoiNames) {
        message += poiNameAdded + ",";
    }

    message = message.substring(0, message.length() -1);
    message += "' " + justification;

    Bundle b = new Bundle();
    b.putString("message", message);
    b.putString("isGroupRecommendation", "false");
    b.putString("isNearPlaceRecommendation", "false");
    b.putString("individualRecommendations", new GsonBuilder().create().toJson(responseBody));
    intent.putExtras(b);
    startActivity(getApplicationContext(), intent, b);
}
```

Figura 62 - Código da função "*startVirtualPet()*"

Na função "*sendNotification()*", é construído um objeto do tipo "*NotificationCompat*" que contem toda a informação necessária para a construção de uma notificação de aplicação de Android. Este objeto, após construção, é inicializado e enviado para o "*NotificationManager*",

para ser lançado pela aplicação no dispositivo móvel, para alertar o utilizador de uma renderização do "*pet*" virtual.

```
public static void sendNotification(Context context, Intent intent) {
    String title = "Hey! I'm Virtual Pet!";
    String message = "I have a message for you!";
    int reqCode = 1;
    PendingIntent pendingIntent = PendingIntent.getActivity(context, reqCode, intent, PendingIntent.FLAG_ONE_SHOT);
    String CHANNEL_ID = "virtual_pet";
    NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(context, CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_send_message)
            .setContentTitle(title)
            .setContentText(message)
            .setAutoCancel(true)
            .setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION))
            .setContentIntent(pendingIntent);
    NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = "Virtual Pet";
        int importance = NotificationManager.IMPORTANCE_HIGH;
        NotificationChannel mChannel = new NotificationChannel(CHANNEL_ID, name, importance);
        notificationManager.createNotificationChannel(mChannel);
    }
    notificationManager.notify(reqCode, notificationBuilder.build());
}
```

Figura 63 - Código da Função "*sendNotification()*"

Após a inicialização da atividade "*VirtualPetActivity*", a função *"onCreate()*" é acionada para construir a "*view*" de renderização, mapear todos os parâmetros enviados para as respetivas variáveis e inicializar todos os eventos referentes aos botões de aceitação e recusa da recomendação.

```
protected void onCreate(Bundle savedInstanceState) {
    setContentView(R.layout.activity_virtual_pet);

    Bundle b = getIntent().getExtras();
    message = b.getString( key: "message");
    isGroupRecommendation = b.getString( key: "isGroupRecommendation");
    isNearPlaceRecommendation = b.getString( key: "isNearPlaceRecommendation");

    if (isNearPlaceRecommendation.equals("true")) {
        nearPlace = new GsonBuilder().create().fromJson(b.getString( key: "nearPlace"), VirtualPetNearPlacesService.VirtualPetPlacesNear.class);
    } else if (isGroupRecommendation.equals("true")) {
        groupRecommendations = new GsonBuilder().create().fromJson(b.getString( key: "groupRecommendations"), GroupRecommendations.class);
        token = b.getString( key: "token");
    } else {
        individualRecommendations = new GsonBuilder().create().fromJson(b.getString( key: "individualRecommendations"), IndividualRecommendations.class);
    }
    super.onCreate(savedInstanceState);

    btn_accept = findViewById(R.id.acceptButton);
    btn_refuse = findViewById(R.id.refuseButton);
```

Figura 64 - Código da Função "*onCreate()*" do "*VirtualPetActivity*"

A "*view*" construída pela função anterior é traduzida para um ficheiro XML que possui um Layout principal que consiste na janela do "*pet*" virtual. Esta janela possui como elementos uma "*GLSurfaceView*" que é o componente responsável por receber e mostrar a renderização do "*pet*" virtual, uma "*ScrollView*" com um "*TextView*" incorporado que recebe e apresenta o texto referente à ação a tomar pelo utilizador, e dois "*Button*", sendo que um serve para aceitar a recomendação do "*pet*" virtual e o outro para negar.

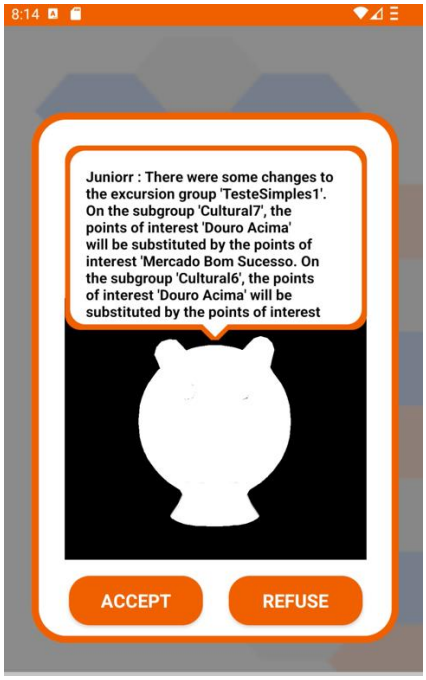


Figura 65 - Exemplo de Ecrã de Notificação do "*Pet*" Virtual

```
<Button
    android:id="@+id/acceptButton"
    android:layout_width="130dp"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="190dp"
    android:layout_marginBottom="17dp"
    android:fontFamily="@font/quicksand_bold"
    android:text="Accept"
    android:textSize="18sp"
    android:textColor="@color/white"
    android:textStyle="bold"
    android:background="@drawable/virtual_pet_button" />

<Button
    android:id="@+id/refuseButton"
    android:layout_width="130dp"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="35dp"
    android:layout_marginBottom="17dp"
    android:fontFamily="@font/quicksand_bold"
    android:text="Refuse"
    android:textSize="18sp"
    android:textColor="@color/white"
    android:textStyle="bold"
    android:background="@drawable/virtual_pet_button" />
```

Figura 66 - Código da "*View*" dos Botões do "*Pet*" Virtual

```xml
<RelativeLayout
    android:id="@+id/petLayout"
    android:layout_width="356dp"
    android:layout_height="515dp"
    android:background="@drawable/virtual_pet_layout"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    >

    <android.opengl.GLSurfaceView
        android:id="@+id/glPetSurface"
        android:layout_width="293dp"
        android:layout_height="254dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="32dp"
        android:layout_marginTop="180dp"/>

    <ScrollView
        android:id="@+id/petTextScoller"
        android:layout_width="293dp"
        android:layout_height="189dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="32dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="31dp"
        android:padding="20dp"
        android:scrollbars="vertical"
        android:background="@drawable/virtual_pet_text"
        android:fillViewport="true">
        <TextView
            android:id="@+id/petTextView"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:fontFamily="@font/quicksand_bold"
            android:textSize="14sp"
            android:textStyle="bold"
            android:textColor="@color/black" />
    </ScrollView>
```

Figura 67 - Código da "*View*" de Renderização e Apresentação da Mensagem do "*Pet*" Virtual

Após a função "*onCreate()*" terminar e a "*view*" ter sido inicializada com sucesso, a função "*initScene()*" é despoletada, acionando a função "*getUserVirtualPetInfo()*". Esta função encarrega-se de fazer uma chamada ao micro serviço de gestão de utilizadores para recolher as informações armazenadas no perfil do utilizador referente ao nome e tipo do "*pet*" virtual. No caso de sucesso desta chamada, a função "*initVirtualPetScene()*" é inicializada.

```java
public void initScene() { getUserVirtualPetInfo(); }

public void getUserVirtualPetInfo() {
    virtualPetFile = "Bear";
    virtualPetName = "";
    virtualPetHistory = "";

    try {
        RetrofitAPI api = ApiUtils.getAPIService();
        userSession = new SessionManager( context: VirtualPetActivity.this);

        api.getUserInfo( auth: "Bearer " + userSession.getToken(), userSession.getUserEmail()).enqueue(new Callback<UserInfo>() {
            @Override
            public void onResponse(Call<UserInfo> call, Response<UserInfo> response) {
                if (response.code() == 200) {
                    UserInfo dto = response.body();
                    if (dto.virtualPet != null && dto.virtualPetName != null) {
                        virtualPetFile = dto.virtualPet;
                        virtualPetName = dto.virtualPetName;
                        virtualPetHistory = dto.virtualPetHistory;
                        initVirtualPetScene();
                    }
                }
            }

            @Override
            public void onFailure(Call<UserInfo> call, Throwable t) {}
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 68 - Código da Função "*getUserVirtualPetInfo()*"

A função “*initVirtualPetScene()*” tem a responsabilidade de renderizar o “*pet*” virtual e preencher a “*view*” apresentada ao utilizador com a renderização e a mensagem de recomendação. A mensagem de recomendação é associada a “*view*” através do método “*setText()*” do “*TextView*” referido anteriormente. No caso de o “*pet*” virtual possuir um nome definido pelo utilizador, este também é associado com a mensagem. No que toca à renderização, este processo começa com a adição de um objeto do tipo “*Light*”, da biblioteca min3d, ao objeto “*scene*”, que representa o elemento “*GLSurfaceView*” da “*view*” apresentada ao utilizador. Este objeto tem a responsabilidade de iluminar a cena do “*pet*” virtual. A seguir, é atribuída uma cor de fundo à cena (neste caso a cor preta) e o “*Parser*” da biblioteca min3d é construído e inicializado pela função “*parse()*” para ler o ficheiro MD2 do “*pet*” virtual selecionado pelo utilizador e converter num objeto animado que possa ser lido e renderizado pela “*GLSurfaceView*”. Este objeto, dependendo do “*pet*” virtual escolhido, é manipulado, nomeadamente a sua posição, escala e “*frames*” por segundo, e adicionado à cena para ser mostrado ao utilizador.

```
public void initVirtualPetScene() {
    TextView titleTextView = findViewById(R.id.petTextView);

    if (virtualPetName.length() > 0) {
        titleTextView.setText(virtualPetName + " : " + message);
    } else {
        titleTextView.setText(message);
    }

    scene.lights().add(new Light());
    scene.backgroundColor().setAll( $r: 0, $g: 0, $b: 0, $a: 0);

    IParser parser = Parser.createParser(Parser.Type.MD2,
            getResources(), resourceID: "com.example.grouplannerapp:raw/" + virtualPetFile.toLowerCase(), generateMipMap: false);
    parser.parse();

    AnimationObject3d pet = parser.getParsedAnimationObject();

    if (virtualPetFile.equals("Bear")) {
        pet.scale().x = pet.scale().y = pet.scale().z = .25f;
        pet.rotation().y = 90;
        pet.position().y = -2.1f;
        pet.position().x = 0.15f;
    } else if (virtualPetFile.equals("Monkey")) {
        pet.scale().x = pet.scale().y = pet.scale().z = .045f;
        pet.rotation().z = -90;
        pet.rotation().x = -90;
    }

    scene.addChild(pet);

    if (virtualPetFile.equals("Bear")) {
        pet.setFps(80);
    } else if (virtualPetFile.equals("Monkey")) {
        pet.setFps(20);
    }

    pet.play();
}
```

Figura 69 - Código da Função “initVirtualPetScene()”

Após todo este processo, o utilizador terá a hipótese de aceitar ou recusar a recomendação proposta pelo “*pet*” virtual.

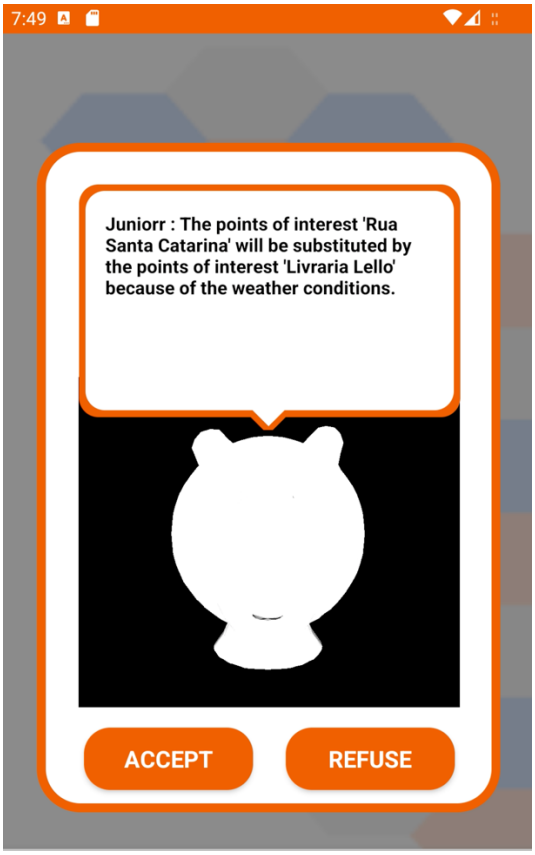


Figura 70 - Ecrã da Notificação de Sugestão de Recomendação Individual pelo "*Pet*" Virtual

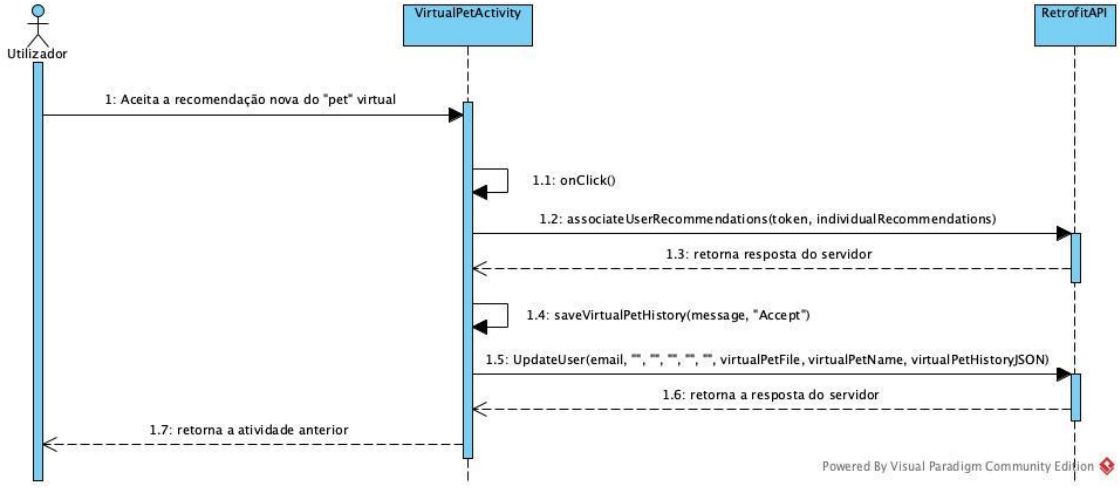


Figura 71 - Diagrama de Sequência de Aceitação da Recomendação Individual

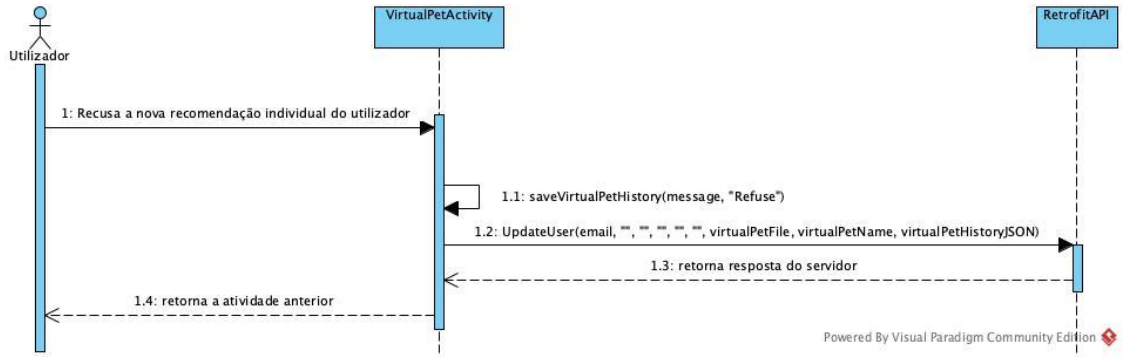


Figura 72 - Diagrama de Sequência de Negação da Recomendação Individual

No caso do utilizador aceitar a recomendação, o evento *"onClick()"* do botão de aceitação é acionado.

```
RetrofitAPI api = ApiUtils.getAPIService();
api.associateUserRecommendations( auth: "Bearer " + userSession.getToken(), individualRecommendations).enqueue(new Callback<ResponseBody>() {
    @Override
    public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
        saveVirtualPetHistory(message,  response: "Accept");
        finish();
    }

    @Override
    public void onFailure(Call<ResponseBody> call, Throwable t) {
    }
});
```

Figura 73 - Código de Aceitação da Recomendação Individual

Nesta função, uma chamada do tipo REST API é efetuada através da função "*associateUserRecommendations()*" para associar ao utilizador em questão, a nova lista de recomendações construída com o ponto de interesse alternativo. Após esta chamada, a função "*saveVirtualPetHistory()*" é ativada com a mensagem apresentada ao utilizador pelo "*pet*" virtual e a resposta de aceitação.

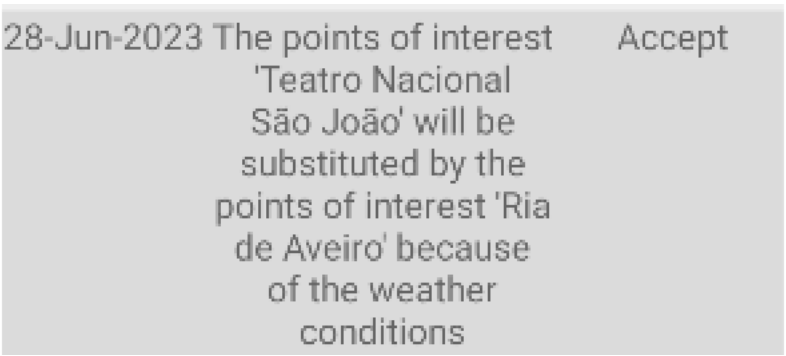


Figura 74 - Ecrã de Exemplo de Interação de Aceitação do Histórico do "*Pet*" Virtual

No caso do utilizador rejeitar a recomendação, o evento *"onClick()"* do botão de negação é acionado, acionando consequentemente a função "*saveVirtualPetHistory()*" com a mensagem apresentada ao utilizador pelo "*pet*" virtual e a resposta de negação.

