UNIVERSIDADE
LUSÓFONA

SmartQuiz

Redes de Negócios de Colaboração e Marketing
que Motivam com Incentivos e Prémios

Relatório Final

Jolina Guvulo

Professor Fernando Teodósio

Trabalho Final de Curso | LEI | Julho de 2020

www.ulusofona.pt

Conteúdos

Índice de Figuras

Índice de Tabelas

Índice de Exemplos de Código

Resumo

Este trabalho foi realizado no âmbito da avaliação final da Licenciatura em Engenharia Informática (LEI). O tema selecionado, Redes de Negócios de Colaboração e Marketing que Motivam com Incentivos e Prémios, foi desenvolvido na forma de uma biblioteca Python.

A biblioteca pode ser utilizada no desenvolvimento de um negócio online para a criação de questionários e, mais importante, para a seleção do questionário mais relevante para um dado cliente. A biblioteca permite ao programador definir: perguntas, respostas, tópicos, prémios, e estratégias a seguir nos desafios apresentados ao utilizador.

O programador pode utilizar a biblioteca para gerar questionários que abordam temas relacionados com os produto e/ou serviços do site e incentivar a participação do utilizador com prémios. Por exemplo, numa editora, os questionários poderiam perguntar sobre livros, autores, e literatura em geral; e propor descontos, pontos, ou livros, como prémios pela participação ou em função do resultado obtido.

Uma característica distintiva da nossa biblioteca é a forma com é obtida a seguinte pergunta num desafio. A biblioteca seleciona as perguntas de acordo com o perfil do cliente, o seu historial de compras, a resposta dada na pergunta anterior, e a estratégia definida. Exemplos de estratégias podem ser: avaliar produtos comprados anteriormente, explorar outros possíveis temas de interesse ou sugerir novos produtos.

A nossa proposta é relevante porque aborda um tema recorrente em qualquer negócio: obter informação relevante dos interesses dos clientes. No nosso caso, utilizando uma abordagem lúdica na qual também é possível sugerir produto e serviços ao cliente.

A biblioteca foi desenhada de forma genérica, isto é, não está vinculada o um tipo de negócio em particular. Contudo, como forma de testar e demonstrar as funcionalidades da biblioteca, desenvolvemos um site (com funcionalidades básicas) para uma editora.

Palavras-chave: Redes Colaborativas, Marketing, Incentivos, Prémios, Python, Quizzes

Abstract

This work was carried out within the scope of the final evaluation of the Degree in Computer Engineering (LEI). The selected theme, Co-operation and Marketing Business Networks that Motivate with Incentives and Rewards, was developed in the form of a Python library.

The library can be used in the development of an online business for the creation of questionnaires and, more importantly, for the selection of the most relevant questions for a given customer. The library allows the programmer to define questions, answers, topics, rewards, and strategies to be followed in the challenges presented to the user.

The programmer can use the library to generate questionnaires that address topics related to the website's products and services, and to encourage user participation with rewards. For example, in a publishing house, questionnaires could ask about books, authors, and literature in general; and to propose discounts, points, or free books for just the client participation or based on the result obtained.

A distinguishing feature of our library is the way in which the next question is obtained in a challenge. The library selects the questions according to the customer's profile, their purchase history, the answer given in the previous question, and the defined strategy. Example of strategies are evaluating previously purchased products, exploring other possible topics of interest, or suggesting new products.

Form questions are relevant to the user because they cover topics of interest (for example, questions about products previously purchased). On the other hand, questionnaires allow the company to better understand the interests of customers and thus optimize marketing directed to them.

Our proposal is relevant because it addresses a recurring theme in any business: obtaining relevant information from customers' interests. In our case, using a playful approach in which it is also possible to suggest products and services to the customer.

The library was designed in a generic way, that is, it is not associated to a particular type of business. However, as a way to test and demonstrate the library's features, we developed a website (with basic features) for a publishing house.

Keyword: Collaborative Networks, Marketing, Incentives, Rewards, Python, Quizzes

1 Identificação do problema

Qualquer negócio, seja online ou não, precisa de publicitar os seus produtos e serviços. Este projeto pretende contribuir para a otimização do modo como as empresas realizam as suas campanhas de marketing.

As campanhas de marketing utilizadas pelas empresas são desenhadas para atingir diversos objetivos, como: atrair novos clientes, incentivar as compras, ou apenas para dar-se a conhecer no mercado.

As campanhas publicitárias têm custos elevados. Por norma, as empresas contratam estes serviços a empresas especializadas na criação e gestão de campanhas de marketing. Por sua vez, estas empresas recorrem a outros serviços, como por exemplo o Google Ads, como forma de fazer chegar os conteúdos publicitários a uma audiência maior na Internet.

Contudo, alguns pequenos negócios podem optar por gerir o seu marketing utilizando as ferramentas disponibilizadas pelos sites de redes sociais como o Facebook.

Nem sempre os objetivos de marketing são atingidos. Existem muitos fatores que podem conduzir ao fracasso de uma campanha, para este projeto são relevantes:

- a falta de incentivo ao cliente,
- a falta de conhecimento sobre os interesses reais do cliente, ou
- ser uma publicidade pouco atrativa (por exemplo, pouco original).

Incentivos

Uma forma comum de incentivar o cliente numa publicidade de um produto é indicando a possibilidade de um desconto. Existem outras formas, tais como: um programa de pontos, o sorteio de produtos, e a oferta de algum produto.

De facto, os consumidores estão habituados e à espera de encontrar estes incentivos.

Falta de conhecimento sobre os interesses reais do cliente

A falta de informação sobre os clientes, conduz a campanhas mal direcionadas, onde são propostos produtos ou serviços sem interesse para o cliente. No limite, a falta de informação sobre um cliente poderá levar ao fim da relação do cliente com a empresa.

O registo da atividade do cliente e os inquéritos de opinião, aliados com o *data analysis*, permitem aos negócios otimizar as propostas feitas aos clientes.

Uma forma diferente de marketing

A maior parte dos anúncios que o consumidor assiste na Internet têm um formato similar: uma imagem ou uma animação contida numa área reservada para publicidade. Porém, existem muitas outras situações, nomeadamente vídeos (ex. YouTube) ou clips de áudio (ex. Spotify).

Em geral, o formato de publicidade audiovisual é muito efetivo. Contudo, **neste trabalho queremos explorar uma abordagem alternativa: uma publicidade interativa sob a forma de questionários**. Os questionários deverão ser desenhados no sentido de atrair a curiosidade do cliente utilizando prémios como incentivo. Também, estes questionários deverão permitir alcançar diversos objetivos de marketing, como será explicado posteriormente.

