Gestão Inteligente de Stocks

Ana Santos
Inês Soares
Nuno Veloso

Orientadores Matilde Pato
Nuno Datia

Relatório beta realizado no âmbito de Projeto e Seminário,
do curso de licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2017/2018

Maio de 2018

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores

Gestão Inteligente de Stocks

42142 Ana Rita Ferreira dos Santos
42162 Inês Lima Amil Soares
42181 Nuno Manuel Olival Veloso

Orientadores: Nuno Miguel Soares Datia
Matilde Pós-de-Mina Pato

Relatório beta realizado no âmbito de Projeto e Seminário,
do curso de licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2017/2018

Maio de 2018

Resumo

Palavras-chave:

Abstract

Keywords:

Lista de Figuras

Lista de Tabelas

Índice

Capítulo 1

Introdução

A gestão de stocks é uma tarefa estruturada e repetitiva, para a qual já existem soluções capazes de fornecer listas de compras. *OutOfMilk*[1] e *Bring*[2] são exemplos dessas soluções no formato de aplicações *mobile*. Contudo carecem de controlo de stocks e conhecimento dos hábitos dos seus utilizadores. Como tal, por meio de uma aplicação *mobile* e *web* com suporte inteligente de um algoritmo de previsão de stocks pretende-se solucionar este problema.

Tendo por base a automatização da recolha de dados recorrendo a sensores, simplifica-se, não só, o controlo de stocks, como também, a análise dos padrões de consumo e reposição numa casa. Desta forma, auxilia-se os utilizadores a manter o stock adequado às suas necessidades, bem como alertá-los para a proximidade do fim da validade e/ou stock dos produtos.

Assim, este trabalho vai no sentido de responder a questões como: "De que forma podemos evitar transtornos causados na altura de reabastecer a nossa despensa? Ou como proceder ao controlo de stocks de alimentos e outros produtos? E como impedir artigos fora de prazo?". Se se entender que uma casa funciona como uma empresa e existem quantidades mínimas recomendadas, é possível gerar uma nota de encomenda com os produtos em falta ou prestes a terminar para o utilizador poder consultar e exercer a compra.

1.1 Contexto

Uma boa gestão de stocks de mercadorias é de extrema importância porque tem reflexos imediatos nos resultados de uma empresa, o que permite manter os clientes satisfeitos não só a nível da quantidade como da qualidade. Para manter o stock ideal não basta bom senso e intuição, é necessário conhecer o fluxo de vendas, utilizar ferramentas adequadas de gestão de informação sobre movimentos e eventuais constrangimentos no fornecimento. Extrapolando para a empresa "casa", o processo é apenas um problema de escala. Organizar a despensa como se de uma empresa se tratasse possibilita uma melhor logística de custos e tempo. Ao elaborar uma lista de stock, onde se vai anotando os produtos que se tem, o que está a acabar

[1] https://www.outofmilk.com/
[2] https://www.getbring.com/#!/app

e o que se tem de comprar, passa por uma solução indispensável. Que por vezes se torna numa tarefa que "não é para todos".

Perante este problema, pretende-se desenvolver um sistema, utilizando uma solução digital, aplicação *mobile* e de *web*, que tem como objetivo ajudar os portugueses nesta repetitiva tarefa que é adotar e manter, ao longo do tempo, a sua despensa sem faltas. Através desta solução, o individuo terá sempre presente informação útil e prática, com possibilidade de utilizar um formato de lembretes e de registar as tendências para uma futura investigação no que diz respeito aos hábitos de consumo.

Destaca-se ainda o facto de, no contexto atual, existir um aumento na facilidade de acesso às novas tecnologias, nomeadamente à *internet*. Em plena era da informação a proliferação dos meios de comunicação e da própria *internet* permitiu que os utilizadores se liguem à rede 24 horas, por dia, através de telemóveis, portáteis, *tablets* e outros. A cada dia que passa assiste-se a uma mudança do comportamento do consumidor nesta área, graças à utilização dos dispositivos móveis dos "8 aos 80"anos. Conforme os dados divulgados em Dezembro de 2016 pelo Gabinete de Estatísticas da União Europeia (Eurostat) [1].