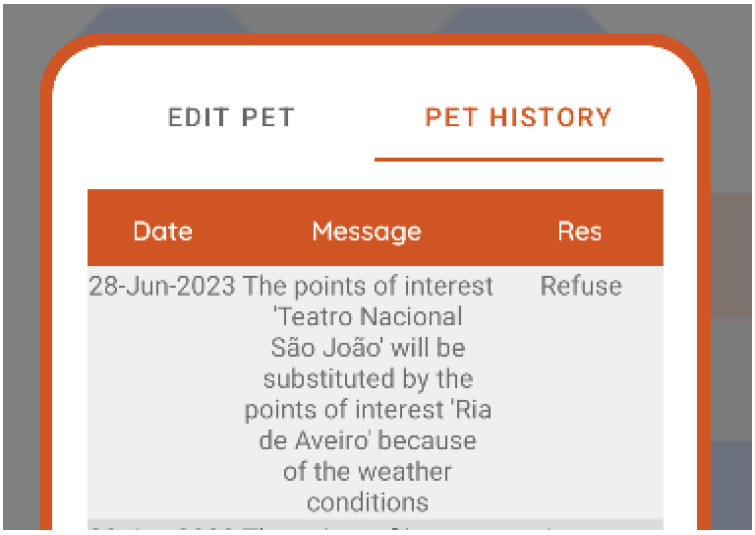
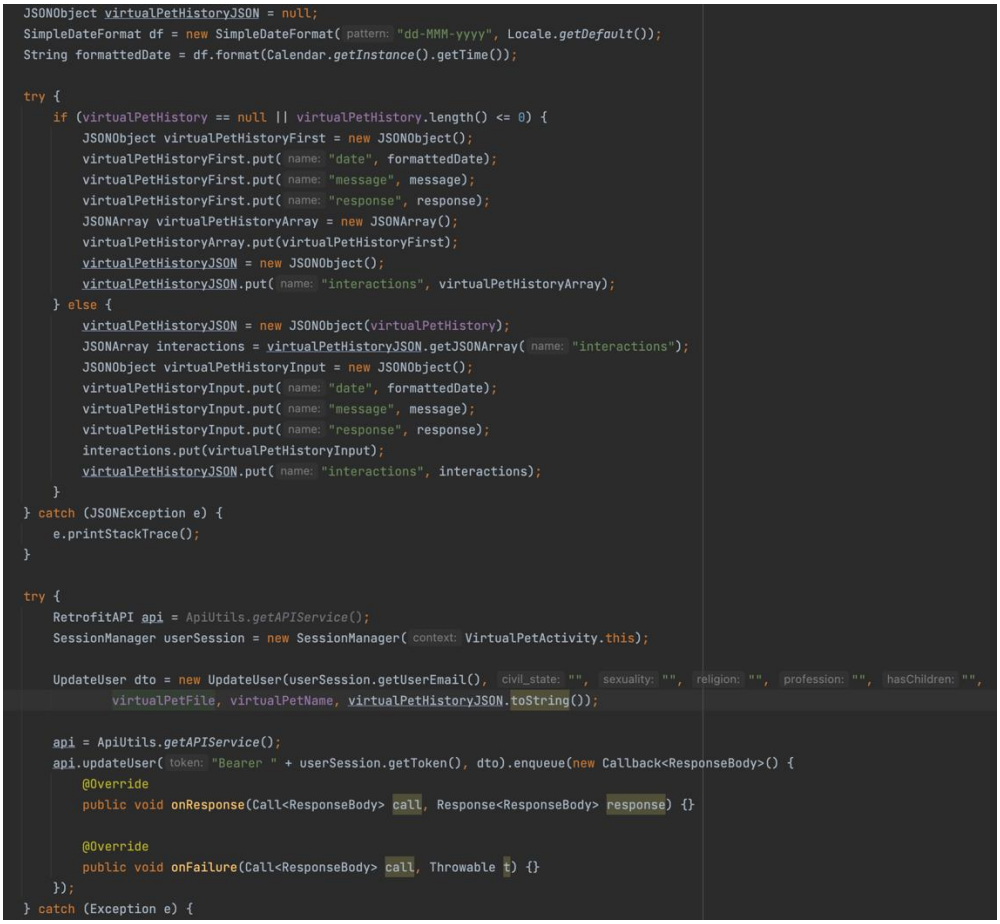


Figura 75 - Ecrã de Exemplo de Interação de Negação do Histórico do "*Pet*" Virtual

```
btn_refuse.setOnClickListener(new View.OnClickListener() {
    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    public void onClick(View v) {
        if (isNearPlaceRecommendation.equals("true")) {
            userSession.setNearPlacesCount(userSession.getNearPlacesCount() + 1);
        }
        saveVirtualPetHistory(message, response: "Refuse");
        finish();
    }
});
```

Figura 76 - Código de Negação da Recomendação Individual

A função "*saveVirtualPetHistory()*" começa por construir uma String com a data atual do sistema. Posteriormente, o objeto "*virtualPetHistory*" obtido na função "*getUserVirtualPetInfo()*" é analisado para entender se este se encontra vazio ou não. No caso de estar vazio, um novo JSON com a data, a mensagem e a resposta da interação do utilizador com o "*pet*" virtual é construído e adicionado a um novo JSONArray. No caso de estar preenchido, o JSON com a data, a mensagem e a resposta da interação do utilizador com o "*pet*" virtual é construído e adicionado ao JSONArray contido no objeto "*virtualPetHistory*". No final, uma chamada do tipo REST API é efetuada pela função "*updateUser()*" para guardar esta nova lista de interações do utilizador com o "*pet*" virtual no perfil do mesmo. Esta função requer a construção de um objeto do tipo "*UpdateUser*" que irá conter toda a informação necessária a atualização do perfil do utilizador.



```
JSONObject virtualPetHistoryJSON = null;
SimpleDateFormat df = new SimpleDateFormat( pattern: "dd-MMM-yyyy", Locale.getDefault());
String formattedDate = df.format(Calendar.getInstance().getTime());

try {
    if (virtualPetHistory == null || virtualPetHistory.length() <= 0) {
        JSONObject virtualPetHistoryFirst = new JSONObject();
        virtualPetHistoryFirst.put( name: "date", formattedDate);
        virtualPetHistoryFirst.put( name: "message", message);
        virtualPetHistoryFirst.put( name: "response", response);
        JSONArray virtualPetHistoryArray = new JSONArray();
        virtualPetHistoryArray.put(virtualPetHistoryFirst);
        virtualPetHistoryJSON = new JSONObject();
        virtualPetHistoryJSON.put( name: "interactions", virtualPetHistoryArray);
    } else {
        virtualPetHistoryJSON = new JSONObject(virtualPetHistory);
        JSONArray interactions = virtualPetHistoryJSON.getJSONArray( name: "interactions");
        JSONObject virtualPetHistoryInput = new JSONObject();
        virtualPetHistoryInput.put( name: "date", formattedDate);
        virtualPetHistoryInput.put( name: "message", message);
        virtualPetHistoryInput.put( name: "response", response);
        interactions.put(virtualPetHistoryInput);
        virtualPetHistoryJSON.put( name: "interactions", interactions);
    }
} catch (JSONException e) {
    e.printStackTrace();
}

try {
    RetrofitAPI api = ApiUtils.getAPIService();
    SessionManager userSession = new SessionManager( context: VirtualPetActivity.this);

    UpdateUser dto = new UpdateUser(userSession.getUserEmail(), civil_state: "", sexuality: "", religion: "", profession: "", hasChildren: "",
            virtualPetFile, virtualPetName, virtualPetHistoryJSON.toString());

    api = ApiUtils.getAPIService();
    api.updateUser( token: "Bearer " + userSession.getToken(), dto).enqueue(new Callback<ResponseBody>() {
        @Override
        public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {}

        @Override
        public void onFailure(Call<ResponseBody> call, Throwable t) {}
    });
} catch (Exception e) {
```

Figura 77 - Código da Função "*saveVirtualPetHistory()*"

Relativamente as recomendações de grupo, o processo é relativamente semelhante ao das recomendações individuais. Neste sentido, apenas é dado ênfase às alterações e processos que diferem, de modo a não repetir informação.

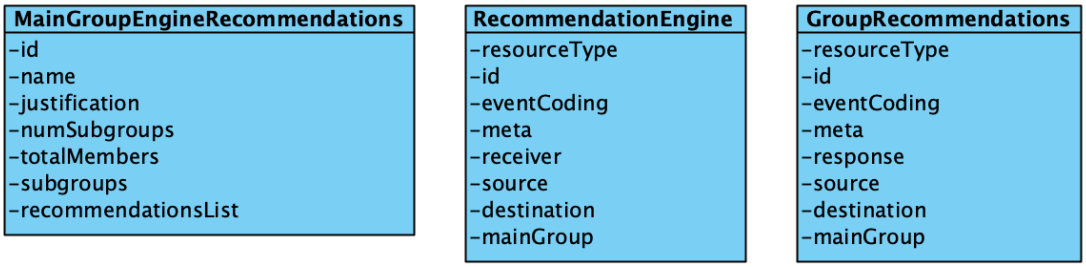


Figura 78 – Classes Utilizadas na Funcionalidade de Sugestão de Recomendações de Grupo

Relativamente as classes utilizadas, novas entidades são reutilizadas neste processo da mesma forma que são utilizadas no processo de pedido de novas recomendações de grupo:

- **MainGroupEngineRecommendations** – esta classe é responsável por agrupar toda a informação relativa a um grupo de excursão, inclusive todos os detalhes dos seus subgrupos previamente gerados.
- **RecommendationEngine** - esta classe é responsável por requisitar e agrupar toda a informação necessária para a chamada REST API de pedido de novas recomendações de um grupo.
- **GroupRecommendations** - esta classe é responsável por agregar toda a informação relativa a um grupo de excursão, para ser utilizada na chamada REST API ao respetivo micro serviço de forma a guardar corretamente toda a informação corretamente na base de dados.

Da mesma forma que as recomendações individuais, esta funcionalidade também possui vários processos. Começando com a requisição de novas recomendações de grupo que irá gerar o "*Worker*" para ser despoletado mais tarde, o processo é muito semelhante ao detalhado anteriormente.

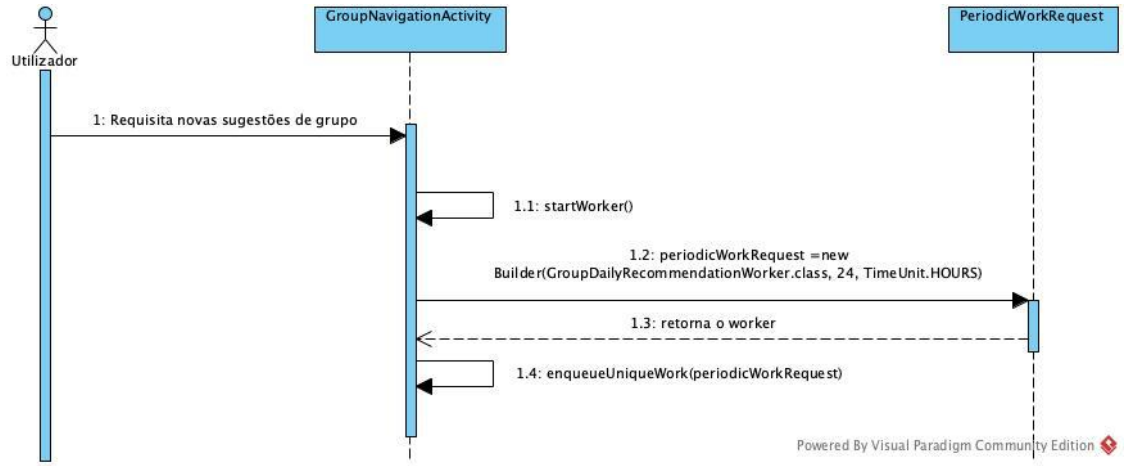


Figura 79 - Diagrama de Sequência de Inicialização do "*Worker*" de Recomendações de Grupo

A única diferença neste processo prende-se com a inicialização do "*Worker*" periódico. Neste caso, um "*Worker*" da classe "*GroupDailyRecommendationWorker*" é inicializado, cumprindo na mesma todos os requisitos e passos detalhados anteriormente para as recomendações individuais.

```
if (diffInDays <= 5) {
    // Se a diferença for de 5 dias ou menos, agenda o trabalho para começar imediatamente
    periodicWorkRequest = new PeriodicWorkRequest.Builder(GroupDailyRecommendationWorker.class,  repeatInterval: 24, TimeUnit.HOURS)
            .setInputData(inputData)
            .build();

    //oneTimeWorkRequest = new OneTimeWorkRequest.Builder(GroupDailyRecommendationWorker.class).setInitialDelay(2, TimeUnit.MINUTES).

} else {
    // Se a diferença for mais de 5 dias, enfileira o trabalho para começar quando a diferença for de 5 dias
    long delay = diffInDays - 5;
    periodicWorkRequest = new PeriodicWorkRequest.Builder(GroupDailyRecommendationWorker.class,  repeatInterval: 24, TimeUnit.HOURS)
            .setInputData(inputData)
            .setInitialDelay(delay, TimeUnit.DAYS)
            .build();
    //oneTimeWorkRequest = new OneTimeWorkRequest.Builder(GroupDailyRecommendationWorker.class).setInitialDelay(2, TimeUnit.MINUTES).
}
```

Figura 80 - Código da função "*startWorker()*" das Recomendações de Grupo

Quando o "*Worker*" em fila é despoletado, o processo que se segue é onde se encontram as maiores diferenças relativamente ao processo das recomendações individuais.

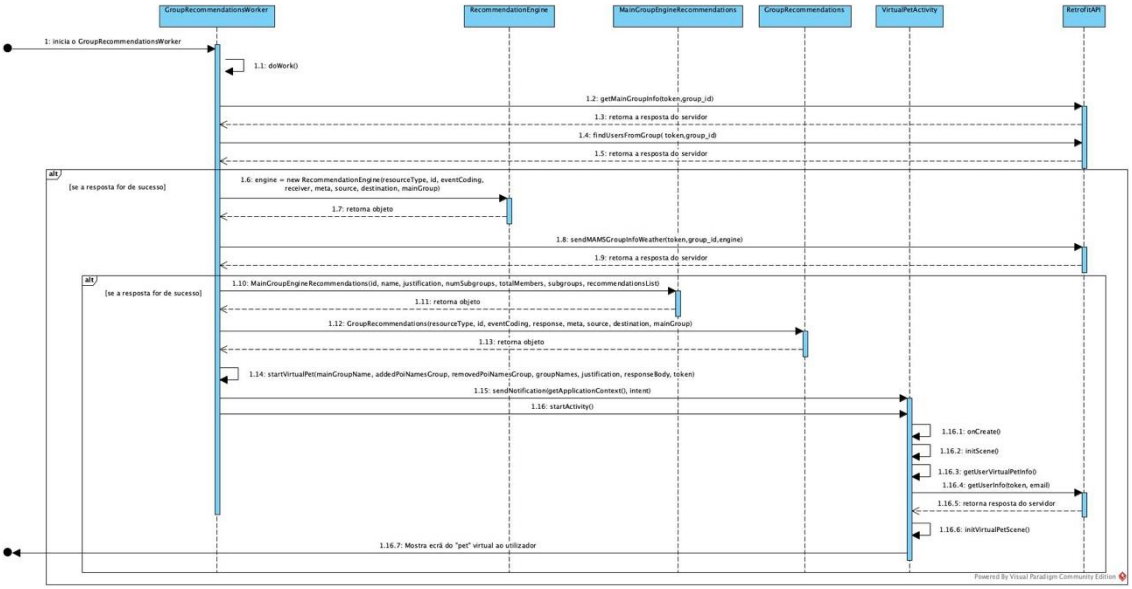


Figura 81 – Diagrama de Sequência de Ativação do "*Worker*" de Recomendações de Grupo

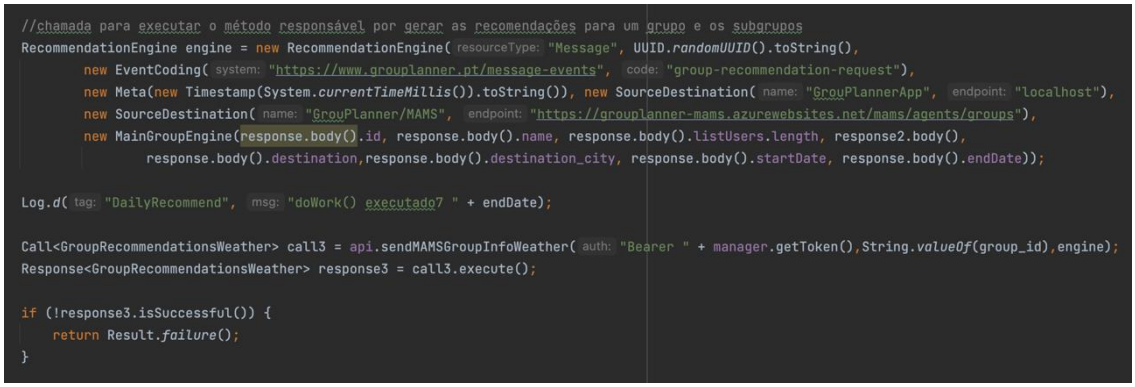
A função "*doWork()*" é inicializada e o processo inicial é o mesmo das recomendações individuais. A função irá analisar se o "*Worker*" iniciou após a data de fim da excursão de grupo e, caso seja verdade, irá cancelar o "*Worker*" de modo a não ser inicializado mais nenhuma vez.

Figura 82 - Código de Cancelamento do "*Worker*" de Recomendações de Grupo

No caso de o "*Worker*" não ser cancelado, a função irá fazer duas chamadas REST API. A chamada da função "*getMainGroupInfo()*" é responsável por encontrar e retornar a excursão de grupo pela qual o "*Worker*" foi inicializado. Para isso, utiliza o "*id*" do respetivo grupo guardado no processo de pedido de novas recomendações de grupo. A chamada da função "*findUsersFromGroup()"* utiliza o mesmo "*id*" para retornar todos os utilizadores que pertencem a este grupo de excursão.

Figura 83 - Código para Extrair Grupo de Excursão e os seus Utilizadores

Esta informação é utilizada na construção de um objeto da classe "*RecommendationEngine*", que será utilizado na função "*sendMAMSGroupInfoWeather()"* para fazer uma chamada ao micro serviço do motor de recomendações para obter novas recomendações de grupo.