Esta forma de publicidade assume um carácter lúdico, sendo interpretada pelo cliente como um desafio a vencer para alcançar um prémio. Para a empresa, será uma oportunidade de construir o perfil do cliente, promover indiretamente os seus produtos, e obter uma avaliação dos seus produtos e serviços, entre outras possibilidades.

A utilização de questionários é comum nos negócios, mas estes questionários frequentemente assumem um carácter formal e as perguntas são diretas. A Figura 1 apresenta alguns exemplos de objetivos que os negócios pretendem alcançar com a realização de questionários.

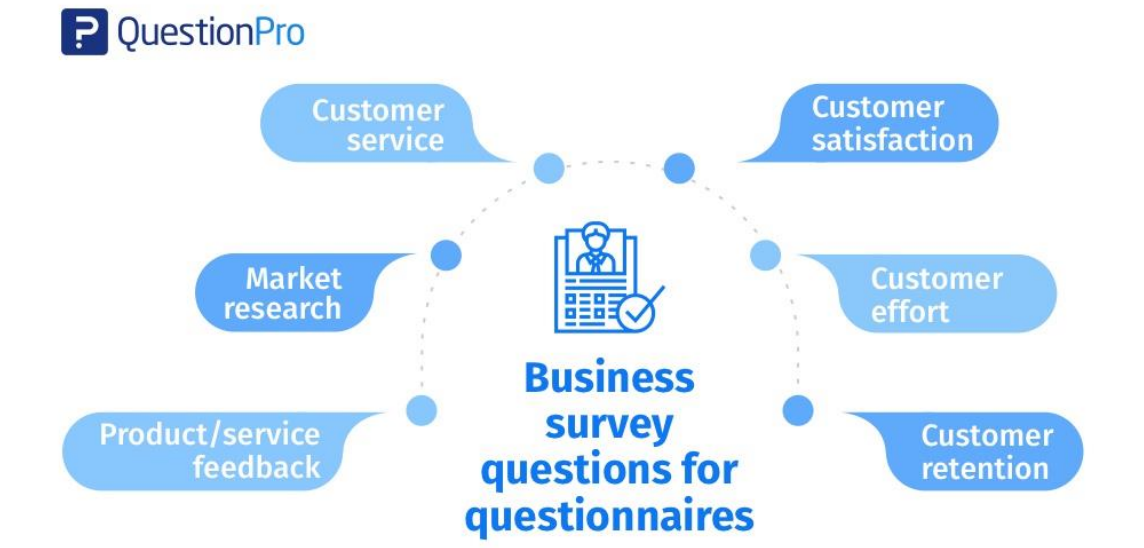

Figura 1. Exemplos de objetivos que os negócios procuram alcançar por meio de questionários. Fonte: www.questionpro.com.

1.1 Caso de estudo: loja online de uma editora

- Formulação detalhada do *case-study* a
- O relatório final deverá expor o âmbito do resultado obtido e realizar analise comparativa destes resultados face ao proposto inicialmente, justificando eventuais diferenças entre proposta e resultados
- Referencias

2 Solução proposta: questionários inteligentes com prémios

Neste Capítulo propomos uma possível solução para os problemas descritos anteriormente. Esta assenta num *framework* para a criação de desafios (questionários estilo trivial) onde o cliente é incentivado a participar tanto pelo tema do desafio como pela existência de um prémio.

Os *quizzes* gerados pelo *framework* poderão ser atrativos visualmente ou divertidos na abordagem dos temas. Contudo, na nossa solução, a relevância e a novidade residem em que as perguntas apresentadas **terão o duplo objetivo de entreter o cliente e atingir objetivos definidos por estratégias de marketing**.

Por exemplo, no contexto do website de uma editora, esta entidade poderia desafiar o cliente com um questionário de cultura geral, de literatura, ou de algum outro tema que seja relevante para o cliente. A relevância do tema pode ser aferida com base em interações anteriores do cliente como, por exemplo, pesquisas que realizou no site do negócio. Também deverá ser proposto um prémio que poderá ter a forma de um desconto, entre outras ofertas.

O sistema que gere os desafios deverá **selecionar as perguntas de forma a atingir vários objetivos de marketing**: desde obter informações sobre os interesses do cliente, perceber que produtos já foram adquiridos pelo cliente, que produtos são desconhecidos pelo cliente, e até sugerir novos produtos, entre outros.

A seguir damos alguns exemplos de perguntas e o objetivo que poderia ser alcançado por parte do nosso caso de estudo, uma editora:

- Perguntas sobre o último livro comprado poderiam permitir indiretamente: obter uma avaliação do cliente sobre esse produto, melhorar o perfil do cliente
- Perguntas sobre outros livros do mesmo autor que foi comprado anteriormente poderiam sugerir indiretamente novos produtos ao cliente e identificar livros que já foram comprados para excluí-los de propostas futuras
- Perguntas menos relacionadas com interações anteriores (por exemplo livros de outro género ou autor) poderiam permitir identificar outros interesses
- Perguntas de cultura geral poderiam criar um perfil de tópicos de interesse do cliente que permitirão futuramente sugerir novos produtos

Uma solução informática que implemente esta abordagem lúdica deverá ser capaz de criar os desafios **sugerindo as perguntas, com base nas respostas anteriores, o historial do cliente e os objetivos de marketing estabelecidos**.

Denominamos os questionários da nossa solução como “inteligentes”, não por testarem os conhecimentos dos clientes, mas sim **por apresentarem uma**

sequência inteligente de perguntas aos clientes: em função das respostas e da estratégia de marketing previamente definida. Dito de outra forma, um desafio não tem uma sequência predefinida de perguntas, o sistema decide a melhor *next question* de forma inteligente.

No próximo Capítulo será feita uma descrição detalhada dos requisitos que uma implementação real da nossa solução deverá satisfazer.