1.2 Metas e Objetivos

Face ao exposto, a existência de um sistema de gestão de stocks na agenda de tarefas de uma organização doméstica poderá ser uma mais valia. Para concretizar esse sistema foi necessário cumprir com objetivos mais específicos que respondessem à seguinte questão:

"Quais as características e funcionalidades que deverá ter o sistema que sejam úteis para os utilizadores e se diferencia das restantes?"

Deste modo, definiram-se os seguintes objetivos:

- Implementar uma interface com o utilizador para dispositivos móveis;
- Implementar uma interface com o utilizador para dispositivos *desktop*;
- Implementar a componente servidora de um sistema de gestão de stocks;
- Implementar um algoritmo de previsão de stocks.

1.3 Abordagem do Projeto

Este trabalho divide-se em duas partes principais. A primeira com o enquadramento teórico, em que se fez uma revisão da literatura focando os principais temas associados ao projeto, nomeadamente, gestão de stocks, a utilização das novas tecnologias. Reviu-se também estratégias de usabilidade e promoção de literatura na construção das aplicações móveis bem como a regulamentação existente e possibilidade de certificação. Ainda nesta parte, efetuou-se investigação exploratória de suporte à elaboração do projeto, assim como análise e discussão dos resultados obtidos.

Na segunda parte encontra-se todo o trabalho desenvolvido para o projeto. Inicialmente definiram-se os requisitos fundamentais ao sistema de gestão de stocks a desenvolver. Aqui, foi também importante comparar as funcionalidades que se pretendiam implementar com as de outros sistemas já existentes, de forma a garantir elementos inovadores na solução. Posteriormente, definiu-se a arquitetura do sistema tal como todas as partes envolvidas. Foi ainda necessário estabelecer o modo como o utilizador iria interagir com o sistema, através do desenho das interfaces gráficas.

1.4 Estrutura do Relatório

O relatório está estruturado em 5 capítulos.

O capítulo 2 introduz o sistema de gestão de stocks desenvolvido, como também, formula o problema, detalhando os requisitos do projeto e casos de uso.

No capítulo 3 o problema é solucionado, sendo apresentada a solução implementada. É ainda efetuada uma análise desta.

No capítulo 4 são abordados, em secções, as aplicações de interação direta com o utilizador, o desenvolvimento e implementação do algoritmo de previsão de stocks, a API *Web* bem como todas as suas particularidades e a modelagem dada aos dados. Explica-se de que forma esses dados foram armazenados, sendo ainda justificadas as decisões tomadas.

É no capítulo 5 que se retiram conclusões face ao trabalho desenvolvido em relação ao trabalho inicialmente previsto. Para finalizar, propõe-se o trabalho a realizar futuramente, na secção ???.

No Anexo A define-se terminologia, quer a básica à gestão de stocks, para melhor compreensão de alguns dos termos utilizados no decorrer do projeto, quer de conceitos de programação.

Capítulo 2

Sistema de Gestão de Stocks

O sistema de gestão de stocks desenvolvido neste projeto, denominado Smart Stocks, é apresentado neste capítulo, bem como os requisitos funcionais e opcionais na secção 2.1.1. Para uma melhor compreensão deste capítulo é recomendada a leitura do Anexo A.1.

2.1 Sistema Smart Stocks

Smart Stocks é um sistema que visa dar suporte à gestão de stocks domésticos. Para tal é necessário recolher determinadas informações, tais como, as características da casa a gerir, as particularidades dos membros co-habitantes da casa e ainda os padrões de consumo e reposição. De forma a facilitar tal tarefa são disponibilizadas listas geridas pelo sistema. Por exemplo, lista de compras e lista dos itens em stock na casa, cuja consistência é garantida às custas dos movimentos de entrada e saída dos itens nos diversos locais de armazenamento.