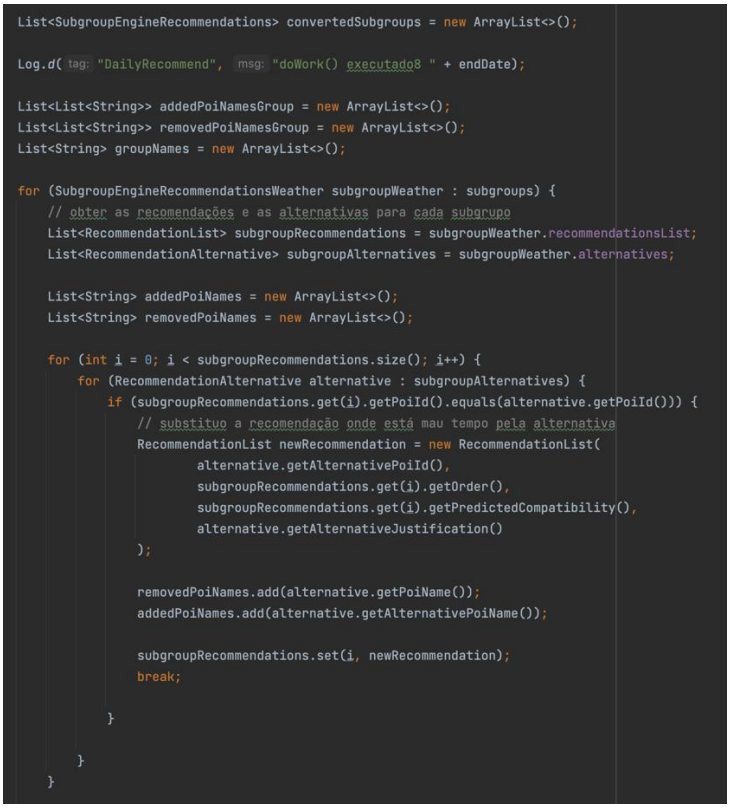


```
//chamada para executar o método responsável por gerar as recomendações para um grupo e os subgrupos
RecommendationEngine engine = new RecommendationEngine( resourceType: "Message", UUID.randomUUID().toString(),
        new EventCoding( system: "https://www.grouplanner.pt/message-events", code: "group-recommendation-request"),
        new Meta(new Timestamp(System.currentTimeMillis()).toString()), new SourceDestination( name: "GrouPlannerApp", endpoint: "localhost"),
        new SourceDestination( name: "GrouPlanner/MAMS", endpoint: "https://grouplanner-mams.azurewebsites.net/mams/agents/groups"),
        new MainGroupEngine(response.body().id, response.body().name, response.body().listUsers.length, response2.body(),
                response.body().destination,response.body().destination_city, response.body().startDate, response.body().endDate));

Log.d( tag: "DailyRecommend", msg: "doWork() executado7 " + endDate);

Call<GroupRecommendationsWeather> call3 = api.sendMAMSGroupInfoWeather( auth: "Bearer " + manager.getToken(),String.valueOf(group_id),engine);
Response<GroupRecommendationsWeather> response3 = call3.execute();

if (!response3.isSuccessful()) {
    return Result.failure();
}
```

Figura 84 - Código da Chamada REST API ao Motor de Recomendações de Grupo

Após o sucesso da chamada, todos os subgrupos presentes na resposta retornada são iterados e, para cada um deles, vão ser extraídas uma lista de recomendações que equivale às recomendações previamente atribuídas a este subgrupo ("*recommendationsList*") e uma lista de recomendações alternativas às já recomendadas ("*alternatives*").

Ambas as listas são iteradas e o processo referido nas recomendações individuais é replicado onde cada "*id*" de cada ponto de interesse da lista de recomendações e comparado com o "*id*" da lista de alternativas. No caso de coincidirem, quer dizer que esse ponto de interesse da lista de recomendações deve ser substituído pelo ponto com o "*id*" correspondente da lista de alternativas.



```
List<SubgroupEngineRecommendations> convertedSubgroups = new ArrayList<>();

Log.d( tag: "DailyRecommend", msg: "doWork() executado8 " + endDate);

List<List<String>> addedPoiNamesGroup = new ArrayList<>();
List<List<String>> removedPoiNamesGroup = new ArrayList<>();
List<String> groupNames = new ArrayList<>();

for (SubgroupEngineRecommendationsWeather subgroupWeather : subgroups) {
    // obter as recomendações e as alternativas para cada subgrupo
    List<RecommendationList> subgroupRecommendations = subgroupWeather.recommendationsList;
    List<RecommendationAlternative> subgroupAlternatives = subgroupWeather.alternatives;

    List<String> addedPoiNames = new ArrayList<>();
    List<String> removedPoiNames = new ArrayList<>();

    for (int i = 0; i < subgroupRecommendations.size(); i++) {
        for (RecommendationAlternative alternative : subgroupAlternatives) {
            if (subgroupRecommendations.get(i).getPoiId().equals(alternative.getPoiId())) {
                // substituo a recomendação onde está mau tempo pela alternativa
                RecommendationList newRecommendation = new RecommendationList(
                        alternative.getAlternativePoiId(),
                        subgroupRecommendations.get(i).getOrder(),
                        subgroupRecommendations.get(i).getPredictedCompatibility(),
                        alternative.getAlternativeJustification()
                );

                removedPoiNames.add(alternative.getPoiName());
                addedPoiNames.add(alternative.getAlternativePoiName());

                subgroupRecommendations.set(i, newRecommendation);
                break;

            }

        }
    }
```

Figura 85 - Código de Iteração das listas de Recomendações do Subgrupo

No final da iteração de cada subgrupo, um objeto do tipo "*SubgroupEngineRecommendations*" é construído com a nova lista de recomendações do subgrupo e o resto de toda a informação do mesmo para ser adicionado a um Array denominado "*convertedSubgroups*", que será utilizado na construção do objeto do grupo de excursão. De realçar também que, para além do nome do subgrupo ser guardado num Array denominado "*groupNames*", todos os nomes dos pontos de interesse removidos e adicionados são guardados respetivamente nas matrizes "*removedPoiNamesGroup*" e "*addedPoiNamesGroup*". A utilização de matrizes em vez de Arrays deve-se ao facto de poderem existir vários subgrupos a ser iterados.

```
// Conversão de SubgroupEngineRecommendationsWeather para SubgroupEngineRecommendations de cada subgrupo
SubgroupEngineRecommendations convertedSubgroup = new SubgroupEngineRecommendations(
        subgroupWeather.id,
        subgroupWeather.name,
        subgroupWeather.description,
        subgroupWeather.justification,
        subgroupWeather.numMembers,
        subgroupWeather.members,
        subgroupRecommendations  // A lista de recomendações atualizada
);

convertedSubgroups.add(convertedSubgroup);
addedPoiNamesGroup.add(addedPoiNames);
removedPoiNamesGroup.add(removedPoiNames);
groupNames.add(subgroupWeather.name);
```

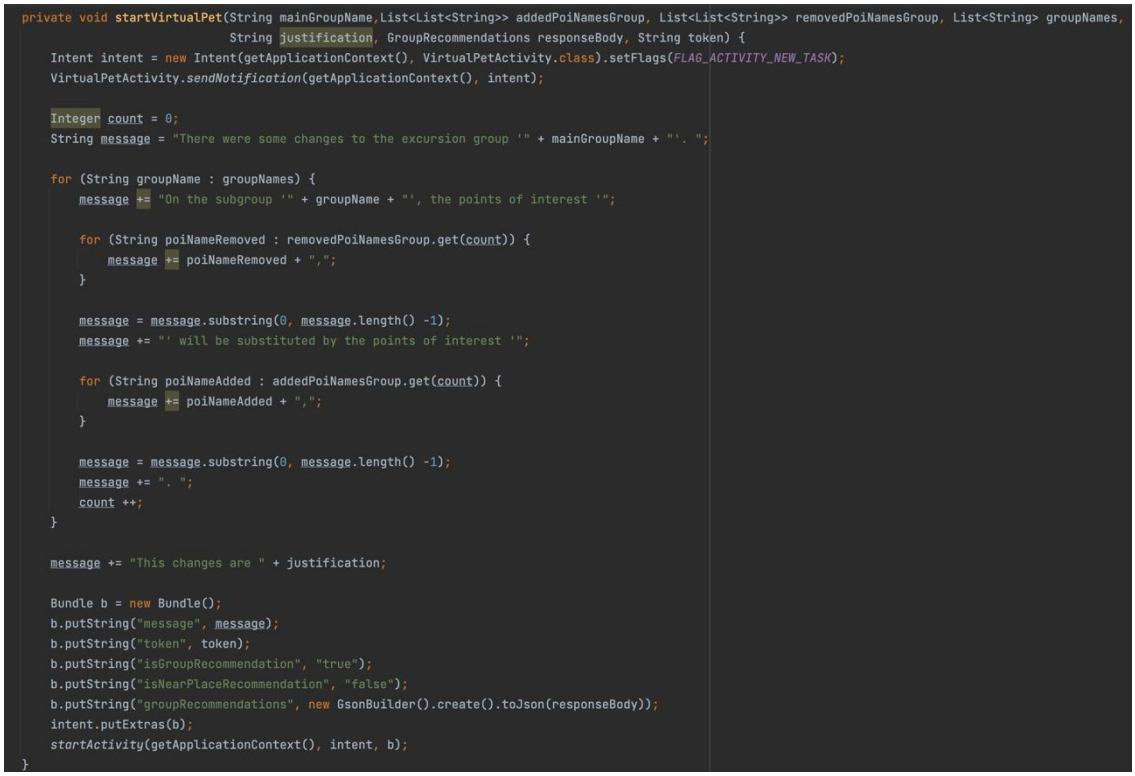
Figura 86 - Código do Final da Iteração de Cada Subgrupo

No final, estes novos subgrupos são agrupados num objeto da classe "*MainGroupEngineRecommendations*" que será posteriormente utilizado na construção do objeto do tipo "*GroupRecommendations*", que irá conter toda a informação do grupo de excursão assim como as alterações das recomendações a cada subgrupo previamente criado. Este objeto é depois enviado como parâmetro para a função "*startVirtualPet()*" com a justificação das mudanças efetuadas, o nome do grupo de excursão e o Array e Matrizes previamente mencionados.

```
MainGroupEngineRecommendations mainGroupEngineRecommendations = new MainGroupEngineRecommendations(
        response3.body().mainGroup.id,
        response3.body().mainGroup.name,
        response3.body().mainGroup.justification,
        response3.body().mainGroup.numSubgroups,
        response3.body().mainGroup.totalMembers,
        convertedSubgroups,
        response3.body().mainGroup.recommendationsList
);

GroupRecommendations groupRecommendations = new GroupRecommendations(response3.body().resourceType,response3.body().id,
        response3.body().eventCoding,response3.body().response, response3.body().meta,
        response3.body().source, response3.body().destination,mainGroupEngineRecommendations);

startVirtualPet(groupRecommendations.mainGroup.name, addedPoiNamesGroup, removedPoiNamesGroup,
        groupNames, justification: "because of the weather conditions.",groupRecommendations, manager.getToken());
```

Figura 87 - Código de Inicialização da Função "*startVirtualPet()*" das Recomendações de Grupo

O código da função "*startVirtualPet()*" é muito semelhante ao das recomendações individuais, com a exceção da construção da mensagem e dos parâmetros enviados para a atividade de renderização do "*pet*" virtual. No que toca a mensagem, esta tem de incorporar todas as mudanças feitas a cada subgrupo do grupo de excursão e, para tal, um ciclo extra teve de ser introduzido para iterar todos os subgrupos. No que toca aos parâmetros, um novo foi criado para enviar um JSON do objeto do tipo "*GroupRecommendations*" e o Boolean "*isGroupRecommendation*" foi sinalizado como verdadeiro para identificar a funcionalidade que inicializou a atividade de renderização do "*pet*" virtual.



```
private void startVirtualPet(String mainGroupName,List<List<String>> addedPoiNamesGroup, List<List<String>> removedPoiNamesGroup, List<String> groupNames,
                             String justification, GroupRecommendations responseBody, String token) {
    Intent intent = new Intent(getApplicationContext(), VirtualPetActivity.class).setFlags(FLAG_ACTIVITY_NEW_TASK);
    VirtualPetActivity.sendNotification(getApplicationContext(), intent);

    Integer count = 0;
    String message = "There were some changes to the excursion group '" + mainGroupName + "'. ";

    for (String groupName : groupNames) {
        message += "On the subgroup '" + groupName + "', the points of interest '";

        for (String poiNameRemoved : removedPoiNamesGroup.get(count)) {
            message += poiNameRemoved + ",";
        }

        message = message.substring(0, message.length() -1);
        message += "' will be substituted by the points of interest '";

        for (String poiNameAdded : addedPoiNamesGroup.get(count)) {
            message += poiNameAdded + ",";
        }

        message = message.substring(0, message.length() -1);
        message += ". ";
        count ++;
    }

    message += "This changes are " + justification;

    Bundle b = new Bundle();
    b.putString("message", message);
    b.putString("token", token);
    b.putString("isGroupRecommendation", "true");
    b.putString("isNearPlaceRecommendation", "false");
    b.putString("groupRecommendations", new GsonBuilder().create().toJson(responseBody));
    intent.putExtras(b);
    startActivity(getApplicationContext(), intent, b);
}
```

Figura 88 - Código da função "*startVirtualPet()*" das Recomendações de Grupo

Daqui para a frente, todo o processo de renderização do "*pet*" virtual é o mesmo referido anteriormente no processo de recomendações individuais.

Após este processo, o líder do grupo pode aceitar ou recusar a sugestão de grupo anunciada pelo "*pet*" virtual, sendo que ambas as ações são relativamente parecidas às das recomendações individuais. O líder do grupo é o único utilizador do grupo que recebe estas sugestões visto que, na aplicação, quem toma as decisões finais é o líder do grupo e só ele pode pedir novas recomendações para o grupo a que pertence.

No caso de o utilizador recusar a sugestão do "*pet*" virtual, o processo é o mesmo relativamente as recomendações individuais, ou seja, a resposta de negação do utilizador é guardada no histórico de interações do "*pet*" virtual.

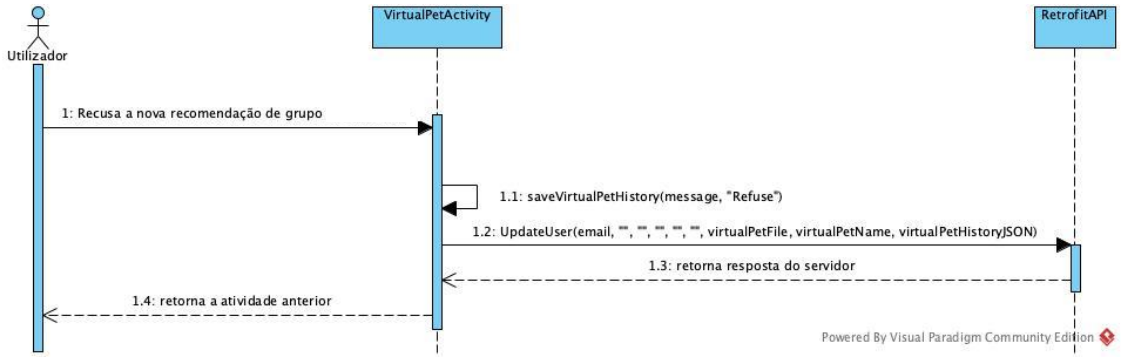


Figura 89 - Diagrama de Sequência de Negação da Recomendação de Grupo

No caso de o utilizador aceitar a sugestão do "*pet*" virtual, o processo é semelhante ao das recomendações individuais.

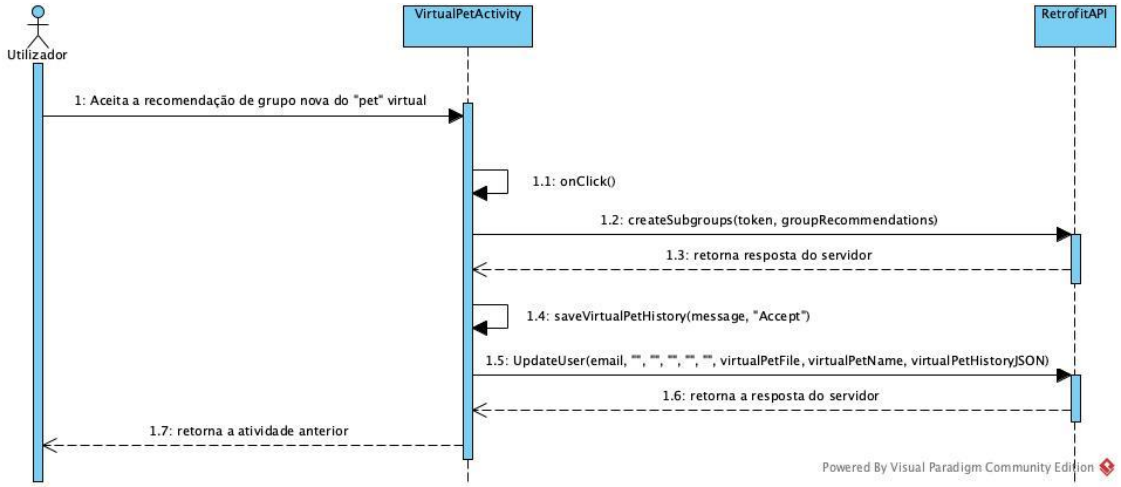


Figura 90 - Diagrama de Sequência de Aceitação da Recomendação de Grupo

Como se pode analisar na figura acima, a única alteração prende-se com a função chamada quando o evento de "*Click*" é acionado. Neste caso, a função "*createSubgroups()*" é despoletada para fazer uma chamada do tipo REST API para o micro serviço, para alterar os subgrupos do grupo principal para acolher as novas recomendações aceites pelo utilizador.

```
} else if (isGroupRecommendation.equals("true")) {
    RetrofitAPI api2 = ApiUtils.getSocialNetworkAPIService();
    api2.createSubgroups( auth: "Bearer " + token, groupRecommendations).enqueue(new Callback<Long>() {
        @Override
        public void onResponse(Call<Long> call, Response<Long> response) {
            saveVirtualPetHistory(message,  response: "Accept");
            finish();
        }

        @Override
        public void onFailure(Call<Long> call, Throwable t) {
        }
    });
```

Figura 91 - Código de Aceitação da Recomendação de Grupo

4.6.4 Enviar Notificações e Direcionar para Novas Recomendações Perto do Utilizador

Esta funcionalidade tem o objetivo de sugerir e direcionar o utilizador a atividades ou locais perto dele que se encaixem no seu perfil preferencial de atividades.

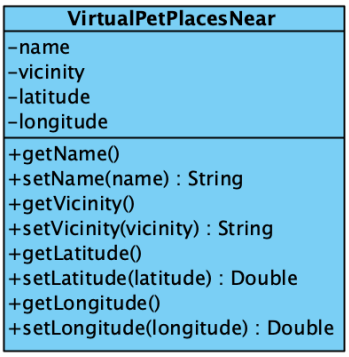


Figura 92 – Classes Utilizadas na Funcionalidade de Recomendação de POI Perto do Utilizador

Para o funcionamento deste requisito, uma nova classe foi criada denominada "*VirtualPetPlacesNear*". Esta classe tem o objetivo de agrupar a informação de um POI sugerido por esta funcionalidade, nomeadamente:

- **Name** – atributo que guarda o nome do POI sugerido.
- **Vicinity** - atributo que guarda o nome da área onde se encontra o POI sugerido (cidade, distrito, etc).
- **Latitude** – atributo que guarda as coordenadas de latitude do POI.
- **Longitude** – atributo que guarda as coordenadas de longitude do POI.