2.1 Exemplo para criar o perfil de um novo utilizador

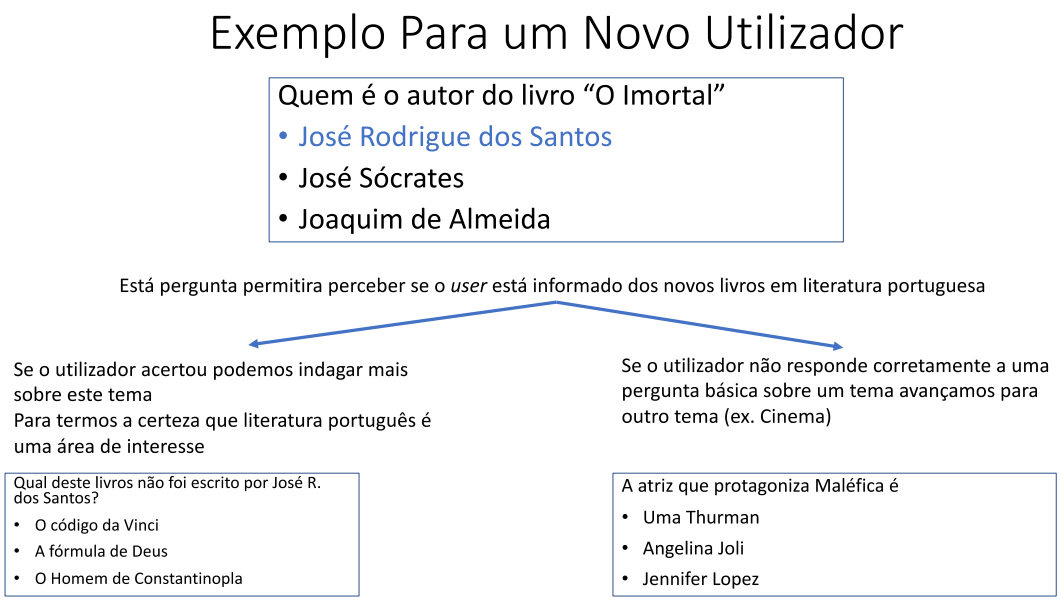

Figura 2. Exemplo de iteração do cliente num questionário onde o objetivo é a identificação interesses. A seguinte pergunta é função da resposta a pergunta anterior.

3 Levantamento e Análise dos Requisitos

Neste capítulo enumeramos os requisitos funcionais e requisitos não funcionais que foram identificados para a solução proposta no Capítulo 2.

Sendo o levantamento superior em âmbito ao desenvolvimento de um Trabalho Final de Curso (TFC), indicamos alguns requisitos que foram implementados parcialmente, mas que ao nosso entender, deveriam fazer parte de uma solução comercial.

Os requisitos apresentados nesta versão final são os mesmos que os apresentados na entrega intermédia, exceto o RF14 que é uma adição. Adicionalmente, melhorámos a descrição de cada requisito. A numeração dos requisitos é a mesma.

3.1 Requisitos funcionais

Requisito	Descrição	Estado
RF01	A biblioteca deverá ser criada de forma genérica e não associada a um tipo de negócio em particular	Feito
RF02	A biblioteca deverá permitir gerir (criar, modificar, eliminar) **perguntas** que podem ser associadas a um ou a vários tópicos	Feito
RF03	A biblioteca deverá permitir gerir **respostas** que podem ser associadas a uma ou a várias perguntas e a um ou vários tópicos	Feito
RF04	A biblioteca deverá permitir gerir **tópicos** que poderão ser associados às perguntas, respostas, clientes, e prémios	Feito
RF05	A biblioteca deverá permitir gerir **prémios** que poderão ser genéricos ou ligados a um dado tópico	Feito
RF06	A biblioteca deverá permitir ao programador criar objetivos/**estratégias** genéricas que serão utilizadas nos desafios (Quizzes)	Feito (ver nota)
RF07	A biblioteca deverá permitir criar **desafios** que seguem uma dada estratégia na geração das sucessivas perguntas	Feito
RF08	Deverá ser possível gerar os desafios (perguntas) em **formato HTML** ou em um formato que permita ao	Feito

	programador criar facilmente o formulário HTML correspondente	
RF09	Os desafios deverão seguir um **algoritmo inteligente** na seleção da pergunta subsequente. O algoritmo terá em conta a estratégia definida, o perfil do cliente e a resposta à pergunta anterior Isto é, os desafios não definem uma sequência fixa de perguntas	Feito (ver nota)
RF10	A biblioteca deverá permitir a ligação a uma base de dados preexistente dos clientes (os *targets* dos questionários)	Feito
RF11	Deverá ser possível **utilizar multimédia** (fotografias, vídeos e áudio) na definição de perguntas, respostas e prémios	Feito
RF12	A biblioteca deverá estar assente numa base de dados que permita conter as classes do sistema	Feito
RF13	Deverá ser utilizada uma linguagem de programação que permita a fácil integração com um projeto web	Feito
RF14	A biblioteca deverá permitir **gerir clientes** no contexto das interações destes clientes com esta solução (não para a gestão geral dos clientes da empresa)	Novo Feito

Tabela 1. Requisitos funcionais da solução.

3.2 Comentário sobre os requisitos funcionais RF06 e RF09

Na definição inicial dos requisitos foi esclarecido que os requisitos funcionais RF06 e RF09 seriam implementados na medida do que é admissível para um TFC. Estes requisitos, numa versão mais elaborada, poderiam, por exemplo, fazer uso de mecanismos de *machine learning*.

Neste TFC os requisitos mencionados foram satisfeitos no sentido de que as classes correspondentes foram implementadas, porém, a classe `Strategy` apenas permite selecionar entre algoritmos fixos embutidos no código e não definir estratégias arbitrárias.

3.3 Novo requisito funcional RF14

Foi adicionado um novo requisito à solução. Este requisito será satisfeito com a existência de tabelas na base de dados que permitirão o registo das interações dos clientes com a solução.

O requisito RF10 deixa assente a existência da entidade Clientes e que a nossa solução não efetua a gestão geral dos mesmos (a empresa terá seguramente a sua forma de gerir os clientes). Contudo, ficou claro durante o desenvolvimento que seria necessário registar na base de dados as interações dos clientes com a nossa solução.

3.4 Requisitos não funcionais

A seguir apresentamos uma tabela enumerando os requisitos não funcionais sugeridos para este projeto.

Requisito	Descrição	Estado
RNF01	Deverá existir uma plataforma que permita testar e demonstrar o funcionamento da solução	Modificado Feito
RNF02	A plataforma do requisito anterior deverá permitir simular diferentes cenários de utilização da solução	Modificado Feito
RNF03	Para assegurar um controlo efetivo da evolução (modificações) do projeto poderá ser utilizado um Sistema de Controlo de Versões	Novo Feito

Tabela 2. Requisitos não funcionais da solução.

3.5 Requisitos não funcionais modificados

A solução proposta é abstrata no sentido de não ser a solução final para um negócio em particular. O nosso projeto é um *framework* (genérico) que permitirá, na sua essência, gerir a base de dados de perguntas e respostas (entre outros elementos) de soluções mais específicas.

Consequentemente, será conveniente contar com um mecanismo que permita demonstrar o funcionamento deste projeto, por exemplo, uma aplicação ilustrativa para o *case-study* utilizado neste TFC.

Para demonstrar a utilização da biblioteca, foi sugerido numa fase inicial dos requisitos não funcionais, criar um website. O website, embora minimalista,

deveria permitir verificar a execução dos requisitos funcionais anteriormente definidos.

Contudo, o projeto paralelo do website demonstrativo foi inexequível por motivos de tempo. Portanto, os requisitos não funcionais foram modificados no sentido de não indicar explicitamente a criação de um website, mas sim a conveniência de existir algum mecanismo para testes e demonstrações.

3.6 Novo requisito não funcional

O novo requisito RNF03 sugere a utilização de um SCV. Este requisito, embora não funcional, deverá ser satisfeito para o projeto ao nível de um TFC.

4 Pertinência e Viabilidade

Neste Capítulo abordamos a pertinência, atualidade, relevância e viabilidade da solução desenvolvida.

4.1 Pertinência e Atualidade

As empresas procuram continuamente **estratégias de marketing diferenciadoras e mais efetivas**, que lhes permitam obter uma vantagem comercial sobre a concorrência. A procura de novas formas de comunicar com os clientes é constante e tem sido condicionada pelas tecnologias disponíveis.

Paralelamente, as empresas procuram **criar um perfil atualizado das preferências do cliente**, por exemplo, para otimizar o marketing direto ao cliente. Nos últimos anos este tema tem sido objeto de um crescente interesse. Cada vez mais as empresas procuram soluções que integrem metodologias provenientes da Ciência de Dados e da Inteligência Artificial.

4.2 Relevância

A solução desenvolvida é relevante porque possibilita dar resposta a estes dois desafios dos negócios: efetivar campanhas de marketing e ao mesmo tempo atualizar o perfil dos clientes. Também, a solução permite integrar metodologias da Ciência de Dados.

Por **permitir executar campanhas de marketin**g a solução contribui diretamente para o aumento das vendas das empresas. Por **contribuir para a construção do perfil dos clientes**, a nossa solução permite reforçar o relacionamento entre a empresa e o cliente e melhorar o marketing direcionado ao mesmo. A **utilização de estratégias inteligentes** permite otimizar todo o processo.

Os questionários gerados pela nossa solução permitirão às empresas, entre outros:

- otimizar o perfil dos clientes,
- identificar produtos que já foram comprados pelos clientes,
- obter uma avaliação de produtos comprados pelo cliente,
- sugerir novos produtos ao cliente
- incentivar o relacionamento com o cliente por meio de prémios

4.3 Flexibilidade

Outro atrativo da solução é a sua flexibilidade. **A solução não está vinculada a um tipo de negócio em particular.** As empresas de marketing ou qualquer

programador em geral podem utilizar a nossa solução como um *framework* para o desenvolvimento de aplicações específicas para diversos tipos de negócios.

Certamente, a solução pode não ser adequada para todos os tipos de negócio. Contudo, a sua utilização estende-se para além do nosso caso de estudo, uma editora, e parece ser facilmente adaptável a qualquer negócio online. Um dos muitos exemplos de outras aplicações, poderia ser o de um clube de futebol, onde os adeptos são desafiados na loja de merchandising com perguntas sobre o seu clube.

4.4 Originalidade

A utilização de questionários onde a empresa incentiva a participação do cliente mediante a atribuição de um prémio, é um mecanismo de marketing comummente utilizado. Contudo, estes questionários são frequentemente apenas um mecanismo de angariação de novos clientes. Por exemplo, terminam solicitando o endereço de email do cliente para resgatar o prémio.

A nossa solução pode ser utilizada para angariar novos clientes e para reforçar o relacionamento com clientes já existentes. Também **acrescenta a capacidade de gerar a sequência de perguntas de forma inteligente**, permitindo otimizar a publicidade apresentada e as informações obtidas.

4.5 Viabilidade

Do exposto nos pontos anteriores, podemos concluir que a solução tem viabilidade de implementação comercial prática em diversos tipos de negócio. Todavia, a solução ainda não foi testada numa situação real.

A solução é viável desde o ponto de vista comercial por vários motivos: atualidade do problema, flexibilidade na implementação e originalidade.

No próximo Capítulo faremos uma introdução das tecnologias utilizadas e principais opções na construção da solução. No Capítulo 7 é feita uma descrição detalhada da mesma.

5 Solução Desenvolvida

Neste Capítulo identificamos as tecnologias utilizadas na elaboração da solução, assim como, algumas opções relevantes tomadas no desenvolvimento da mesma. Explicamos, entre outros, a **opção de desenvolver um *framework* genérico** e também a forma **como são registados os tópicos de interesse**.

5.1 Criação de uma biblioteca genérica

A criação de uma biblioteca genérica em contraposição à criação de uma solução específica (aplicável por exemplo ao nosso estudo de caso), foi uma opção de desenvolvimento tomada para estender a abrangência do projeto.

Com base no estudo de caso que motivou o nosso projeto (uma editora), foi inicialmente sugerido para este TFC a realização de um website com questionários. Os questionários abordariam temas relevantes para o negócio da editora (por exemplo, perguntas de cultura geral). Os prémios poderiam ser descontos em livros ou outros artigos da editora.

A proposta inicial era, portanto, limitada na sua abrangência. Contudo, era mais complexa desde o ponto de vista das tecnologias envolvidas: seria necessário criar o motor para gerir os desafios e também seria necessário criar um website completamente funcional.

Na discussão final do primeiro semestre, **foi solicitada a mudança para uma solução genérica: um *framework* na forma de uma biblioteca de classes**. Esta nova abordagem continua a ser pertinente, acrescenta flexibilidade e, portanto, amplia a viabilidade da solução.

5.2 Git e GitHUB

O projeto foi desenvolvido utilizando o Git[2] como sistema de controlo de versões (SCV). Existem diversos SCV[3], com vantagens e desvantagens sobre o Git. Porém, o Git é o mais popular, em grande medida pela sua utilização no site de hospedagem de código-fonte GitHub. A nossa opção pelo Git foi determinada igualmente pelo facto de ser o SCV que utilizamos durante o curso.

A hospedagem do código-fonte foi feita no site GitHub. O projeto é privado neste momento. Contudo, uma das vantagens do GitHub[4] consiste em ser uma plataforma colaborativa onde milhares (de facto milhões) de programadores colaboram no desenvolvimento de projetos. Futuramente, pretendemos tornar pública uma versão deste projeto.

5.3 Python

O *framework* foi desenvolvido na forma de uma biblioteca para Python. Python é uma linguagem de programação de propósito geral e de alto nível, interpretada e orientada a objetos[5].

A escolha pelo Python não foi arbitrária. Python facilita muito a escrita de código: por ser uma linguagem interpretada, possuir uma comunidade de programadores muito ampla, o que permite obter suporte rapidamente[6], e devido à sua abrangente biblioteca padrão (*batteries included*)[7].