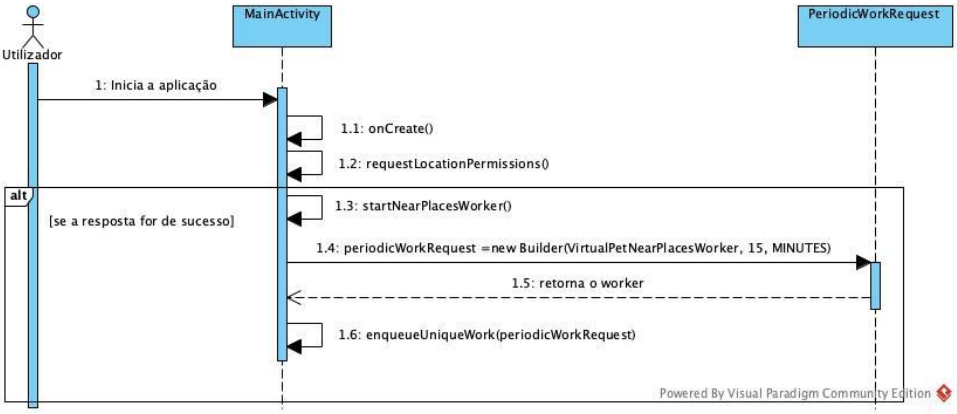


Figura 93 - Diagrama de Sequência da Inicialização do "*Worker*"

Ao iniciar a aplicação, os primeiros processos desta funcionalidade são acionados. Na função "*onCreate()*" da "*MainActivity*" descrita anteriormente no primeiro requisito, a função "*requestLocationPermissions()"* é iniciada. Nesta função, será avaliado se o utilizador já permitiu que a aplicação aceda a sua localização. Caso não tenha permitido, a função "*requestPermissions()"* é chamada, recebendo como parâmetro o tipo de permissão que queremos requisitar o acesso ao utilizador e o código único do pedido. Esta função mostra uma janela ao utilizador para requisitar o acesso e, consoante a interação do utilizador, a função "*onRequestPermissionsResult()*" é despoletada, que analisa se a resposta dada pelo utilizador foi permissiva. No caso de o utilizador aceitar o acesso ou no caso de já o ter feito numa outra altura, a função "*startNearPlacesWorker()*" é ativada para criar um "*Worker*" do tipo "*VirtualPetNearPlacesWorker*" que será inicializado a cada quinze minutos. Se este "*Worker*" já tiver sido inicializado numa outra iteração da aplicação então o processo de inicialização é parado.

```
public void requestLocationPermissions() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED ||
                ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED
        ) {
            requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION}, requestCode: 1000);
        } else {
            startNearPlacesWorker();
        }
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    if (requestCode == 1000) {
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            startNearPlacesWorker();
        }
    }
}

public void startNearPlacesWorker() {
    SessionManager manager = new SessionManager( context: this);
    WorkManager instance = WorkManager.getInstance();
    ListenableFuture<List<WorkInfo>> statuses = instance.getWorkInfosForUniqueWork( uniqueWorkName: "tag_near_places_" + manager.getUsername());
    try {
        boolean running = false;
        List<WorkInfo> workInfoList = statuses.get();
        for (WorkInfo workInfo : workInfoList) {
            WorkInfo.State state = workInfo.getState();
            running = state == WorkInfo.State.RUNNING | state == WorkInfo.State.ENQUEUED;
        }
        if (!running) {
            PeriodicWorkRequest periodicWorkRequest = new PeriodicWorkRequest.Builder(VirtualPetNearPlacesWorker.class, repeatInterval: 15, TimeUnit.MINUTES).build();
            WorkManager.getInstance(this).enqueueUniquePeriodicWork( uniqueWorkName: "tag_near_places_" + manager.getUsername(),
                    ExistingPeriodicWorkPolicy.REPLACE, periodicWorkRequest);
        }
    } catch (ExecutionException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
```

Figura 94 - Código de Inicialização do "*Worker*"

Quando chega a altura de o "*Worker*" ativar, um novo processo é ativado na aplicação.

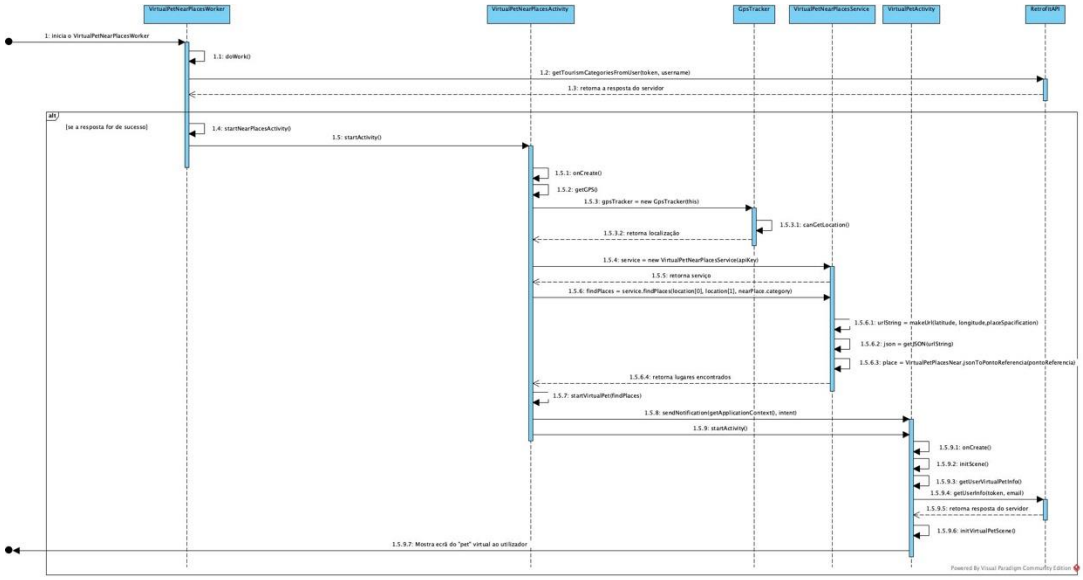


Figura 95 - Diagrama de Sequência de Sugestão de POI pertos do Utilizador

A função "*doWork()"* é acionada que começa por efetuar uma chamada do tipo REST API através da função "*getTourismCategoriesFromUser()*". Esta função, através do "*username*" do utilizador, retorna uma lista de todas as categorias de turismo de interesse/relevo associadas ao utilizador. A seguir, é obtida a data da última ativação do "*Worker*" sendo que, no caso de não existir, é inicializada com a data atual do sistema, senão é analisado se a data guardada difere da atual e, se for o caso, a data guardada é substituída pela data atual do sistema, inicializando o contador a zero. Este contador permite contabilizar o número de vezes que o utilizador nega as sugestões desta funcionalidade num dia. No caso de a chamada da lista de categorias não retornar vazia e o contador não for superior a três, então a função "*startNearPlacesActivity()"* será inicializada.

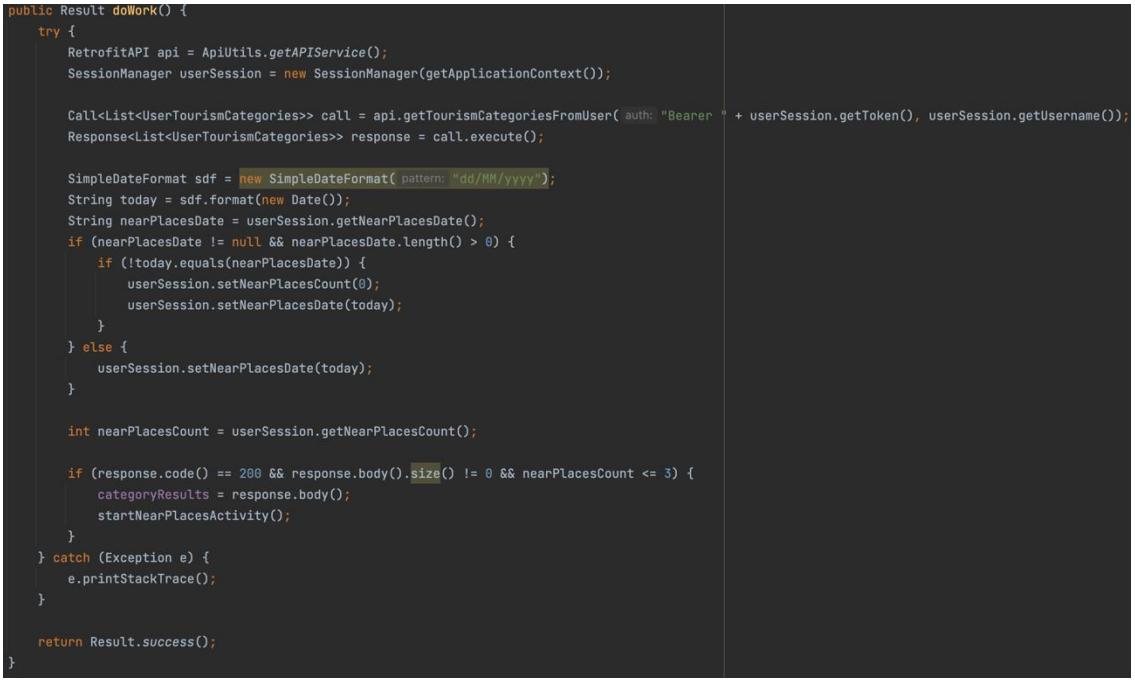


Figura 96 - Código da Função "*doWork*()"

Nesta função, a lista de categorias é ordenada conforme a relevância de cada categoria. Após a ordenação, esta lista é enviada para a atividade "*VirtualPetNearPlacesActivity*" que será inicializada nesta função. A razão pela qual é necessário inicializar uma atividade antes da renderização do "*pet*" virtual prende-se com o facto de ser necessário recolher as coordenadas do dispositivo móvel do utilizador e isto não é possível durante o processamento do "*Worker*".

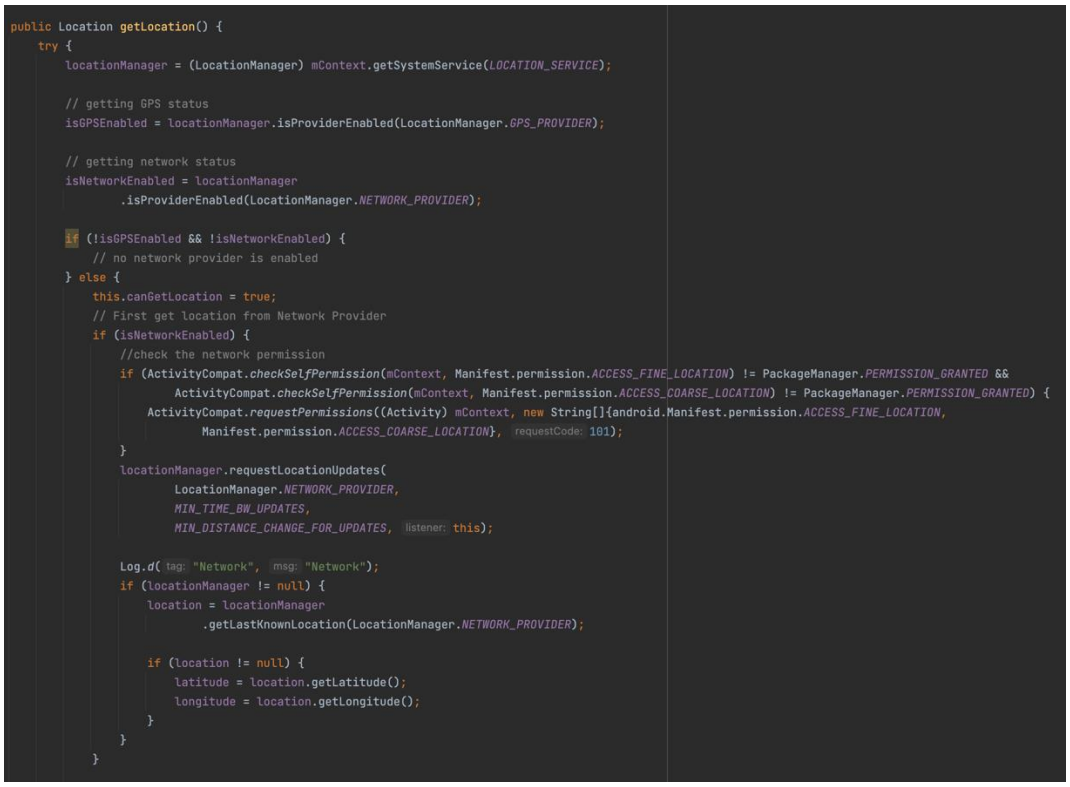
Figura 97 – Código da Função "*startNearPlacesActivity()*"

Após a inicialização da atividade, a função "*onCreate*()" é despoletada e começa com a chamada da função "*getGPS*()" para retornar as coordenadas do dispositivo móvel do utilizador.

```
private double[] getGPS() {
    double[] gps = new double[2];
    GpsTracker gpsTracker = new GpsTracker( context: this);
    if(gpsTracker.canGetLocation()){
        gps[0] = gpsTracker.getLatitude();
        gps[1] = gpsTracker.getLongitude();
    }else{
        gpsTracker.showSettingsAlert();
    }
    return gps;
}
```

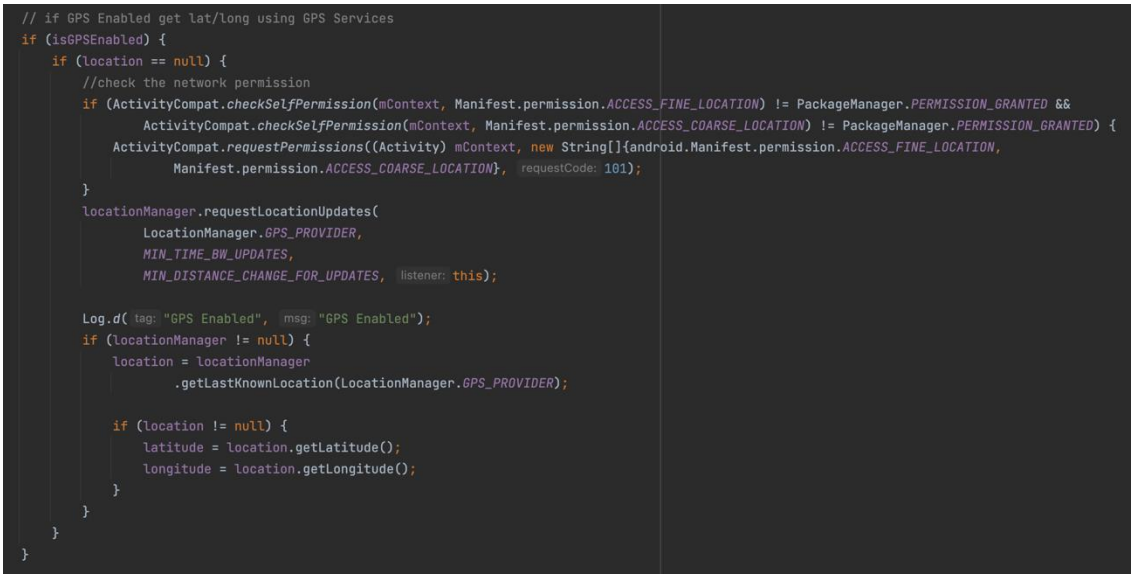
Figura 98 - Código da Função "*getGPS()*"

Para obter as coordenadas, um objeto do tipo "*GpsTracker*" é inicializado sendo que, na sua inicialização, a função "*getLocation()*" é ativada. Nesta função, analisa-se se o GPS e a rede estão ativos no dispositivo do utilizador e caso isto se verifique tenta-se obter as coordenadas geográficas do dispositivo, primeiro pela rede e depois pelo GPS.



```
public Location getLocation() {
    try {
        locationManager = (LocationManager) mContext.getSystemService(LOCATION_SERVICE);

        // getting GPS status
        isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

        // getting network status
        isNetworkEnabled = locationManager
                .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

        if (!isGPSEnabled && !isNetworkEnabled) {
            // no network provider is enabled
        } else {
            this.canGetLocation = true;
            // First get location from Network Provider
            if (isNetworkEnabled) {
                //check the network permission
                if (ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
                        ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions((Activity) mContext, new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION,
                            Manifest.permission.ACCESS_COARSE_LOCATION}, requestCode: 101);
                }
                locationManager.requestLocationUpdates(
                        LocationManager.NETWORK_PROVIDER,
                        MIN_TIME_BW_UPDATES,
                        MIN_DISTANCE_CHANGE_FOR_UPDATES, listener: this);

                Log.d( tag: "Network", msg: "Network");
                if (locationManager != null) {
                    location = locationManager
                            .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

                    if (location != null) {
                        latitude = location.getLatitude();
                        longitude = location.getLongitude();
                    }
                }
            }
```

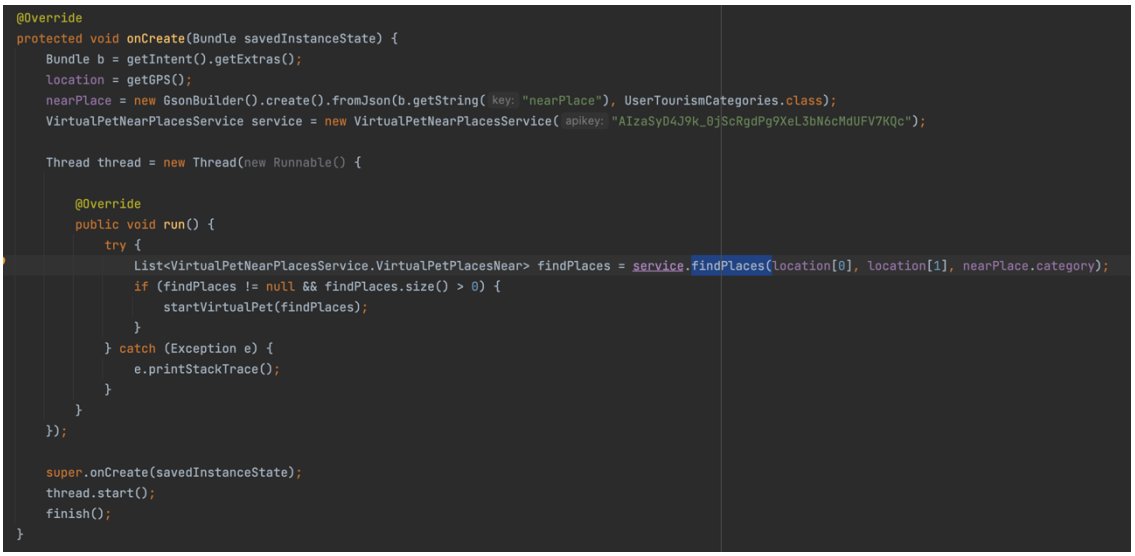
Figura 99 - Código Inicial da Função "*getLocation()*"



```
// if GPS Enabled get lat/long using GPS Services
if (isGPSEnabled) {
    if (location == null) {
        //check the network permission
        if (ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions((Activity) mContext, new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION,
                    Manifest.permission.ACCESS_COARSE_LOCATION}, requestCode: 101);
        }
        locationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER,
                MIN_TIME_BW_UPDATES,
                MIN_DISTANCE_CHANGE_FOR_UPDATES, listener: this);

        Log.d( tag: "GPS Enabled", msg: "GPS Enabled");
        if (locationManager != null) {
            location = locationManager
                    .getLastKnownLocation(LocationManager.GPS_PROVIDER);

            if (location != null) {
                latitude = location.getLatitude();
                longitude = location.getLongitude();
            }
        }
    }
}
```

Figura 100 - Código Final da Função "*getLocation()*"

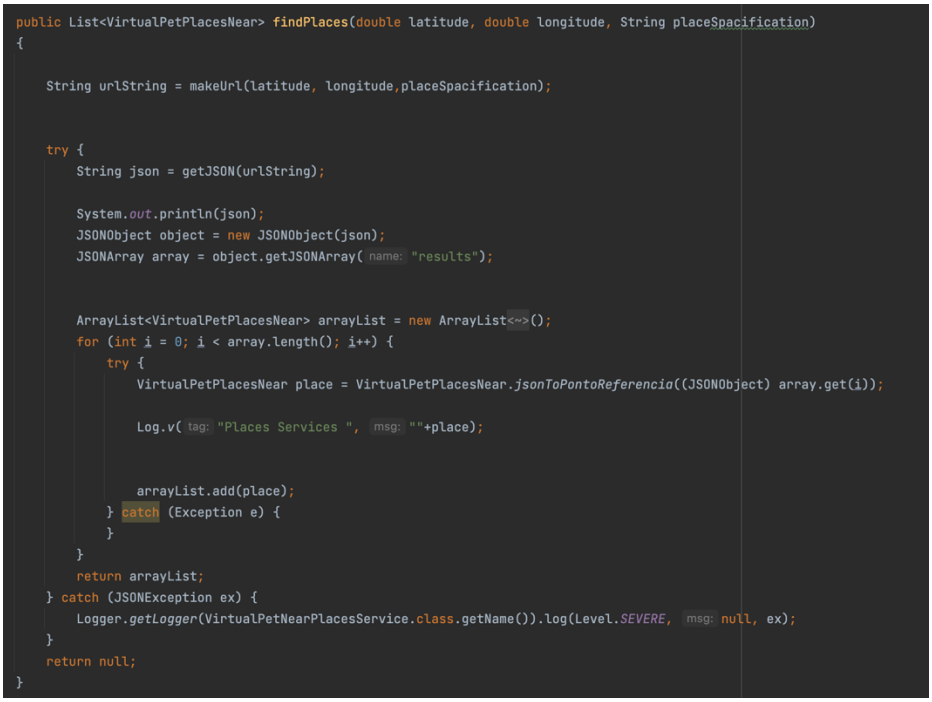
Após a função "*getGPS*()" retornar as coordenadas do dispositivo, um objeto do tipo "*VirtualPetNearPlacesService*" é inicializado com a "*ApiKey*" do Google Places e uma nova "*Thread*" é inicializada. O motivo pelo qual esta "*Thread*" existe, é devido ao objeto do tipo "*VirtualPetNearPlacesService*" que irá efetuar uma chamada Android do tipo "*URLConnection*" e esta não pode ser acionada na mesma "*Thread*" de funcionamento da atividade.