Um dos requisitos funcionais para a solução desenvolvida (RF13) é a utilização de uma tecnologia que permita a fácil integração em projetos web. Neste sentido JavaScript seria a primeira escolha[8]. Contudo, Python é cada vez mais utilizada em projetos web por meio de *web-frameworks* como Django e Flask. Estes *web-frameworks* totalizam em conjunto mais de 25% das preferências entre os programadores inquiridos no site StackOverflow[9].

Em futuras versões da nossa solução será conveniente introduzir metodologias mais elaboradas de análises de dados e inclusive *machine-learning*. Neste sentido, Python é uma das linguagens mais utilizadas na Ciência de Dados com bibliotecas muito populares como NumPy, Pandas; SciPy, Matplotlib, e Scikit-Learn entre muitas outras[10]. Enquanto que o JavaScript ainda não possui bibliotecas tão elaboradas.

5.4 SQLite

O funcionamento da solução requer a utilização de um Sistema de Gestão de Base de Dados (SGBD) (RF12). No nosso caso, a seleção de uma implementação especifica de SQL não é relevante e fazê-lo limitaria a abrangência da solução. Tanto o Python como o Django e o Flask, possuem mecanismos de abstração (Object-relational Mapping) que permitem fazer a gestão de um banco de dados independente da implementação de SQL utilizada[11].

No sentido de testar a solução desenvolvida, foi utilizado o SQLite. O SQLite é uma biblioteca em linguagem C que implementa um SQL, rápido, independente, de alta confiabilidade e com todos os requisitos exigidos a esta linguagem[12].

Ao contrário de muitos outros SGBD, o SQLite não é um mecanismo de banco de dados cliente-servidor. Em vez disso, ele é incorporado no programa final. Desta forma o SQLite facilita o desenvolvimento por não ser necessário instalar um servidor de SQL como aconteceria ao utilizarmos PostgreSQL ou MariaDB. A base de dados é um ficheiro no sistema de arquivos que pode ser facilmente sincronizada pelos programadores que colaboram no projeto se for incluída no SCV.

5.5 Modelo de Classes

Muitos dos requisitos definidos no Capítulo 3 foram implementados na forma de classes da nossa biblioteca. A Figura 3 apresenta o diagrama das classes da solução e relação entre elas.

TODO FIGURA

Figura 3. Diagrama de classe da solução desenvolvida.

5.6 Registo de tópicos: A classe `Label`

Um dos problemas com que nos deparámos durante o desenvolvimento, na fase de desenho do modelo de classes, foi a solução que daríamos ao requisito funcional RF04, "*A biblioteca deverá permitir gerir tópicos que poderão ser associados às perguntas, respostas, clientes, e prémios"*.

Inicialmente, utilizando como referência o nosso estudo de caso, pensámos numa implementação complexa e hierárquica para identificar os tópicos. Na Figura 4, ilustramos o caso de alguns tópicos, definidos hierarquicamente, associados a um dado cliente.

TODO DIAGRAMA

Figura 4. Exemplo de tópicos associados a um cliente e organizados por categorias numa estrutura hierárquica. As formas a vermelho indicam temas nos quais o cliente deverá ter interesse (tópicos onde o cliente demostrou conhecimentos suficientes).

No exemplo da figura anterior, vemos que seria difícil identificar automaticamente o tópico abstrato "romance" como um tema de interesse. São identificadas duas subcategorias diferentes: "músicas românticas" e "comédias românticas". Uma estrutura hierárquica por si só não permite concluir que "músicas românticas" e "comédias românticas" têm uma informação comum, o conceito abstrato de "romance".

Outra limitação do modelo hierárquico consiste na complexidade da sua implementação e utilização prática.

A solução proposta por nós é a utilização de *keywords* que na nossa biblioteca são objetos da classe `Label`. A Figura 5 apresenta como seria registada a mesma informação constante na figura anterior, mas recorrendo a *labels*.

TODO DIAGRAMA

Figura 5.Exemplo de tópicos associados a um clientes utilizando labels (keywords).As formas a vermelho indicam temas nos que o cliente tem manifestado interesse.

A subcategoria "Músicas românticas" é representada por dois objetos `Label`: "Música" e "Romance", a subcategoria "Comédias românticas" poderia ser

representada pelos objetos `Label`: “Cinema”, “Comédia”, “Romance” (e talvez outros como “Hollywood”, “Celebridades”). O asdlkfjaksdjfkasdhfasdf

6 Benchmarking

A utilização de questionários é uma prática comum nos negócios online. Estes questionários são desenhados para alcançar diferentes objetivos, por exemplo: obter feedback de produtos ou serviços, estudos de mercados, entre outros.

Neste Capítulo apresentamos alguns exemplos de como é feita a abordagem deste tema e como poderia ser utilizada a nossa solução para alcançar o mesmo objetivo.

6.1 Formulário no perfil do utilizador

Em muitos websites e aplicações podemos encontrar na área do utilizador uma secção que permite ao cliente indicar interesses. Dita secção é facultativa e muitas vezes nunca é preenchida pelos clientes.

A Figura 6, apresenta o formulário que podemos encontrar na área de cliente do website das lojas Continente. O questionário consiste em apenas um formulário com caixas de seleção. É incluído um conjunto limitado de opções relativas a possíveis interesses do cliente.

Figura 6. Exemplo de formulário na área do cliente onde podem ser definidos interesses. Neste caso para receção de comunicações comerciais.

Utilizando a nossa solução, o mesmo objetivo poderia ser conseguido propondo ao cliente um prémio por completar um desafio de perguntas sobre marcas e produtos. As respostas positivas por parte do cliente podem ou não ser interpretadas como tópicos de interesse atual do cliente. Contudo, as respostas negativas, indiciam de forma mais categórica tipos de produtos que não são do interesse do cliente.

TODO Exemplos de perguntas e respostas.

6.2 Formulários longos e aborrecidos

Uma versão melhorada do formulário simples explicado no ponto anterior, pode ser obtida através do preenchimento de um questionário mais detalhado por parte do cliente. Isto pode acontecer durante a visita ao website da empresa e mais frequentemente mediante o envio de convites por email.

Estes questionários podem ter objetivos diversos: conhecer melhor o cliente, pedir a avaliação de um serviço ou produto, ou fazer um estudo de mercado.

A perceção generalizada sobre este tipo de questionário é que são longos e terminam por aborrecer o cliente. Por vezes existe um incentivo, usualmente um prémio não garantido, por exemplo um sorteio.

A nossa abordagem a este tipo de questionários assenta em primeiro lugar em garantir um prémio. A nossa solução pode então ser utilizada para desenhar um questionário interativo onde por exemplo, após receber uma resposta negativa sobre algum serviço ou produto as perguntas subsequentes irão indagar mais sobre esse assunto.