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    Bundle b = getIntent().getExtras();
    location = getGPS();
    nearPlace = new GsonBuilder().create().fromJson(b.getString( key: "nearPlace"), UserTourismCategories.class);
    VirtualPetNearPlacesService service = new VirtualPetNearPlacesService( apikey: "AIzaSyD4J9k_0jScRgdPg9XeL3bN6cMdUFV7KQc");

    Thread thread = new Thread(new Runnable() {

        @Override
        public void run() {
            try {
                List<VirtualPetNearPlacesService.VirtualPetPlacesNear> findPlaces = service.findPlaces(location[0], location[1], nearPlace.category);
                if (findPlaces != null && findPlaces.size() > 0) {
                    startVirtualPet(findPlaces);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });

    super.onCreate(savedInstanceState);
    thread.start();
    finish();
}
```

Figura 101 - Código da Função "*onCreate*()" da "*VirtualPetNearPlacesActivity*"

Nesta "*Thread*", a função "*findPlaces()*" da classe "*VirtualPetNearPlacesService*" é acionada levando como parâmetros a latitude e longitude do dispositivo e a categoria de turismo mais relevante para o utilizador.

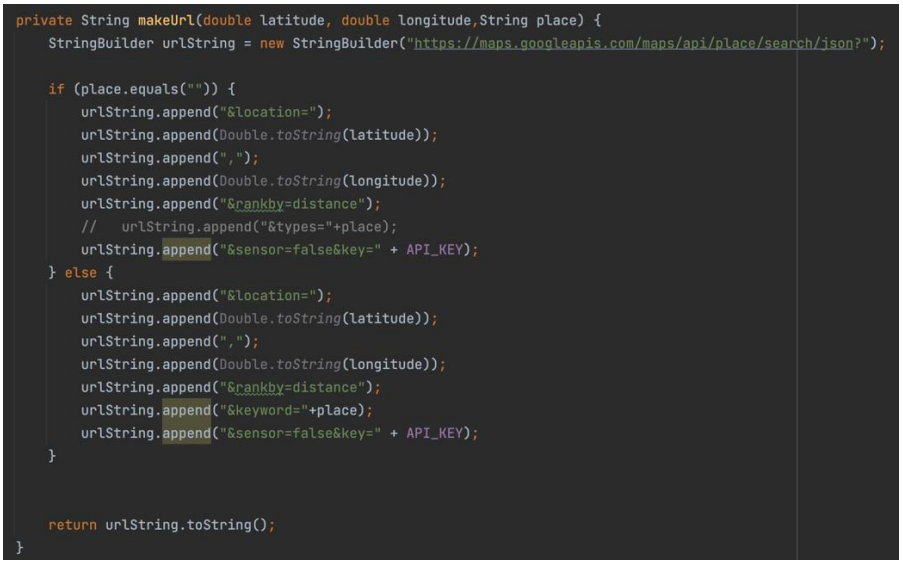


```
public List<VirtualPetPlacesNear> findPlaces(double latitude, double longitude, String placeSpacification)
{

    String urlString = makeUrl(latitude, longitude,placeSpacification);


    try {
        String json = getJSON(urlString);

        System.out.println(json);
        JSONObject object = new JSONObject(json);
        JSONArray array = object.getJSONArray( name: "results");


        ArrayList<VirtualPetPlacesNear> arrayList = new ArrayList<~>();
        for (int i = 0; i < array.length(); i++) {
            try {
                VirtualPetPlacesNear place = VirtualPetPlacesNear.jsonToPontoReferencia((JSONObject) array.get(i));

                Log.v( tag: "Places Services ",  msg: ""+place);


                arrayList.add(place);
            } catch (Exception e) {
            }
        }
        return arrayList;
    } catch (JSONException ex) {
        Logger.getLogger(VirtualPetNearPlacesService.class.getName()).log(Level.SEVERE,  msg: null, ex);
    }
    return null;
}
```

Figura 102 - Código da Função "*findPlaces()*"

Nesta função começa-se por construir o URL para a chamada a API do Google Places através da função "*makeUrl()*". Aqui, utiliza-se o "*host*" da chamada que se pretende efetuar e apenas adicionam-se os parâmetros URL nomeadamente:

- **Location** – parâmetro que contem a latitude e longitude do dispositivo.
- **RankBy** – parâmetro para ordenar os resultados, neste caso definido para a distancia menor.
- **Keyword** – palavra a ser utilizada como termo de procura dos POI que neste caso será a categoria de turismo.
- **Key** – parâmetro que contem a "*ApiKey*".



```
private String makeUrl(double latitude, double longitude,String place) {
    StringBuilder urlString = new StringBuilder("https://maps.googleapis.com/maps/api/place/search/json?");

    if (place.equals("")) {
        urlString.append("&location=");
        urlString.append(Double.toString(latitude));
        urlString.append(",");
        urlString.append(Double.toString(longitude));
        urlString.append("&rankby=distance");
        //   urlString.append("&types="+place);
        urlString.append("&sensor=false&key=" + API_KEY);
    } else {
        urlString.append("&location=");
        urlString.append(Double.toString(latitude));
        urlString.append(",");
        urlString.append(Double.toString(longitude));
        urlString.append("&rankby=distance");
        urlString.append("&keyword="+place);
        urlString.append("&sensor=false&key=" + API_KEY);
    }


    return urlString.toString();
}
```

Figura 103 - Código da Função "*makeUrl()*"

Após retornar a String do URL construído, a função "*getJSON*()" é acionada. Esta função chama o método "*getUrlContents*()" que será responsável por utilizar a String construída e converter um objeto do tipo URL de modo a invocar a função "*openConnection*()" da mesma. Esta função efetuará um pedido GET HTTPS e irá processar a resposta da mesma por meio de um "*BufferedReader*", que transforma todo o seu conteúdo numa única String de modo que esta seja possível ser traduzida para um objeto JSON.

No final, o JSON retornado é manipulado para retirar o Array de resultados de POI num raio de mil metros do utilizador referentes a categoria pesquisada e transformado num novo Array de objetos do tipo "*VirtualPetPlacesNear*" que será utilizado na atividade de renderização do "*pet*" virtual.

```
protected String getJSON(String url) { return getUrlContents(url); }

private String getUrlContents(String theUrl)
{
    StringBuilder content = new StringBuilder();

    try {
        URL url = new URL(theUrl);
        URLConnection urlConnection = url.openConnection();
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(urlConnection.getInputStream()),  sz: 8);
        String line;
        while ((line = bufferedReader.readLine()) != null)
        {
            content.append(line + "\n");
        }

        bufferedReader.close();
    }

    catch (Exception e)
    {

        e.printStackTrace();

    }

    return content.toString();
}
```

Figura 104 - Código da Função "*getUrlContents()*"

No caso de algum POI ser retornado pela classe "*VirtualPetNearPlacesService*", a função "*startVirtualPet()*" é acionada. Esta função é igual a das recomendações individuais, mudando apenas a construção das mensagens, o parâmetro "*nearPlace*" que é o POI encontrado mais perto do utilizador e os Booleans que definem que tipo de funcionalidade é que acionou a renderização do "*pet*" virtual.

```
private void startVirtualPet(List<VirtualPetNearPlacesService.VirtualPetPlacesNear> findPlaces) {
    Intent intent = new Intent(getApplicationContext(), VirtualPetActivity.class).setFlags(FLAG_ACTIVITY_NEW_TASK);
    VirtualPetActivity.sendNotification(getApplicationContext(), intent);

    String message = "The points of interest '" + findPlaces.get(0).getName() + "' is near you and might be of your interest. Do you want to go there?";

    Bundle b = new Bundle();
    b.putString("message", message);
    b.putString("isGroupRecommendation", "false");
    b.putString("isNearPlaceRecommendation", "true");
    b.putString("nearPlace", new GsonBuilder().create().toJson(findPlaces.get(0)));
    intent.putExtras(b);
    startActivity(intent, b);
}
```

Figura 105 - Código da Função "*startVirtualPet()*"

Daqui em diante, o processo é igual ao das recomendações individuais mencionado acima, resultando a renderização do "*pet*" virtual a recomendar a visita por parte do utilizador ao POI recomendado.

Após a sugestão do POI por parte do "*pet*" virtual o utilizador pode negar ou aceitar a recomendação.

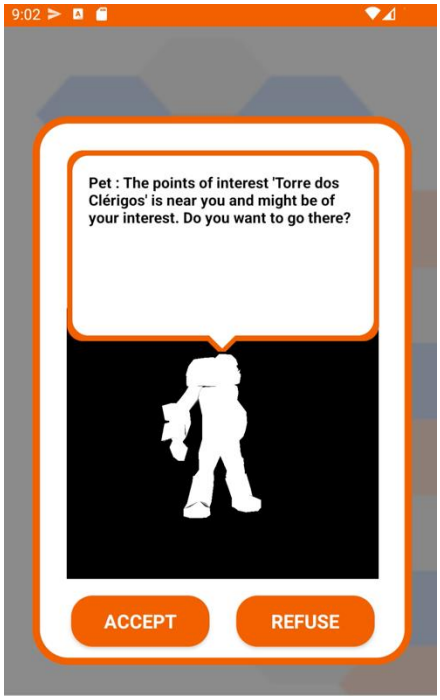


Figura 106 - Ecrã da Aplicação para Recusa ou Aceitação da Recomendação de POI perto do Utilizador

No caso de recusa, o processo é bastante semelhante às recomendações individuais e de grupo detalhadas no último requisito.

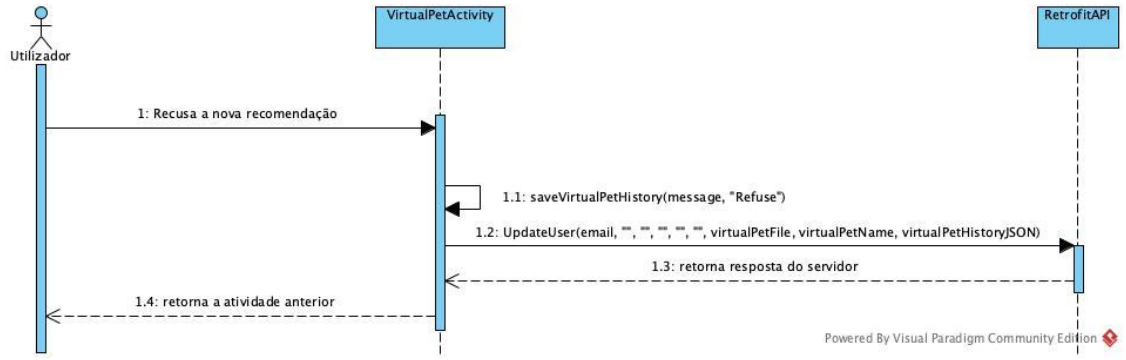


Figura 107 - Diagrama de Sequência de Negação da Recomendação de POI

A única diferença prende-se com o contador referido anteriormente na função "*doWork()*". Em caso de recusa, este contador é incrementado por um para contar o número de vezes que o utilizador recusou as recomendações.

```
public void onClick(View v) {
    if (isNearPlaceRecommendation.equals("true")) {
        userSession.setNearPlacesCount(userSession.getNearPlacesCount() + 1);
    }
    saveVirtualPetHistory(message, response: "Refuse");
    finish();
}
```

Figura 108 - Código de Negação do POI Sugerido

No caso do utilizador aceitar a recomendação do "*pet*" virtual, o evento "*onClick()*" do botão de aceitação é ativado.

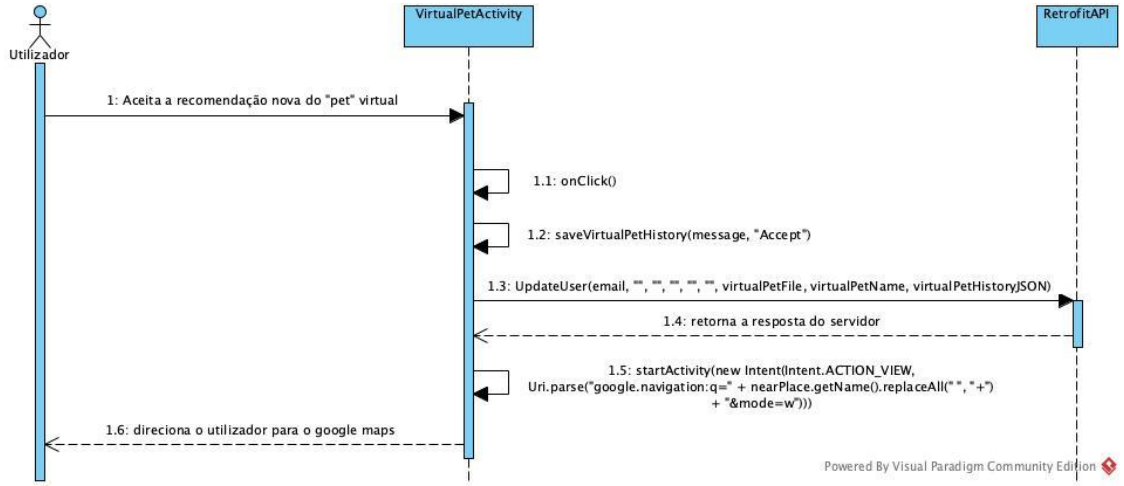


Figura 109 - Diagrama de Sequência de Aceitação da Recomendação de POI

Nesta função, temos na mesma presente todo o processo de guardar o histórico de interações do utilizador e o "*pet*" virtual através da função "*saveVirtualPetHistory()*". No entanto, temos também a construção de um URI destinado à aplicação do Google Maps. Este URI irá conter o nome do POI a visitar e o modo de passeio (ou seja, sinalizar que o utilizador se encontra a pé) como parâmetros e será usado para inicializar a aplicação do Google Maps com as indicações corretas de modo a guiar o utilizador para o POI que ele aceitou visitar.