7 Método e Planeamento

Na Tabela 3, apresentamos as tarefas definidas para a execução do projeto. A concretização deste projeto, nos tempos inicialmente previstos, esteve fortemente condicionada pelo facto do TFC ter sido desenvolvido por um único aluno.

	Duração	Início	Fim
Projecto	**160 dias?**	**30/10/2019 08:00**	**26.06.2020 17:00**
Reunião com Orientador	10,5 dias?	30/10/2019 12:00	30/out
Definição conceptual	1 dia	03/11/2019 08:00	03/11/2019 15:00
Planeamento	2 dias	06/11/2019 08:00	07/11/2019 15:00
Levantamento e Explicação de requisitos	2 dias	08/11/2019 08:00	10/11/2019 15:00
1. Execução	**130 dias ?**	16/11/2019 08:00	19.06.2020 17:00
1.1. Resumo	1 dias	05/11/2019 08:00	06/11/2019 15:00
1.2.Identificação do problema	1 dias	06/11/2019 08:00	07/11/2019 15:00
1.3. Viabilidade	1 dias	07/11/2019 08:00	08/11/2019 15:00
1.4. Pertinência	1 dias	08/11/2019 08:00	09/11/2019 15:00
1.5. Benchmarking	1 dias	09/11/2019 08:00	10/11/2019
1.6. Método e Planeamento	4 dias	10/11/2019 08:00	13/11/2019 15:00
1.7. Resultados	1 dia	13/11/2019	14/11/2019 15:00
1.8. Conclusão	1 dia	14/11/2019 08:00	15/11/2019 15:00
1.9. Reunião com Orientador	1 dia	15/11/2019 08:00	16/11/2019 15:00
2. Controlo	**150 dias?**	30/10/2019 08:00	25/06/2020 17:00
2.1 Testes	1 dia	15/06/2020 08:00	16/06/2020 15:00
2.2. Ajustes	150?	30/10/2019 08:00	25/06/2020 17:00
3. Avaliação 1º Semestre	**42 dias?**	24/11/2019 08:00	26/06/2020 17:00
3.1. Entrega de relatório intercalar	1 dia	24/11/2019 08:00	24/11/2019 17:00
3.2. Desenvolvimeno da biblioteca	127 dias	01/02/2020 14:00	03/07/2020 17:00
3.3. Entrega de relatório intermédio	1 dia	17/01/2020 08:00	17/01/2020 17:00
3.4. Período para avaliações intermédias	1 dia	06/02/2020 08:00	07/02/2020 17:00
4. Avaliação 2º Semestre	**42 dias?**	26/04/2020 08:00	25/06/2020 17:00
4.1. Entrega de relatório intercalar	1 dia	26/04/2020 08:00	10/05/2020 17:00
4.2. Fazer um Website demostrativoem Dja	2 dias	10/07/2020 08:00	13/07/2020 17:00
4.3. Entrega do relatório final	1 dia	17/07/2020 00:00	17/05/2020 00:00

Tabela 3. Tarefas definidas, duração estimada e datas de início e fim.

Nos moldes atuais o projeto do TFC é realizado em dois semestres. Isto permitiu conciliar o desenvolvimento do mesmo com a realização de outros projetos académicos e testes. Contudo, a mudança de objetivos que sugerimos no relatório da Avaliação Intermédia (janeiro de 2020), conduziu a que de facto tivéssemos menos tempo para a concretização do projeto. Mesmo assim, consideramos que o objetivo proposto foi cumprindo.

O projeto não foi apresentado na primeira fase devido a problemas na gestão do tempo dedicado à escrita do relatório. O tempo necessário foi muito superior ao programado inicialmente.

Também foi feito um intento infrutuoso de satisfazer os requisitos não funcionais. Estes requisitos, sugerem a criação de um website demostrativo do funcionamento da biblioteca. Foi criado um site[13] utilizando o *framework* Django. Contudo, não foi possível passar da fase de configuração do *framework* (configuração dos URL's, o template das *Views*, etc.) e da criação da página inicial.

A demostração pode ser feita utilizando um ambiente interativo como o Jupyter Notebook[14]. A maior parte dos exemplos de código apresentados neste documento foram obtidos utilizando Jupyter Notebook.

8 Resultados

Neste capítulo faremos uma descrição detalhada dos resultados tendo em conta os requisitos levantados no Capítulo 3.

Convém lembrar que **o principal *output* deste projeto é o código de uma biblioteca para Python**. Como explicado no Ponto 5.1, este TFC evoluiu de uma solução específica (website + questionários), apenas aplicável a negócios análogos ao do nosso estudo de caso, para uma solução mais abrangente (biblioteca para a gestão de questionários inteligentes).

8.1 Tecnologias utilizadas

A concretização de alguns requisitos funcionais e não funcionais envolveu a utilização das tecnologias enumeradas abaixo. Estas tecnologias assim como as opções de desenho foram abordadas em detalhe no Capítulo 5 (Solução Desenvolvida).

- **Python**[5]: Linguagem de programação utilizada para o desenvolvimento da solução. A utilização de uma linguagem que permita a fácil integração desta solução em projetos web, satisfaz o requisito funcional **RF13**.
- **Jupyter Notebook**[14] **+ iPython Terminal**: Utilizados para testes e demostrações da solução. A existência de um mecanismo que permita demonstrar o funcionamento da solução satisfaz os requisitos não funcionais, **RNF01 e RNF02**.
- **SQLite**[12]: Sistema de Gestão de Bases de Dados utilizado para testes e demonstração da solução. A existência de um SGBD satisfaz o requisito funcional **RF12**. Contudo, a biblioteca pode funcionar com todos os SGBD suportados por Python (ver ponto 5.4).
- **Git**[2] **+ GitHub**[4]: Foi utilizado o Sistema de Controlo de Versões Git em paralelo com a plataforma de alojamento de código-fonte GitHub. A utilização de um SCV é recomendada no requisito não funcional **RNF03**.

8.2 Biblioteca desenvolvida

A opção de desenvolver uma biblioteca que pode ser utilizada como um *framework* foi explicada no Ponto 5.1. Aqui apenas ilustramos a sua utilização.