```
public void onClick(View v) {
    if (isNearPlaceRecommendation.equals("true")) {
        Uri gmmIntentUri = Uri.parse("google.navigation:q=" + nearPlace.getName().replaceAll( regex: " ", replacement: "+") + "&mode=w");
        Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
        mapIntent.setPackage("com.google.android.apps.maps");
        saveVirtualPetHistory(message, response: "Accept");
        startActivity(mapIntent);
        finish();
```

Figura 110 - Código de Aceitação da Recomendação de POI

5 Experimentação e Avaliação

Neste capítulo são avaliadas e documentadas as experiências efetuadas em torno do protótipo do "*pet*" virtual e todas as funcionalidades inerentes. Primeiro são definidos os critérios a serem provados, depois descrevem-se os métodos de avaliação utilizados para comprovar os critérios definidos e, finalmente, todos os dados obtidos no processo de avaliação são analisados.

5.1 Critérios de Avaliação

A avaliação do protótipo do "*pet*" virtual está dividido em dois pontos:

- **Avaliação do funcionamento e desempenho do sistema** – é fundamental garantir que o protótipo desenvolvido é desprovido de falhas. Uma análise detalhada é necessária para detetar e corrigir possíveis erros que sejam encontrados. Para isso deve-se verificar que todas as funcionalidades relacionadas com o "*pet*" virtual não possuem falhas que quebrem a funcionalidade, que todas as chamadas efetuadas possuem uma boa comunicação e são concluídas numa janela temporal aceitável.
- **Avaliação da satisfação do utilizador** – é muito importante garantir que o utilizador final se encontra satisfeito com o protótipo desenvolvido. A opinião do utilizador final da renderização do "*pet*" virtual e das funcionalidades inerentes é crucial para identificar problemas e aspetos a melhorar na solução desenvolvida.

Para se obter conclusões, foram utilizados testes de hipótese para fazer uma comparação entre a hipótese nula (H0) e as outras hipóteses para rejeitar ou não a hipótese nula.

No que toca à "Avaliação do funcionamento e desempenho do sistema" dois critérios foram definidos:

Critério 1: Avaliação do funcionamento do protótipo

- H0 – Os testes realizados à aplicação não apresentam falhas.
- H1 – Os testes realizados à aplicação apresentam falhas.

Critério 2: Avaliação do desempenho do protótipo

- H0 – As funcionalidades descritas não demoram mais de dez segundos a serem executadas.
- H1 – As funcionalidades descritas demoram mais de dez segundos a serem executadas.

No que toca à "Avaliação da satisfação do utilizador" apenas um critério foi definido:

Critério 3: Avaliação do nível de satisfação do utilizador

- H0 – As elações retiradas das repostas a um inquérito pelos utilizadores são positivas.
- H1 – As elações retiradas das repostas a um inquérito pelos utilizadores são negativas.

5.2 Metodologia de Avaliação

Para os diferentes tipos de critérios definidos acima foram idealizados e aplicados diferentes tipos de metodologias para a sua avaliação. Para avaliar o critério 1, foram efetuados testes ao "*software*", nomeadamente testes de integração, para verificar que a qualidade da comunicação entre a aplicação Android e todos os serviços externos é satisfatória e testes de aceitação para verificar a ausência de falhas no fluxo de todas as funcionalidades descritas no capítulo anterior.

No que toca ao critério 2, foram efetuadas 15 tentativas para cada funcionalidade do "*pet*" virtual, utilizando sempre o mesmo perfil de utilizador, registando a duração de cada tentativa para obter uma média do tempo de duração para rejeitar a H0. O tempo de dez segundos foi idealizado conforme a média de tempo de processamento das outras funcionalidades já presentes no "*software*".

Relativamente ao critério 3, um grupo de utilizadores foi convidado a experimentar as novas funcionalidades do "*pet*" virtual, seguindo um guião de experimentação detalhado para partilhar a sua opinião por meio de inquéritos. Estes inquéritos são constituídos por um pré-inquérito (Anexo B) que avalia o público que vai utilizar a aplicação e um pós-inquérito (Anexo C) para avaliar a opinião dos utilizadores nos aspetos relacionados com a utilidade das funcionalidades do "*pet*" virtual. Ambos os formulários foram feitos na plataforma Google Forms, sendo que a todos os participantes foi explicado o objetivo e motivação do estudo efetuado e pedido o seu consentimento informado no que toca à obtenção de dados pessoais dos mesmos e à participação no estudo.

5.3 Análise da Avaliação

Nesta secção avaliamos todos os resultados obtidos para os critérios descritos neste capítulo.

5.3.1 Análise do Critério 1

Para avaliar este critério, foram efetuados vários testes descritos na secção acima.

O caso de teste "Criar utilizador com todos os campos do *pet* virtual preenchidos" tem o objetivo de verificar se o utilizador consegue criar um perfil com sucesso e com toda a informação do "*pet*" virtual guardada.

Tabela 13 - Caso de Teste: Criar utilizador com todos os campos do "*pet*" virtual preenchidos

Criar utilizador com todos os campos do "*pet*" virtual preenchidos	
Passos a Efetuar	**Resultado Obtido**
1. Aceder à aplicação 2. Carregar no link para criar conta 3. Preencher todos os campos do formulário de criação de conta 4. Submeter o formulário 5. Submeter o código recebido no email usado na criação de conta 6. Aceder à página de perfil 7. Aceder à página de edição do "*pet*" virtual 8. Confirmar os campos do "*pet*" virtual	O utilizador consegue criar uma conta com sucesso e com todos os campos preenchidos e guardados corretamente.

O caso de teste "Criar utilizador sem todos os campos do *pet* virtual preenchidos" tem o objetivo de verificar se o utilizador consegue criar um perfil mesmo não tendo todos os campos do "*pet*" virtual preenchidos.

Tabela 14 - Caso de Teste: Criar utilizador sem todos os campos do "*pet*" virtual preenchidos

Criar utilizador sem todos os campos do "*pet*" virtual preenchidos	
Passos a Efetuar	**Resultado Obtido**
1. Aceder à aplicação 2. Carregar no link para criar conta 3. Preencher todos os campos do formulário de criação de conta menos os referentes ao "pet" virtual 4. Submeter o formulário	O utilizador não consegue criar a conta e é alertado para os campos em falta.

O caso de teste "Editar e guardar os campos do *pet* virtual do perfil do utilizador completamente preenchidos" tem o objetivo de verificar se o utilizador consegue editar os campos do "*pet*" virtual com sucesso e guardar toda a informação do "*pet*" virtual completamente preenchida.

Tabela 15 - Caso de Teste: Editar e guardar os campos do "*pet*" virtual do perfil do utilizador completamente preenchidos

Editar e guardar os campos do "*pet*" virtual do perfil do utilizador completamente preenchidos	
Passos a Efetuar	**Resultado Obtido**

1. Aceder à aplicação 2. Fazer login na conta do utilizador 3. Aceder à página de perfil 4. Aceder à página de edição do "pet" virtual 5. Alterar os campos do "pet" virtual, sem deixar nenhum campo incompleto ou vazio 6. Submeter as alterações	O utilizador consegue guardar as alterações efetuadas.

O caso de teste "Editar e guardar os campos do *pet* virtual do perfil do utilizador sem estar completamente preenchidos" tem o objetivo de verificar se o utilizador consegue editar os campos do "*pet*" virtual com sucesso e guardar toda a informação do "*pet*" virtual mesmo sem estar completamente preenchida.

Tabela 16 - Caso de Teste: Editar e guardar os campos do "*pet*" virtual do perfil do utilizador sem estar completamente preenchidos

Editar e guardar os campos do "*pet*" virtual do perfil do utilizador sem estar completamente preenchidos	
Passos a Efetuar	**Resultado Obtido**
1. Aceder à aplicação 2. Fazer login na conta do utilizador 3. Aceder à página de perfil 4. Aceder à página de edição do "*pet*" virtual 5. Alterar os campos do "*pet*" virtual, deixando campos incompletos ou vazios 6. Submeter as alterações	O utilizador não consegue guardar as alterações e é alertado para os campos em falta.

O caso de teste "Consultar o histórico de interações do *pet* virtual" tem o objetivo de verificar se o utilizador consegue aceder e consultar o histórico de interações que teve com o "*pet*" virtual.

Tabela 17 - Caso de Teste: Consultar o histórico de interações do "*pet*" virtual

Consultar o histórico de interações do "*pet*" virtual	
Passos a Efetuar	**Resultado Obtido**
1. Aceder à aplicação 2. Fazer login na conta do utilizador 3. Aceder à página de perfil 4. Aceder à página de edição do "*pet*" virtual 5. Carregar no "*tab*" do histórico das interações entre o utilizador e o "pet" virtual	O utilizador é capaz de consultar o histórico de interações no menu de edição do "*pet*" virtual.

O caso de teste "Receber uma nova recomendação individual por parte do *pet* virtual após pedir recomendações para uma excursão" tem o objetivo de verificar se o utilizador recebe uma nova sugestão de recomendação individual por parte do "*pet*" virtual para substituir uma recomendação que não pode visitar de momento.

Tabela 18 - Caso de Teste: Receber uma nova recomendação individual por parte do "*pet*" virtual após pedir recomendações para uma excursão

Receber uma nova recomendação individual por parte do "*pet*" virtual após pedir recomendações para uma excursão	
Passos a Efetuar	**Resultado Obtido**
1. Aceder à aplicação 2. Fazer login na conta do utilizador 3. Aceder à página de recomendações individuais de pontos de interesse 4. Gerar uma nova lista de recomendações com a data de início da excursão igual a data atual e a data de fim da excursão dois dias após a data atual 5. Verificar as recomendações 6. Esperar quinze minutos 7. Verificar se recebe uma nova sugestão de recomendação individual por parte do "*pet*" virtual	O utilizador é capaz de receber uma nova recomendação individual para uma excursão quando não é possível visitar um ponto de interesse recomendado previamente.

O caso de teste "Receber uma nova recomendação de grupo por parte do *pet* virtual após pedir recomendações para uma excursão de grupo" tem o objetivo de verificar se o utilizador recebe uma nova sugestão de recomendação para os subgrupos do seu grupo de excursão por parte do "*pet*" virtual para substituir uma recomendação que não pode visitar de momento.

Tabela 19 - Caso de Teste: Receber uma nova recomendação de grupo por parte do "*pet*" virtual após pedir recomendações para uma excursão de grupo

Receber uma nova recomendação de grupo por parte do "*pet*" virtual após pedir recomendações para uma excursão de grupo	
Passos a Efetuar	**Resultado Obtido**
1. Aceder à aplicação 2. Fazer login na conta do utilizador que seja líder de um grupo de excursão 3. Aceder à página de seleção de grupos 4. Escolher o grupo do qual o utilizador é líder	O utilizador é capaz de receber uma nova recomendação de grupo para um subgrupo de uma excursão quando não é possível visitar um ponto de interesse recomendado previamente.

5. Gerar novas recomendações de grupo com a data de início da excursão igual à data atual e a data de fim da excursão dois dias após a data atual 6. Verificar subgrupos criados e as recomendações associadas 7. Esperar quinze minutos 8. Verificar se recebe uma nova sugestão de recomendação para os subgrupos criados por parte do "*pet*" virtual	

O caso de teste "Rejeitar recomendação individual por parte do *pet* virtual" tem o objetivo de verificar se a interação entre o "*pet*" virtual e o utilizador fica guardada no histórico e se nenhuma recomendação é alterada no processo.

Tabela 20 - Caso de Teste: Rejeitar recomendação individual por parte do "*pet*" virtual

Rejeitar recomendação individual por parte do "*pet*" virtual	
Passos a Efetuar	**Resultado Obtido**
1. Repetir passos referidos no caso de teste "Receber uma nova recomendação individual por parte do *pet* virtual após pedir recomendações para uma excursão" 2. Rejeitar a recomendação do "*pet*" virtual 3. Aceder à página de recomendações individuais 4. Verificar se nenhuma alteração foi efetuada 5. Aceder à página de perfil do utilizador 6. Aceder à página de edição do "*pet*" virtual 7. Carregar no "*tab*" do histórico do "*pet*" virtual 8. Verificar se uma nova entrada na tabela foi efetuada relatando a interação de rejeição da sugestão	O utilizador ao rejeitar a recomendação, nenhuma alteração é efetuada a lista de recomendações da excursão e a interação é guardada no histórico do "*pet*" virtual.

O caso de teste "Rejeitar recomendação de grupo por parte do *pet* virtual" tem o objetivo de verificar se a interação entre o "*pet*" virtual e o utilizador fica guardada no histórico e se nenhuma recomendação dos subgrupos é alterada no processo.

Tabela 21 - Caso de Teste: Rejeitar recomendação de grupo por parte do "*pet"* virtual

Rejeitar recomendação de grupo por parte do "*pet*" virtual	
Passos a Efetuar	**Resultado Obtido**
1. Repetir passos referidos no caso de teste "Receber uma nova recomendação de grupo por parte do *pet* virtual após pedir recomendações para uma excursão de grupo" 2. Rejeitar a recomendação do "*pet*" virtual 3. Aceder à página de seleção de grupos de excursão 4. Selecionar o grupo referido na sugestão do "*pet*" virtual 5. Verificar se nenhuma alteração foi efetuada a lista de recomendações de cada subgrupo 6. Aceder à página de perfil do utilizador 7. Aceder à página de edição do "*pet*" virtual 8. Carregar no "*tab*" do histórico do "*pet*" virtual 9. Verificar se uma nova entrada na tabela foi efetuada relatando a interação de rejeição da sugestão	O utilizador ao rejeitar a recomendação, nenhuma alteração é efetuada a lista de recomendações dos subgrupos da excursão e a interação é guardada no histórico do "*pet*" virtual.

O caso de teste "Aceitar recomendação individual por parte do *pet* virtual" tem o objetivo de verificar se a interação entre o "*pet*" virtual e o utilizador fica guardada no histórico e se a recomendação sugerida é alterada no processo.

Tabela 22 - Caso de Teste: Aceitar recomendação individual por parte do "*pet*" virtual

Aceitar recomendação individual por parte do "*pet*" virtual	
Passos a Efetuar	**Resultado Obtido**
1. Repetir passos referidos no caso de teste "Receber uma nova recomendação individual por parte do *pet* virtual após pedir recomendações para uma excursão" 2. Aceitar a recomendação do "*pet*" virtual 3. Aceder à página de recomendações individuais 4. Verificar se a alteração foi efetuada	O utilizador ao aceitar a recomendação, a lista de recomendações é alterada com a nova recomendação e a interação é guardada no histórico do "*pet*" virtual.

5. Aceder à página de perfil do utilizador 6. Aceder à página de edição do "*pet*" virtual 7. Carregar no "*tab*" do histórico do "*pet*" virtual 8. Verificar se uma nova entrada na tabela foi efetuada relatando a interação de aceitação da sugestão	

O caso de teste "Aceitar recomendação de grupo por parte do *pet* virtual" tem o objetivo de verificar se a interação entre o "*pet*" virtual e o utilizador fica guardada no histórico e as recomendações dos subgrupos sugeridas são alteradas no processo.

Tabela 23 - Caso de Teste: Aceitar recomendação de grupo por parte do "*pet*" virtual

Aceitar recomendação de grupo por parte do "*pet*" virtual	
Passos a Efetuar	**Resultado Obtido**
1. Repetir passos referidos no caso de teste "Receber uma nova recomendação de grupo por parte do *pet* virtual após pedir recomendações para uma excursão de grupo" 2. Aceitar a recomendação do "*pet*" virtual 3. Aceder à página de seleção de grupos de excursão 4. Selecionar o grupo referido na sugestão do "*pet*" virtual 5. Verificar se as alterações foram efetuadas na lista de recomendações de cada subgrupo 6. Aceder à página de perfil do utilizador 7. Aceder à página de edição do "*pet*" virtual 8. Carregar no "*tab*" do histórico do "*pet*" virtual 9. Verificar se uma nova entrada na tabela foi efetuada relatando a interação de aceitação da sugestão	O utilizador ao aceitar a recomendação de grupo, a lista de recomendações do subgrupo é alterada com a nova recomendação e a interação é guardada no histórico do "*pet*" virtual.

O caso de teste "Receber uma nova recomendação por parte do *pet* virtual perto da localização do utilizador" tem o objetivo de verificar se o utilizador recebe uma nova sugestão de recomendação por parte do "*pet*" virtual perto do utilizador para ele visitar.

Tabela 24 - Caso de Teste: Receber uma nova recomendação por parte do "*pet*" virtual perto da localização do utilizador

Receber uma nova recomendação por parte do "*pet*" virtual perto da localização do utilizador	
Passos a Efetuar	**Resultado Obtido**