A biblioteca desenvolvida tem por nome SmartQuiz e pode ser utilizada de acordo com o exemplo de código a seguir.

```
# Fazemos o import das clases que vamos utilizar
from SmartQuiz import SmartQuiz, Label, LabelSet, Answer, Question, User
```

```
# Criamos o objeto principal. O que faz a ligação à base de dados
sq = SmartQuiz("data_base.db")
```

```
Connecting to 'data_base.db'...
Connection succeed
Loading data...
Loading succeed
```

```
# Podemos verifica a conexão utilizando a instancia sq
sq.isConnected()
sq.dbLocation
```

```
True
```

```
'data_base.db'
```

```
# Obtemos por exemplo os Labels. Devolve um objeto LabelSet
ls = sq.labels

# Temos alguns métodos "estaticos" nas classes
# que utilizamos principalmente para validar
LabelSet.isLabelSequence(ls)
```

```
True
```

```
# Também foi implementado o método _str_ (toString) em todas as clases
ls
```

```
{'Geography': <SmartQuiz.Label at 0x1077a3710>,
 'Portugal': <SmartQuiz.Label at 0x1077a3dd0>,
 'Angola': <SmartQuiz.Label at 0x10802cd50>,
 'Spain': <SmartQuiz.Label at 0x10802cf10>,
 'Literature': <SmartQuiz.Label at 0x10802c990>,
 'Romace': <SmartQuiz.Label at 0x10802c790>,
```

Código 1. Exemplo de utilização da classe SmartQuiz, a classe principal da biblioteca. Esta classe faz a ligação à base de dados.

8.3 Classes desenvolvidas

No Ponto 5.5 foi apresentado um diagrama de classes. Abaixo, na Tabela 4, apresentamos as classes criadas e uma descrição sumária da função que cada uma tem na biblioteca.

Classe	Descrição
SmartQuiz	• Cria o objeto principal que faz a ligação a uma base de dados • Permite gerir as classes User, Label, Question, Answer, e Reward e o mapeamento com a base de dados (**RF12**)
Label	Um tópico que pode ser associado a objetos Question, Answer, User, entre outros (**RF04**).
LabelSet	Uma coleção de objetos Label
User	• Um utilizador da base de dados (**RF10 e RF14**) • Pode ter uma lista de objetos Label
Question	• Uma pergunta do banco de dados (**RF02**) • Pode ter uma lista de objetos Label • Pode ter objetos multimédia associados
Answer	• Uma resposta que pode ser associada a várias questões (**RF03**) • Pode ter uma lista de objetos Label • Pode ter objetos multimédia associados
Reward	• Um prémio que pode ser atribuído ao utilizador (**RF05**) • Pode ter uma lista de objetos Label • Pode ter objetos multimédia associados
Strategy	• Define o objetivo de marketing do questionário (**RF06**) • Nesta implementação será apenas para selecionar entre tipos de estratégias predefinidos (sugerir produtos, identificar interesses do cliente, etc.) • Permite indicar Labels para utilizar ou não utilizar nas perguntas • Uma implementação futura deveria permitir ao programador definir estratégias arbitrárias
Quiz	• O questionário aplicado a um User que vai seguir uma dada Strategy (**RF07**) • As perguntas não vão ser predefinidas, o objeto tem um método `next` que permite obter a próxima pergunta em função da resposta anterior e atributos da instância `Strategy`.

Tabela 4. Descrição resumida das classes implementadas na biblioteca.

8.4 Classes `Label` e `LabelSet`

A classe `Label` pretende implementar o requisito funcional RF04: *A biblioteca deverá permitir gerir* ***tópicos*** *que poderão ser associados a perguntas, respostas, clientes e prémios*. Também foi criada a classe `LabelSet` para gerir coleções de `Labels`.

No Ponto 5.6, foi explicada a nossa opção por este mecanismo simples de adicionar informação às classes. Na nossa solução, um `Label` funciona como uma *keyword* que pode representar um tópico, assunto, tema, ou a interpretação que o programador que utilize a nossa biblioteca pretenda atribuir.

Na base de dados (Figura XXX) os `Labels` são texto. Por exemplo: "Geografia", "Politica", "Portugal", "Hollywood".

```
# ls (o LabelSet na base de dados) foi obtido no exemplo anterior
"Portugal" in ls # True
"Music"    in ls # False
```

```
True
```

```
False
```

```
# Criamos um novo Label
l  = Label("  music")

# Alguns properties
l.id == None # None. Ainda não tem um id
l.name       # O texto utilizado: eliminado espaços e capitalizando
```

```
True
```

```
'Music'
```

```
# Ou podemos criar um conjuto de Labels com LabelSet
ll = LabelSet(["Ex1", "Ex2"])

ll.size
print(ll)
```

```
2
```

```
LABELSET: 2 | {'Ex2', 'Ex1'}
```

```
# Adicionamos 'Music' à base de dados
sq.addLabel(l)
```

```
'Music': 1
```

```
# Adicionamos vários labels de exemplo de uma vez
sq.appendLabels(ll)
```

```
'Ex1': 1
'Ex2': 1
```

Código 2. Exemplo de utilização das classes Label e LabelSet. São criados novos *labels* e adicionados à base de dados.

8.5 Classe `Answer`

As respostas são implementadas com a classe `Answer`, esta classe permite definir o texto da resposta e os tópicos que se aplicam à resposta utilizando um `LabelSet`. A seguir apresentamos um exemplo de código com a utilização desta classe.

```
# Obtemos algumas respostas pelo id
aa = sq.getObjects(sq.answers, [1,2,3])

# Vamos imprimir o que foi obtido
for a in aa.values():
    print(a)
```

```
ANSWER: 1 | 'Brad Pit' | {'Actor'}
ANSWER: 2 | 'Fernando Pessoa' | {'Classical', 'Portugal', 'Writer'}
ANSWER: 3 | 'Cristiano Ronaldo' | {'Celebrity', 'Football', 'Portugal'}
```

```
# Vamos criar uma nova resposta
# Utilizamos o mesmo LabelSet do exemplo anterior
a = Answer("Exemplo de resposta", labels=ll)
print(a) # O id é None
```

```
ANSWER: None | 'Exemplo de resposta' | {'Ex2', 'Ex1'}
```

```
# Adicionamos a resposta à base de dados
sq.addAnswer(a)

# O id obtido
a.id
```

```
New answer: OK

12
```

```
# Tentamos adicionar novamente a mesma resposta
sq.addAnswer(a)
```

```
Ocorreu um erro: 'UNIQUE constraint failed: Answers.answer'
```

Código 3. Exemplo de utilização das classes Answer. Foi criada uma nova resposta e esta foi adicionada à base de dados.

8.6 Classe `Question`

As perguntas são implementadas com a classe `Question`, esta classe permite definir a pergunta, os tópicos (`LabelSet`) a que esta pergunta responde, e a sua resposta.

Lembramos que uma mesma resposta, por exemplo, “Fernado Pessoa”, poderia estar associada a várias perguntas.

RF7 e RF10

Na definição inicial dos requisitos foi esclarecido que o RF06 (criação de uma classe que permita a programador definir estratégias **arbitrarias**) não será implementado totalmente numa primeira versão. A classe correspondente foi implementada apenas para indicar o nome de algumas estratégias codificadas diretamente (embutidas) no código da classe Desafio. Por exemplo poderá selecionar-se: “avaliação produto”, “propor produto”, “identificar interesses”; como objetivo a seguir num desafio.