1. Aceder à aplicação 2. Efetuar o login de conta 3. Caminhar perto do centro de uma grande cidade (ex. baixa do Porto) 4. Esperar quinze minutos 5. Verificar se o "*pet*" virtual recomenda a visita de um POI perto da localização atual do utilizador	O utilizador é capaz de receber uma nova recomendação de pontos de interesse para visitar perto dele.

O caso de teste "Aceitar a nova recomendação por parte do *pet* virtual perto da localização do utilizador" tem o objetivo de verificar se o utilizador, ao aceitar a nova sugestão de recomendação por parte do "*pet*" virtual perto dele, é direcionado para a localização do ponto de interesse e a interação entre ambos é guardada no histórico do "*pet*" virtual.

Tabela 25 - Caso de Teste: Aceitar a nova recomendação por parte do "*pet*" virtual perto da localização do utilizador

Aceitar a nova recomendação por parte do "*pet*" virtual perto da localização do utilizador	
Passos a Efetuar	**Resultado Obtido**
1. Repetir passos do caso de teste "Receber uma nova recomendação por parte do *pet* virtual perto da localização do utilizador" 2. Aceitar a recomendação do "*pet*" virtual 3. Verificar se o utilizador é reencaminhado para o Google Maps com o trajeto da localização atual do mesmo até ao POI recomendado 4. Aceder à página de perfil do utilizador 5. Aceder à página de edição do "*pet*" virtual 6. Carregar no "*tab*" do histórico do "*pet*" virtual 7. Verificar se uma nova entrada na tabela foi efetuada relatando a interação de aceitação da sugestão	O utilizador, ao aceitar a nova recomendação, é direcionado para ela por parte da aplicação e a interação entre ambos é guardada no histórico.

O caso de teste "Rejeitar a nova recomendação por parte do *pet* virtual perto da localização do utilizador" tem o objetivo de verificar se o utilizador, ao rejeitar a nova sugestão

de recomendação por parte do "*pet*" virtual perto dele, apenas a interação entre ambos é guardada no histórico do "*pet*" virtual.

Tabela 26 - Caso de Teste: Rejeitar a nova recomendação por parte do "*pet*" virtual perto da localização do utilizador

Rejeitar a nova recomendação por parte do "*pet*" virtual perto da localização do utilizador	
Passos a Efetuar	**Resultado Obtido**
1. Repetir passos do caso de teste "Receber uma nova recomendação por parte do *pet* virtual perto da localização do utilizador" 2. Rejeitar a recomendação do "*pet*" virtual 3. Aceder à página de perfil do utilizador 4. Aceder à página de edição do "*pet*" virtual 5. Carregar no "*tab*" do histórico do "*pet*" virtual 6. Verificar se uma nova entrada na tabela foi efetuada relatando a interação de aceitação da sugestão	O utilizador, ao rejeitar a nova recomendação, apenas a interação entre ambos é guardada no histórico.

O caso de teste "Rejeitar mais de três vezes nova recomendação por parte do *pet* virtual perto da localização do utilizador" tem o objetivo de verificar se o utilizador, ao rejeitar mais de três vezes a nova sugestão de recomendação por parte do "*pet*" virtual perto dele, se esta funcionalidade não é despoletada mais nenhuma vez durante o dia.

Tabela 27 - Caso de Teste: Rejeitar mais de três vezes nova recomendação por parte do "*pet*" virtual perto da localização do utilizador

Rejeitar mais de três vezes nova recomendação por parte do "*pet*" virtual perto da localização do utilizador	
Passos a Efetuar	**Resultado Obtido**
1. Repetir o caso de teste "Rejeitar a nova recomendação por parte do *pet* virtual perto da localização do utilizador" três vezes no mesmo dia 2. Esperar quinze minutos 3. Verificar se mais nenhuma sugestão é feita pelo "*pet*" virtual	O utilizador, ao rejeitar a nova recomendação mais de três vezes, não recebe mais recomendações deste tipo.

5.3.2 Análise do Critério 2

Para executar a metodologia descrita para o critério 2, foram cronometrados os diferentes processos das funcionalidades desenvolvidas utilizando um computador com uma ligação WI-FI com 50.04 Mbps de transferência e 36.06 Mbps de carregamento. A tabela seguinte apresenta os resultados da aplicação desta metodologia por funcionalidade.

Tabela 28 - Analise do Tempo de Resposta de Cada Funcionalidade Desenvolvida

Criar "*Pet*" Virtual		
Menor Tempo (s)	Maior Tempo (s)	Tempo Médio (s)
5.40	8.17	7.60
Editar "*Pet*" Virtual		
Menor Tempo (s)	Maior Tempo (s)	Tempo Médio (s)
3.14	6.60	4.45
Enviar notificações para alterar recomendações de acordo com o contexto da atividade		
Menor Tempo (s)	Maior Tempo (s)	Tempo Médio (s)
8.1	10.51	9.32
Receber notificações a direcionar para novas recomendações de POI próximos		
Menor Tempo (s)	Maior Tempo (s)	Tempo Médio (s)
7.38	9.98	8.87

Após a análise de todas as funcionalidades, concluímos que todas elas cumprem a hipótese estipulada ao respeitar o tempo limite de 10 segundos.

5.3.3 Análise do Critério 3

Para a análise deste critério, foram questionadas vinte pessoas aleatórias utilizando o pré-questionário disponível no Anexo B. Este grupo era constituído maioritariamente por pessoas do sexo masculino, com as idades compreendidas entre os vinte cinco e os quarenta anos, empregadas por conta de outrem e com o grau de licenciatura. De realçar que esta amostra de pessoas, viaja com alguma frequência, utiliza normalmente aplicações ligadas ao turismo como método de auxílio para a suas viagens e encontra-se familiarizado com o conceito de "*pet*" virtual utilizado em aplicações.

Este mesmo grupo de pessoas, foi convidado a experimentar as funcionalidades desenvolvidas em torno do "*pet*" virtual, seguindo um guião dividido em vários casos.

Caso 1

1. Aceder a aplicação por meio de um dispositivo móvel Android
2. Iniciar o processo de criar uma conta
3. Preencher todos os campos do formulário de criação de conta com a exceção dos campos do "*pet*" virtual
4. Submeter o formulário

5. Preencher os campos do formulário referente ao "*pet*" virtual
6. Submeter o formulário
7. Completar o processo de criação de conta
8. Aceder ao perfil do utilizador na aplicação
9. Carregar no ícone do "*pet*" virtual
10. Verificar se toda a informação inserida no formulário de criação de conta foi guardada

Caso 2

1. Aceder ao perfil do utilizador na aplicação
2. Carregar no ícone do "*pet*" virtual
3. Apagar a informação dos campos do "*pet*" virtual
4. Submeter o formulário
5. Editar os campos do "*pet*" virtual com novos valores
6. Submeter o formulário
7. Repetir os passos 1 e 2
8. Verificar se a informação foi guardada

Caso 3

1. Abrir a aplicação GrouPlanner
2. Efetuar o "*login*" de conta
3. Verificar a lista de recomendações de grupo associadas a conta
4. Aguardar pelo ajuste de recomendações contextual
5. Verificar a sugestão de ponte de interesse a ser substituído pelo "*pet*" virtual
6. Rejeitar a sugestão do "*pet*" virtual
7. Verificar a lista de recomendações de grupo associadas a conta para confirmar que nenhuma alteração foi efetuada
8. Repetir passos 4 e 5
9. Aceitar a sugestão do "*pet*" virtual
10. Aceder ao perfil do utilizador na aplicação
11. Carregar no ícone do "*pet*" virtual
12. Carregar no "*tab*" de histórico do "*pet*" virtual
13. Verificar se as interações efetuadas com o "*pet*" virtual foram guardadas
14. Verificar a lista de recomendações de grupo associadas a conta para confirmar que a alteração foi efetuada

Caso 4

1. Abrir a aplicação GrouPlanner
2. Efetuar o "*login*" de conta

3. Verificar a lista de recomendações individuais associadas a conta
4. Aguardar pelo ajuste de recomendações contextual
5. Verificar a sugestão de ponto de interesse a ser substituído pelo "*pet*" virtual
6. Rejeitar a sugestão do "*pet*" virtual
7. Verificar a lista de recomendações individuais associadas a conta para confirmar que nenhuma alteração foi efetuada
8. Repetir passos 4 e 5
9. Aceitar a sugestão do "*pet*" virtual
10. Aceder ao perfil do utilizador na aplicação
11. Carregar no ícone do "*pet*" virtual
12. Carregar no "*tab*" de histórico do "*pet*" virtual
13. Verificar se as interações efetuadas com o "*pet*" virtual foram guardadas
14. Verificar a lista de recomendações individuais associadas a conta para confirmar que a alteração foi efetuada

Caso 5

1. Abrir a aplicação GrouPlanner
2. Efetuar o "*login*" de conta
3. Deslocar-se até o centro da cidade do Porto (por exemplo, Praça dos Aliados)
4. Durante 15 minutos, efetuar um passeio pela cidade
5. Após os 15 minutos, verificar a sugestão de ponto de interesse a visitar pelo "*pet*" virtual perto do utilizador
6. Rejeitar a sugestão do "*pet*" virtual
7. Repetir passos 4 e 5
8. Aceitar a sugestão do "*pet*" virtual
9. Verificar se o sistema direciona para o ponto de interesse sugerido
10. Repetir passos 4, 5 e 6
11. Repetir novamente os passos 4, 5 e 6
12. Repetir o passo 4 e verificar se nova sugestão é feita pelo "*pet*" virtual
13. Aceder ao perfil do utilizador na aplicação
14. Carregar no ícone do "*pet*" virtual
15. Carregar no "*tab*" de histórico do "*pet*" virtual
16. Verificar se as interações efetuadas com o "*pet*" virtual foram guardadas

A tabela seguinte, descreve as respostas obtidas no pós-questionário presente no Anexo B. Cada pergunta possui respostas que seguem uma escala de valores de um a cinco, onde um corresponde a "Discordo Totalmente" e cinco a "Concordo Totalmente".

Tabela 29 - Análise do Pós-Questionários de Validação do Sistema

Questões	**1**	**2**	**3**	**4**	**5**

1. Em relação à criação do "*pet*" virtual, a funcionalidade encontra-se bem implementada	0	0	0	8	12
2. Em relação à edição do "*pet*" virtual, a funcionalidade encontra-se bem implementada	0	0	2	8	10
3. A funcionalidade de recomendações individuais contextuais está bem implementada	0	2	3	6	9
4. A funcionalidade de recomendações de grupo contextuais está bem implementada	0	3	5	5	7
5. A funcionalidade de recomendações de POI perto do utilizador está bem implementada	0	0	3	12	5
6. Existe um alto nível de personalização do "*pet*" virtual	7	10	3	0	0
7. Os modelos escolhidos para o "*pet*" virtual são adequados	0	2	6	7	5
8. A implementação do "*pet*" virtual faz me ter mais empatia com a aplicação	0	0	7	9	4
9. A implementação do "*pet*" virtual faz me ter mais foco e atração pela aplicação	0	0	4	8	8
10. A utilização do "*pet*" virtual para as funcionalidades desenvolvidas é vantajoso	0	0	10	7	3
11. A utilização do "*pet*" virtual deve expandir-se para outras funcionalidades	0	2	5	10	3
12. O nível de interação entre utilizador e "*pet*" virtual é elevado	4	10	6	0	0

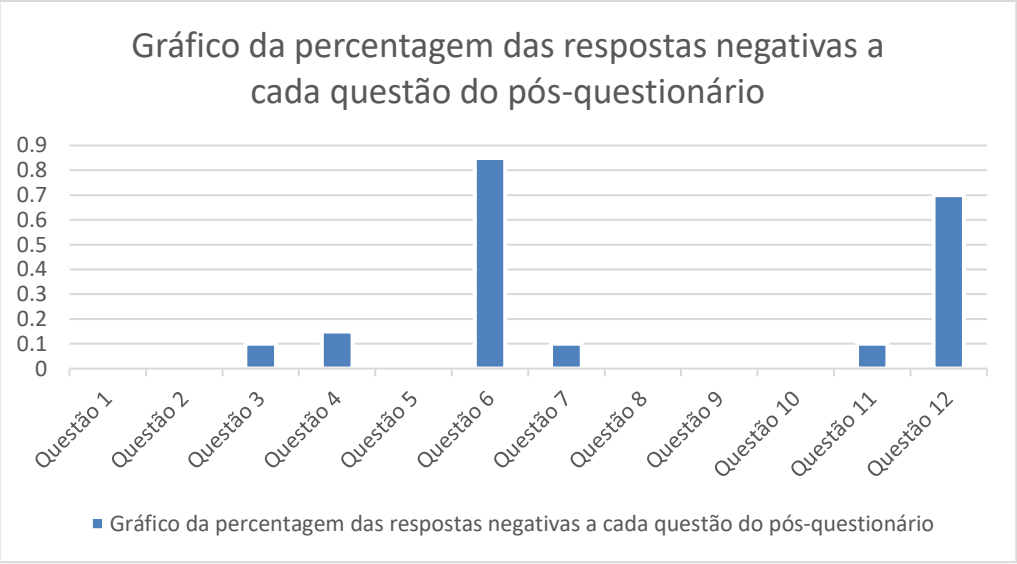


Figura 111 - Gráfico da Percentagem das Respostas Negativas a Cada Questão do Pós-Questionário

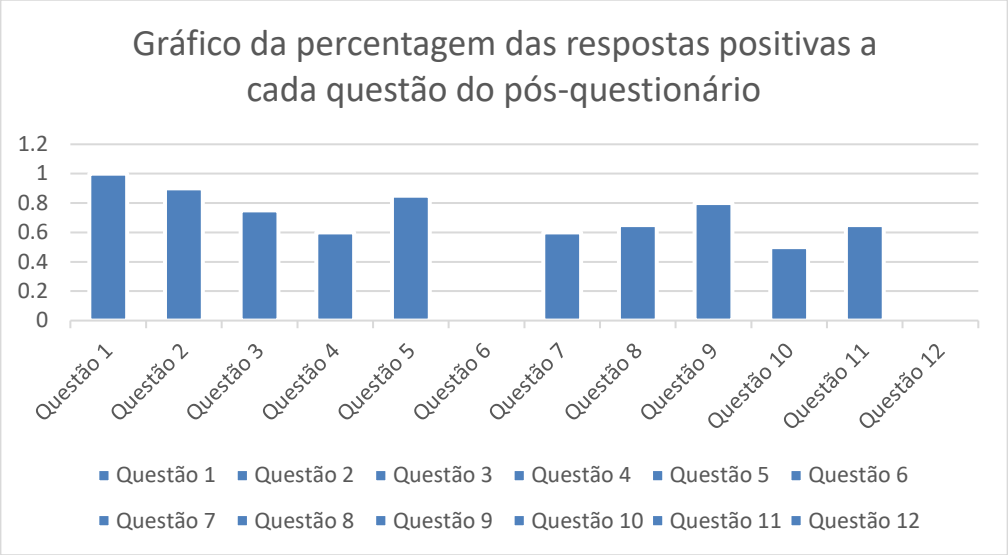


Figura 112 - Gráfico da Percentagem das Respostas Positivas a Cada Questão do Pós-Questionário

Considerando as hipóteses definidas para este critério e analisando a tabela apresentada acima, concluímos que existe uma satisfação geral pelas funcionalidades desenvolvidas para o sistema. De realçar também a apreciação por parte do grupo de estudo para o envolvimento do "*pet*" virtual nas funcionalidades de recomendação do sistema e no sentimento de maior atração e empatia pelos mesmos em torno da aplicação devido à presença do "*pet*" virtual como método de comunicação. Em contrapartida, algumas pessoas apontam melhorias ao sistema nomeadamente a uma maior personalização do "*pet*" virtual, por exemplo, a definição da cor do mesmo, mais métodos de interação entre o utilizador e o "*pet*" virtual, por exemplo, poder interagir com o "*pet*" virtual de forma mais pessoal, e melhoramento da funcionalidade de recomendação de pontos de interesse perto do utilizador.

6 Conclusão

Este capítulo contém as conclusões e hipóteses resultantes do trabalho desenvolvido, assim como uma abordagem ao trabalho futuro a ser efetuado em torno do projeto de modo a enriquecer o plano e objetivos do trabalho proposto.

6.1 Conclusões Finais

Os resultados obtidos neste trabalho foram bastante satisfatórios tendo em conta os objetivos propostos no início do projeto. Conforme as experiências realizadas, as funcionalidades desenvolvidas durante o trabalho, para além de enriquecerem a aplicação a nível funcional, também enriquecem a aplicação num contexto social. A nível funcional temos a adaptação das recomendações em relação ao seu contexto assim como a sugestão em tempo real de novas tendo em conta o perfil do utilizador e o direcionamento dos mesmos para o local. A nível social temos o enriquecimento da aplicação ao aumentar a atração e foco do utilizador na mesma devido à implementação do "*pet*" virtual.

Embora este trabalho seja um protótipo inicial da aplicação destas funcionalidades, estas revelam grande impacto no produto final, ao permitirem um novo dinamismo na aplicação no que toca a "*interface*" com o utilizador. Porém, é possível admitir que um trabalho futuro deve ser direcionado a estas funcionalidades, de modo a enriquecer, alargar e aperfeiçoar a implementação e renderização do "*pet*" virtual. De acordo com os testes de experimentação realizados com utilizadores reais, conclui-se que é necessário um maior nível de personalização e idealização de métodos de interação entre o "*pet*" virtual e o utilizador.

6.2 Trabalho Futuro

Numa perspetiva de continuação do trabalho implementado, os seguintes pontos devem ser tidos em consideração:

- Atualizar o formato do modelo 3D do "*pet*" virtual para uma estrutura de dados mais recente, visto que o formato .md2 encontra-se desatualizado.
- Melhorar o sistema de renderização para mostrar as cores do "*pet*" virtual.
- Idealizar novas aplicações para a funcionalidade do "*pet*" virtual.
- Melhorar a funcionalidade de recomendação de POI perto do utilizador para não repetir sugestões.

Referências

[1] Masthoff, J.: Group recommender systems: Combining individual models. Recommender systems handbook (2011)

[2] Delic, A., Masthoff, J.: Group Recommender Systems. Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization (2018)

[3] Gavalas, D., Kenteris, M.: A web-based pervasive recommendation system for mobile tourist guides. Personal and Ubiquitous Computing 15 (2011)

[4] Tkalcic, M., Chen, L.: Personality and recommender systems. Recommender systems handbook (2015)

[5] de CA Ziesemer, A., Müller, L., Silveira, M.S.: Just rate it! Gamification as part of recommendation. International Conference on Human-Computer Interaction (2014)

[6] Mortara, M., Catalano, C.E., Bellotti, F., Fiucci, G., Houry-Panchetti, M., Petridis, P.: Learning cultural heritage by serious games. Journal of Cultural Heritage 15 (2014)

[7] "New Product Development Methods" [Online]. Available: https://www.speakingofproducts.com/new-product-development-methods. Access at: 25 jan. 2023

[8] U.N. World Tourism Organization (UNWTO). «Concepts and definitions» (em inglês).

[9] G Wall, A Mathieson: Tourism: change, impacts, and opportunities (2006)

[10] Y Yoon, M Uysal: An examination of the effects of motivation and satisfaction on destination loyalty: a structural model (2005)

[11] Sigala et al.: Big Data and Innovation in Tourism, Travel, and Hospitality (2019)

[12] B Meng, K Choi: The role of authenticity in forming slow tourists' intentions: Developing an extended model of goal-directed behavior (2016)

[13] Fotiadis et al.: Novel coronavirus and tourism: coping, recovery, and regeneration issues (2019)

[14] R Law, ICC Chan, L Wang: A comprehensive review of mobile technology use in hospitality and tourism (2018)

[15] ZX Liang, TK Hui: Residents' quality of life and attitudes toward tourism development in China (2016)

[16] H Chen, I Rahman: Cultural tourism: An analysis of engagement, cultural contact, memorable tourism experience and destination loyalty (2018)

[17] G Adomavicius, A Tuzhilin: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions (2005)

[18] Masthoff, J., Group recommender systems: aggregation, satisfaction and group attributes, in recommender systems handbook. (2015)

[19] J Hamari, J Koivisto, H Sarsa: Does gamification work? -a literature review of empirical studies on gamification (2014)

[20] S Deterding, M Sicart, L Nacke, K O'Hara: Gamification. using game-design elements in non-gaming contexts (2011)

[21] D Buhalis, A Amaranggana: Smart tourism destinations enhancing tourism experience through personalisation of services (2015)

[22] Z Xiang, Q Du, Y Ma, W Fan: A comparative analysis of major online review platforms: Implications for social media analytics in hospitality and tourism (2017)

[23] Albright, J.: Socioeconomic status and the career aspirations of Australian school students: Testing enduring assumptions (2015)

[24] SJ Ahn: Using Avatars and Agents to Promote Real-World Health Behavior Changes (2016)