Tendo em conta a complexidade do requisito RF09, que requereria a utilização de *machine leraning*, este requisito será implemento sem analisar estatisticamente o registo de interações dos clientes. Contudo, a nossa implementação sim vai sugerir as perguntas com base na resposta anterior e o objetivo indicado. Também será mantido um registo histórico de interações com o sistema por parte dos utilizadores, para utilização em algoritmo a implementar futuramente.

9 Conclusão

A partir da análise do estudo de caso da loja online de uma editora, propomos a realização de questionários, por exemplo de cultura geral, como uma forma interativa de reforçar a relação do cliente com a empresa e simultaneamente atingir objetivos de marketing. O cliente seria desafiado a testar os seus conhecimentos sendo incentivado com um prémio.

Contudo, a solução final desenvolvida tem uma abrangência superior à do nosso estudo de caso. A solução desenvolvida é um *framework* que permite criar e gerir os desafios com independência do tipo de negócio.

Uma característica distintiva dos questionários gerados com a nossa solução é que a ordem das perguntas não é predeterminada. As perguntas são geradas em função da estratégia de marketing pretendida e das respostas dadas pelo cliente.

9.1 Trabalhos futuros

Para facilitar a promoção desta solução poderia ser criado um website demostrativo. O site deverá permitir aplicar questionários a um utilizador fictício. Também deveria ser possível visualizar a informação que o sistema compilou na forma de "Tópicos de interesse" e "Tópicos de não interesse". Neste sentido, a base de dados de testes também deverá ser reforçada com mais perguntas e respostas.

Na implementação atual, as estratégias que podem ser utilizadas nos desafios podem não ser suficientes para satisfazer as diversas estratégias comerciais e de marketing das empresas. Desta forma, o *framework* deveria evoluir para possibilitar ao programador construir as suas próprias estratégias.

Bibliografia

1. AAA
2. About Git. https://git-scm.com/about.
3. Comparison of version-control software. https://en.wikipedia.org/wiki/Comparison_of_version-control_software.
4. About GitHub. https://github.com/about.
5. Python Programming Language. https://www.python.org/about/.
6. Python community trends. https://opensource.com/article/18/5/numbers-python-community-trends.
7. Python's Batteries Included Philosophy. https://www.python.org/dev/peps/pep-0206/.
8. JavaScript Programming Language. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.
9. Web Framework Popularity. https://insights.stackoverflow.com/survey/2019#technology-_-web-frameworks.
10. Python for Data Science. https://realpython.com/tutorials/data-science/.
11. Object-relational Mappers in Python. https://www.fullstackpython.com/object-relational-mappers-orms.html.
12. About SQLite. https://www.sqlite.org/about.html.
13. SmartQuizz Demo Site. https://smartquiz.pythonanywhere.com/.
14. About Jupyter. https://jupyter.org/about.

Glossário

SCV	Sistema de Controlo de Versões.
SGBD	Sistema de Gestão de Base de Dados.
TFC	Trabalho Final de Curso.

APAGAR

Utilização da biblioteca

A seguir exemplificamos a utilização de biblioteca.

```
# O nome do nosso módulo é SmartQuiz

from SmartQuiz import SmartQuiz, User, Question, Tag, Answser,
Challenge

# Criamos um objeto Quiz com base num ficheiro ou banco de dados
# O banco de dados contém users, perguntas, temas, etc.

sq= SmartQuiz (file ou database)

# Criamos um novo utilizador e o adicionamos à base de dados

u = User(id=3, fields={ "sex":"M", "birthday": "…", …})

sq.addUser(u)

# id seria o id desse user na base de dados principal da
aplicação
# Nos teríamos apenas uma copia com alguns dados dos
utilizadores


# Criamos vários tags

t1=Tag("Literatura")

t2=Tag("Portugal")

t3=Tag("Cinema")

# Os tags não vão ter uma estrutura complexa

# Por exemplo com dependências como Literatura->Portuguesa-
>Romance

# Uma pergunta com essa dependência teria simplesmente esses 3
tags

# Adicionamos os tags à nossa base de dados sq

sq.addTag(t1)

sq.addTag(t2)
```

```
...
# Criamos perguntas
q1 = Question("Quem é o autor do livro 'O Imortal'",
      type="multiple",
      tags=["Literatura", "Português", "Atualidade"]
)
q2 = Question("Miguel de Cervantes é um autor espanhol",
      type="boolean", answer=True,
      tags=["Literatura", "Espanhol", "Clássico"]
)
q3 = Question("De que pais é oriunda a escritora Isabel
Allende",
      type="Multiple",
      tags=[...]
)


# Adicionamos as perguntas à base de dados
sq.addQuestion([q1,q2,q3,...])


# Criamos respostas
# Um mesmo objeto resposta pode ser associado a diferentes
perguntas
a1 = Answer("José Rodrigues dos Santos", tags=["Literatura",
"Português")
a2 = Answer("Angelina Jolie", tags=["Cinema", "Actor",
"Celbrity")
a3 = Answer("José Rodrigues Lopes", ....)
a4 = Answer("Chile", tags="Pais")
...


# Atribuímos respostas à pergunta q1
q1.addAnswer(a1, score=1)   # score =  1 é uma resposta certa
q1.addAnswer(a2, score=-1)  # score = -1 é uma resposta absurda
```

```
                                # denota desconhecimento do tópico
q1.addAnswer(a3, score=0)       # score =  0 é um resposta errada
                                # mas que indica algum conhecimento
do tópico
...


# Assumindo eu o user u1 fez recentemente o registo no site
# O sistema poderá desafia u1
# para obter o máximo de informação sobre seus interesses

# Criamos um desafio

desafio = sq.Challenge(user=1, objective="NewTags")

# os objetivos podem ser diversos: encontrar new tags (tópicos
de interesse)
# apresentar novos produtos para um dado tag


# Cada pergunta depende da anterior
# Por isso a pergunta inicial vai ser None

q = None

while not condições_de_paragem:

     # obtemos a próxima pergunta a fazer (objeto question) dado
pelo desafio

     q = defafio.nexQuestion(previous=q)

     # obtemos uma representação html ou json
     # do formulário que corresponde à pergunta

     form = q.to_html() ou q.to_json()

     enviar_dados_ao_frontend(form)

     id_resposta = esperando_resposta()

     # validamos a resposta
     # deverá também registar esta iteração na base de dado
     # com o registo de interações com este user
     # utilizado futuramente para melhorar a sugestão de
perguntas
     q.check_answer(id_resposta)
```