[25] Pennisi, P., Tonacci, A., Tartarisco, G., Billeci, L., Ruta, L., Gangemi, S., & Pioggia, G.: Autism and social robotics: A systematic review (2016)
[26] B Klimova, P Maresova, K Kuca: Non-pharmacological approaches to the prevention and treatment of Alzheimer's disease with respect to the rising treatment costs (2016)
[27] MJ Kim, CK Lee, MW Preis: The impact of innovation and gratification on authentic experience, subjective well-being, and behavioral intention in tourism virtual reality: The moderating role of technology readiness (2020)
[28] KA Leighty, AJ Valuska, AP Grand, TL Bettinger: Impact of visual context on public perceptions of non-human primate performers (2015)
[29] What is the Value Proposition Canvas? [Online] Available: https://www.b2binternational.com/research/methods/faq/what-is-the-value-proposition-canvas. Access at: 4 fev. 2023
[30] Lapierre, J.: "Customer-perceived value in industrial contexts", Journal of Business & Industrial Marketing (2000)
[31] James F. Petrick, Ph.D.: Development of a Multi-Dimensional Scale for Measuring the Perceived Value of a Service (2017)
[32] Susana Nicola, Eduarda Pinto Ferreira, J. J. Pinto Ferreira.: A Novel framework for modeling value for the customer, an essay on negotiation (2012)
[33] Warwick Manufacturing Group: “Quality Function Deployment,” in Product Excellence using Six Sigma (2006)
[34] S. Nicola: “Análise Valor” (2021)
[35] N. Rich and M. Holweg: “Value Analysis, Value Engineering,” (2000)
[36] T. L. Saaty: Decision making with the analytic hierarchy process (2008)
[37] GrouPlanner [Online] Available: http://www.gecad.isep.ipp.pt/grouplanner/. Access at: 10 fev. 20203
[38] GECAD [Online] Available: https://www.gecad.isep.ipp.pt/GECAD/Pages/Presentation/Home.aspx. Access at: 10 fev. 2023
[39] S Folinas, T Metaxas: Tourism: The great patient of coronavirus COVID-2019 (2020)
[40] M Nilashi, O Ibrahim, E Yadegaridehkordi: Travelers decision making using online review in social network sites: A case on TripAdvisor (2018)
[41] M Sklar, B Shaw, A Hogue: Recommending interesting events in real-time with foursquare check-ins (2012)
[42] H Mehta, P Kanani, P Lande: Google maps (2019)
[43] J Krasnodebski, J Dines: Considering supplier relations and monetization in designing recommendation systems (2016)
[44] B Arslan: AirBnb host recommendation engine (2021)
[45] Omio [Online] Available: https://www.omio.pt/. Access at: 23 fev. 2023
[46] Iceland Naturally: Promoting Iceland in North America [Online] Available: https://fluentresearch.com. Access at: 1 out. 2023
[47] Sightsy.com | A Destination Management Platform [Online] Available: https://sightsy.com. Access at: 23 aug. 2023
[48] ResearchGate [Online] Available: https://www.researchgate.net/figure/The-New-Concept-Development-NCD-model-Koen-et-al-2001_fig12_324208591. Access at: 30 aug. 2023
[49] L Ardissono, A Goy, G Petrone, M Segnan, P Torasso: INTRIGUE: PERSONALIZED RECOMMENDATION OF TOURIST ATTRACTIONS FOR DESKTOP AND HANDSET DEVICES (2003)

[50] M. Batet, A. Moreno, D. Sanchez, D. Isern, A. Valls, Turist@: Agent-based personalized recommendation of tourist activities (2012)
[51] Nguyen, T.N., Ricci, F.: A chat-based group recommender system for tourism. (2018)
[52] Blender [Online] Available: https://www.blender.org/. Access at: 2 set. 2023
[53] Maya: crie mundos expansivos, personagens complexas e efeitos espetaculares [Online] Available: https://www.autodesk.pt/products/maya/overview?term=1-YEAR&tab=subscription. Access at: 5 set. 2023
[54] 3ds Max: crie mundos em grande escala e projetos de alta qualidade [Online] Available: https://www.autodesk.pt/products/3ds-max/overview?term=1-YEAR&tab=subscription. Access at: 6 set. 2023
[55] Cinema 4D [Online] Available: http://cinema4d.com.br/
[56] SketchUp Portugal [Online] Available: https://www.sketchup.ibercad.pt/?gclid=Cj0KCQiA3eGfBhCeARIsACpJNU-mGUO1Rfn8VC4oE4OZFAVuv-_Y2IgCpM5aQMPt-wywJbftZZkP_0EaAoOAEALw_wcB. Access at: 14 set. 2023
[57] Sceneform (1.15.0) [Online] Available: https://developers.google.com/sceneform/
[58] The Asset-Importer-Lib Documentation [Online] Available: https://assimp-docs.readthedocs.io/en/v5.1.0/. Access at: 4 out. 2023
[59] Rajawali [Online] Available: https://github.com/Rajawali/Rajawali. Access at: 4 out. 2023
[60] Announcing "min3D", a 3D framework for Android [Online] Available: http://zeropointnine.com/blog/a-3d-framework-for-android-min3d/. Access at: 4 out. 2023
[61] Alves, P., Martins, A., Novais, P., & Marreiros, G: Improving Group Recommendations using Personality, Dynamic Clustering and Multi-Agent MicroServices. (2023).
[62] Alves, P., Gomes, D., Rodrigues, C., Carneiro, J., Novais, P., & Marreiros, G.: Grouplanner: A Group Recommender System for Tourism with Multi-agent MicroServices. In International Conference on Practical Applications of Agents and Multi-Age (2022, July)
[63] F. Leon: “ActressMAS, a .NET Multi-Agent Framework Inspired by the Actor Model”, Mathematics (2022)
[64] Android Retrofit: Primeiros passos com a Retrofit API Available: https://www.devmedia.com.br/android-retrofit-primeiros-passos-com-a-retrofit-api/31857. Access at: 7 out. 2023
[65] R. Malan, H.-P. Company, and D. Bredemeyer: “Functional Requirements and Use Cases Functional Requirements” (2001)
[66] Sommerville, I: Software Engineering. Addison-Wesley. (2011)
[67] Alves, P., Martins, H., Saraiva, P., Carneiro, J., Novais, P., & Marreiros, G.: Group recommender systems for tourism: how does personality predict preferences for attractions, travel motivations, preferences and concerns? (2023)
[68] García, I. J., Sebastiá, L., Onaindia, E., & Guzman, C.: A multi-agent approach for tourism recommendation. (2009)
[69] SISTEMAS DE RECOMENDAÇÃO: COMO FUNCIONAM E EXEMPLOS PRÁTICOS [Online] Available: https://blog.somostera.com/data-science/sistemas-de-recomendacao. Access at: 9 out. 2023
[70] TAMANHO DO MERCADO DE GAMIFICAÇÃO E ANÁLISE DE PARTICIPAÇÃO NA EUROPA E AMÉRICA LATINA – TENDÊNCIAS E PREVISÕES DE CRESCIMENTO (2023 – 2028) [Online] Available:

https://www.mordorintelligence.com/pt/industry-reports/europe-and-latin-america-gamification-market. Access at: 9 out. 2023

[71] VIRTUAL PETS ARE BACK and Why We Need Them Most Now [Online] Available: https://maekan.com/story/virtual-pets-are-back-and-why-we-need-them-most-now/. Access at: 9 out. 2023

[72] A Brief History of (Virtual) Pets — from Tamagotchi to MetaPals [Online] Available: https://medium.com/@metapals/a-brief-history-of-virtual-pets-from-tamagotchi-to-metapals-d5d4da1b5f28. Access at: 9 out. 2023

[73] MetaPals [Online] Available: https://metapals.pet/ Access at: 9 out. 2023

[74] Martins, A. : Recomendação de Pontos de Interesse para Grupos - Uma Abordagem baseada em Sistemas Multi-Agente e no Feedback dos Turistas (2023)

[75] "Neko Atsume": o que era fofo ficou ainda mais fofo [Online] Available: https://super.abril.com.br/coluna/combo/neko-atsume-o-que-era-fofo-ficou-ainda-mais-fofo. Access at: 11 out. 2023

[76] MetaPals [Online] Available: https://chrome.google.com/webstore/detail/metapals/. Access at: 11 out. 2023

[77] Blender [Online] Available: https://all3dp.com/topic/blender/. Access at: 11 out. 2023

[78] Try Maya free for 30 days [Online] Available: https://www.autodesk.com/products/maya/free-trial. Access at: 11 out. 2023

[79] AutoDesk 3DS MAX [Online] Available: https://camservsolutions.com/3ds-max/. Access at: 11 out. 2023

[80] Crie Animações Incríveis: Aprenda Cinema 4D com a Hospedagem de Vídeo [Online] Available: https://jmvstream.com/pt-br/developer/crie-animacoes-incriveis-aprenda-cinema-4d-com-a-hospedagem-de-video/. Access at: 11 out. 2023

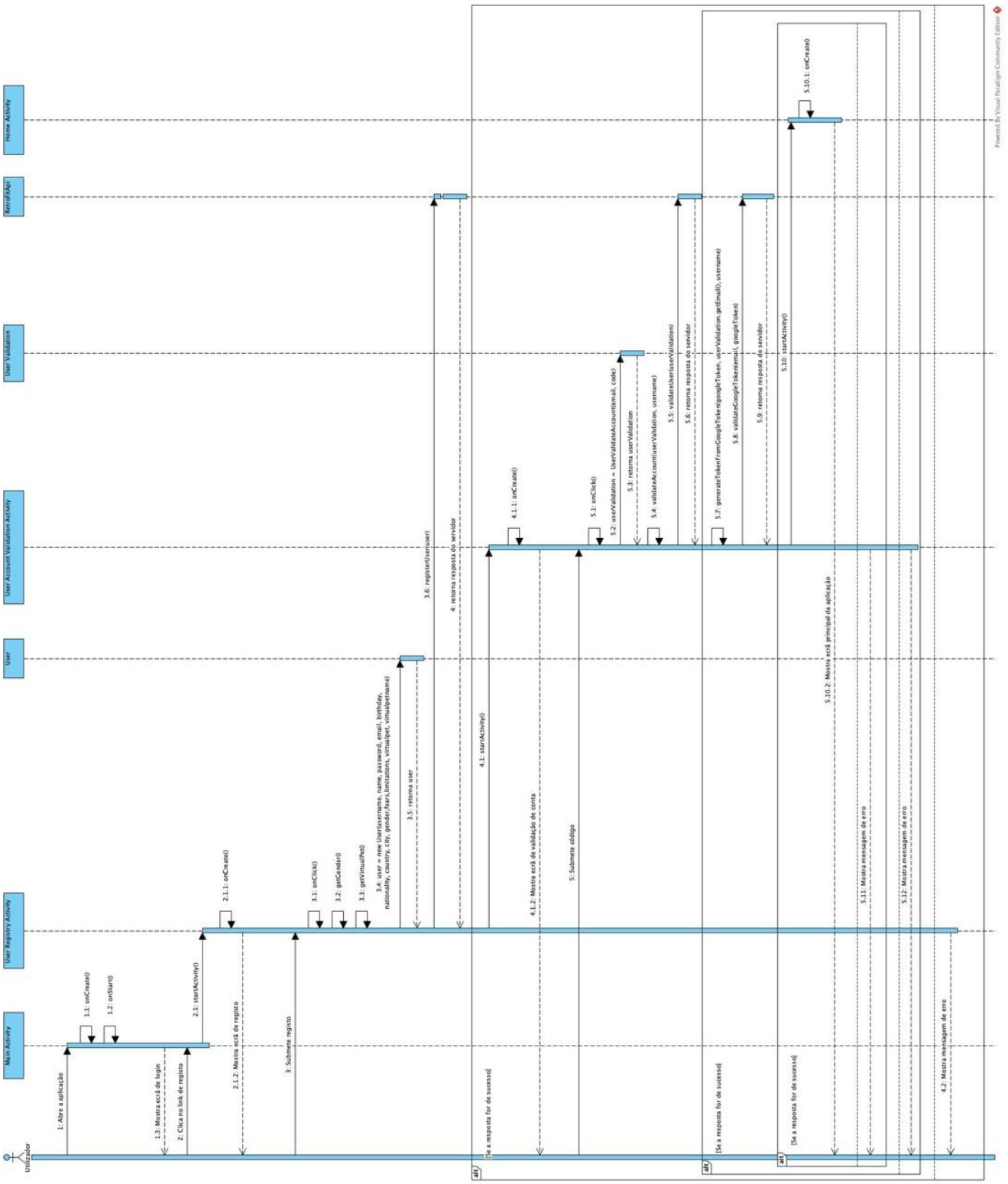
[81] Criação de um plano 2D para o seu modelo de SketchUp no LayOut [Online] Available: https://blog-pt.sketchup.com/pt-blog-articles/cria%C3%A7%C3%A3o-de-um-plano-2d-para-o-seu-modelo-de-sketchup-no-layout-2. Access at: 11 out. 2023

[82] The New Concept Development (NCD)-model [Online] Available: https://www.researchgate.net/figure/The-New-Concept-Development-NCD-model-Koen-et-al-2001_fig12_324208591. Access at: 11 out. 2023

[83] MVC (Model View Controller) Architecture Pattern in Android with Example [Online] Available: https://www.geeksforgeeks.org/mvc-model-view-controller-architecture-pattern-in-android-with-example/. Access at: 11 out. 2023

[84] Android App Architecture ViewModel Concept [Online] Available: https://medium.com/@mohitsharma_49363/android-app-architecture-viewmodel-concept-af04da8822c1. Access at: 11 out. 2023

Anexo A – Diagrama de Sequência do Processo de Criação do "*Pet*" Virtual



Anexo B – Pré-Questionário da Experimentação da Aplicação

Simulação do "Pet" Virtual da Aplicação Grouplanner

Neste estudo, será convidado a experimentar a nova implementação do "Pet" Virtual na aplicação Grouplanner.
O objetivo deste estudo não vai ser revelado de forma a não influenciar os resultados do mesmo.

* Indica uma pergunta obrigatória

Declaração de Confidencialidade e Tratamento dos Dados

Os dados obtidos neste estudo são confidenciais e serão utilizados APENAS de forma agregada e para fins apenas relacionados com o estudo mencionado. Os dados NUNCA serão divulgados de forma individualizada NEM a terceiros. Os dados serão ANONIMIZADOS quando o estudo terminar, com data prevista para 31 de novembro de 2023.

A entidade responsável pelo tratamento e armazenamento dos dados é o GECAD - Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, centro de investigação do ISEP - Instituto Superior de Engenharia do Instituto Politécnico do Porto.

Este estudo foi financiado pelo projeto GrouPlanner e é atualmente financiado pelo projeto smarTravel, sob os Fundo Europeu de Desenvolvimento Regional POCI-01-0145-FEDER-29178 e POCI-01-0247-FEDER-179946, respetivamente, e por fundos nacionais através da FCT (Fundação para a Ciência e a Tecnologia) dentro dos projetos UIDB/00319/2020 e UIDB/00760/2020. Mais informações sobre este projeto em http://www.gecad.isep.ipp.pt/grouplanner

Qualquer questão relacionada com o projeto e/ou questionário deve ser colocada para o seguinte email: 1160921@isep.ipp.pt (Edgar Cordeiro), ou diretora do GECAD: mgt@isep.ipp.pt (Prof. Dra. Goreti Marreiros). Todos os participantes têm o direito de solicitar acesso aos respetivos dados, retificação, cancelamento, limitação e oposição, podendo retirar consentimento em qualquer altura, assim como o direito de apresentar reclamação a uma autoridade de controlo caso considere que houve ilicitude no tratamento, devendo fazê-lo para os contactos anteriormente indicados.

1. Consentimento *

Marcar apenas uma oval.

- ◯ Consinto em participar no questionário e que os dados recolhidos sejam utilizados para os fins acima referidos

Caracterização dos Participantes

As seguintes questões servem para caracterizar demograficamente os participantes antes da
realização da simulação. Responda de forma verdadeira por favor.

2. Nome do Participante *

_____

3. Idade *

Marcar apenas uma oval.

- ◯ Menos de 18 anos
- ◯ Entre 18 e 25 anos
- ◯ Entre 25 e 40 anos
- ◯ Entre 40 e 65 anos
- ◯ Mais de 65 anos

4. Género *

Marcar apenas uma oval.

- ◯ Masculino
- ◯ Feminino
- ◯ Outro

5. Estado Civil *

Marcar apenas uma oval.

- ◯ Solteiro(a)
- ◯ Num relacionamento
- ◯ Divorciado(a)/Separado(a)
- ◯ Viúvo(a)

6. Tem filhos? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

7. Nível Educacional *

Marcar apenas uma oval.

- ◯ Ensino Básico
- ◯ Ensino Secundário
- ◯ Bacharelato
- ◯ Licenciatura
- ◯ Mestrado
- ◯ Doutoramento
- ◯ Outro

8. Área de Formação *

Marcar apenas uma oval.

- ◯ Engenharia e Tecnologia
- ◯ Ciências Exatas
- ◯ Humanidades
- ◯ Ciências Medicas e de Saude
- ◯ Ciências Naturais
- ◯ Ciências Sociais
- ◯ Outra
- ◯ Nenhuma

9. Situação Profissional *

Marcar apenas uma oval.

- ◯ Desempregado(a)
- ◯ Estudante
- ◯ Trabalhador-Estudante
- ◯ Trabalhador(a) por conta de outrem
- ◯ Trabalhador(a) por conta própria
- ◯ Reformado(a)
- ◯ Doméstico(a)
- ◯ Outro

10. Profissão *

11. Quão quanta frequência viaja? *

Marcar apenas uma oval.

- ◯ Nunca
- ◯ Raramente
- ◯ Algumas Vezes
- ◯ Muitas Vezes
- ◯ Constantemente

12. Costuma utilizar aplicações relacionadas com turismo e planeamento de viagens? *

Marcar apenas uma oval.

- ◯ Nunca
- ◯ Raramente
- ◯ Algumas Vezes
- ◯ Muitas Vezes
- ◯ Constantemente

13. Encontra-se familiarizado com o conceito e aplicação de "Pets" virtuais em aplicações? *

Marcar apenas uma oval.

- ◯ Muito Pouco Familiarizado
- ◯ Pouco Familiarizado
- ◯ Familiarizado
- ◯ Bastante Familiarizado
- ◯ Muito Familiarizado

Anexo C - Pós-Questionário da Experimentação da Aplicação

Simulação do "Pet" Virtual da Aplicação Grouplanner

Neste estudo, será convidado a experimentar a nova implementação do "Pet" Virtual na aplicação Grouplanner.
O objetivo deste estudo não vai ser revelado de forma a não influenciar os resultados do mesmo.

* Indica uma pergunta obrigatória

Declaração de Confidencialidade e Tratamento dos Dados

Os dados obtidos neste estudo são confidenciais e serão utilizados APENAS de forma agregada e para fins apenas relaciona- dos com o estudo mencionado. Os dados NUNCA serão divulgados de forma individualizada NEM a terceiros. Os dados serão ANONIMIZADOS quando o estudo terminar, com data prevista para 31 de novembro de 2023.

A entidade responsável pelo tratamento e armazenamento dos dados é o GECAD - Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, centro de investigação do ISEP - Instituto Superior de Engenharia do Instituto Politécnico do Porto.

Este estudo foi financiado pelo projeto GrouPlanner e é atualmente financiado pelo projeto smarTravel, sob os Fundo Europeu de Desenvolvimento Regional POCI-01-0145-FEDER-29178 e POCI-01-0247-FEDER-179946, respetivamente, e por fundos na- cionais através da FCT (Fundação para a Ciência e a Tecnologia) dentro dos projetos UIDB/00319/2020 e UIDB/00760/2020. Mais informações sobre este projeto em http://www.gecad.isep.ipp.pt/grouplanner

Qualquer questão relacionada com o projeto e/ou questionário deve ser colocada para o seguinte email: 1160921@isep.ipp.pt (Edgar Cordeiro), ou diretora do GECAD: mgt@isep.ipp.pt (Prof. Dra. Goreti Marreiros). Todos os participantes têm o direito de solicitar acesso aos respetivos dados, retificação, cancelamento, limitação e oposição, podendo retirar consentimento em qualquer altura, assim como o direito de apresentar reclamação a uma autoridade de controlo caso considere que houve ilicitude no tratamento, devendo fazê-lo para os contactos anteriormente indicados.

1. Consentimento *

Marcar apenas uma oval.

- ◯ Consinto em participar no questionário e que os dados recolhidos sejam utilizados para os fins acima referidos.

Identificação

2. Nome do Participante *

_____

Questões sobre a simulação do "Pet" Virtual

3. Em relação à criação do "pet" virtual, a funcionalidade encontra-se bem implementada *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

4. Em relação à edição do "pet" virtual, a funcionalidade encontra-se bem implementada *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

5. A funcionalidade de recomendações individuais contextuais está bem implementada *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

6. A funcionalidade de recomendações de grupo contextuais está bem implementada *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

7. A funcionalidade de recomendações de POI perto do utilizador está bem implementada *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

8. Existe um alto nível de personalização do "pet" virtual *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

9. Os modelos escolhidos para o "pet" virtual são adequados *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

10. A implementação do "pet" virtual faz me ter mais empatia com a aplicação *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

11. A implementação do "pet" virtual faz me ter mais foco e atração pela aplicação *

Marcar apenas uma oval.

- Discordo Totalmente
- Discordo
- Não Concordo Nem Discordo
- Concordo
- Concordo Totalment

12. A utilização do "pet" virtual para as funcionalidades desenvolvidas é vantajoso *

Marcar apenas uma oval.

- Discordo Totalmente
- Discordo
- Não Concordo Nem Discordo
- Concordo
- Concordo Totalment

13. A utilização do "pet" virtual deve expandir-se para outras funcionalidades *

Marcar apenas uma oval.

- Discordo Totalmente
- Discordo
- Não Concordo Nem Discordo
- Concordo
- Concordo Totalment

14. O nível de interação entre utilizador e "pet" virtual é elevado *

Marcar apenas uma oval.

- ◯ Discordo Totalmente
- ◯ Discordo
- ◯ Não Concordo Nem Discordo
- ◯ Concordo
- ◯ Concordo Totalment

15. Tem sugestões de melhoria? Se sim, quais?