2.1.1 Requisitos

Como forma de assegurar as funcionalidades pretendidas para o sistema, são exigidos os seguintes requisitos funcionais e opcionais.

Requisitos Funcionais

- Informar o utilizador dos itens existentes, a sua validade e a sua quantidade;
- Alertar sobre os produtos que estão perto da data de validade;
- Gerar listas de compras com os produtos em falta;
- Possibilidade de especificar listas com os produtos a ter sempre em stock bem como as suas quantidades mínimas;
- Partilhar listas entre utilizadores da mesma casa;
- Criar Listas;
- Especificar alergias dos membros da casa.

Requisitos Opcionais

- Criar listas de produtos quase a expirar;
- Criar listas de produtos indesejados (Lista Negra);
- Criar listas de contenção em situações de emergência (Lista SOS);
- Inserir refeições extraordinárias de eventos a realizar num futuro próximo, para acrescentar alimentos não básicos à lista de compras.

2.1.2 Entidades

Em seguida identificam-se as diversas entidades relevantes que compõem o sistema de informação, que permite gerir os itens em stock numa dada casa.

Casa

- Cada casa é caracterizada por um identificador único, um nome, atribuído por um utilizador no momento de registo da casa.
- Deve ser possível saber o número de bebés, crianças, adultos e seniores que vivem nessa casa.
- Uma casa está associada a um ou mais utilizadores, podendo um utilizador ter várias casas.
- Podem existir um ou mais administradores de uma casa.
- A casa pode ter vários itens em stock.
- Para cada casa existem vários locais de armazenamento dos itens, por exemplo armários, frigoríficos, etc.
- Em cada casa deve ser possível conhecer as alergias assim como quantos membros possuem essa alergia (os membros não precisam necessariamente de estar registados).

Utilizador

- Uma pessoa é representada por um utilizador que é caracterizado por um email ou por um nome de utilizador, pelo nome da pessoa, a sua idade e uma *password*.

Listas

- Cada lista é composta por um identificador único e um nome.
- Uma lista pode ter vários produtos.

- Existem dois tipos de listas: de sistema e de utilizador.
- As listas de sistema são comuns a todos os utilizadores registados, contudo são particulares a cada casa.
- Um utilizador pode criar as suas listas, partilhando-as com outros utilizadores da casa ou tornando-as secretas.

Categoria

- Uma categoria é identificada univocamente por um número ou por um nome.

Produtos

- Um produto é constituído por um identificador único, um nome, se é ou não comestível, e a validade perecível.
- Para os produtos presentes numa lista pode ser possível saber a sua marca e a quantidade.
- Um produto pertence a uma categoria, podendo uma categoria ter vários produtos.
- Um produto pode ser concretizado por diversos itens em stock na casa.

Item em Stock

- Um item em stock é a concretização de um produto que existe numa casa. É identificado univocamente por um número ou por uma marca, uma variedade e um segmento, é também caracterizado por uma descrição, o local de conservação, a quantidade e as datas de validade.
- Para cada item deve ser possível saber os seus movimentos de entrada e saída de um local de armazenamento.
- Deve também ser possível saber os alergénios de cada item presente na casa.

Movimento

- Para cada movimento deve ser possível saber o tipo do movimento (entrada ou saída), a data em que ocorreu e a quantidade de produtos.

Local de armazenamento

- Cada local de armazenamento é caraterizado por um identificador único, a temperatura e um nome.
- Deve ser possível saber a quantidade de cada item presente no local.

- Um local de armazenamento pode ter vários itens em stock presentes na casa e estar associado a diversos movimentos.

2.2 Arquitetura da Solução

Nesta secção pretende-se abordar de forma geral a solução implementada para resolver o problema apresentado na secção 2.1.

2.2.1 Abordagem

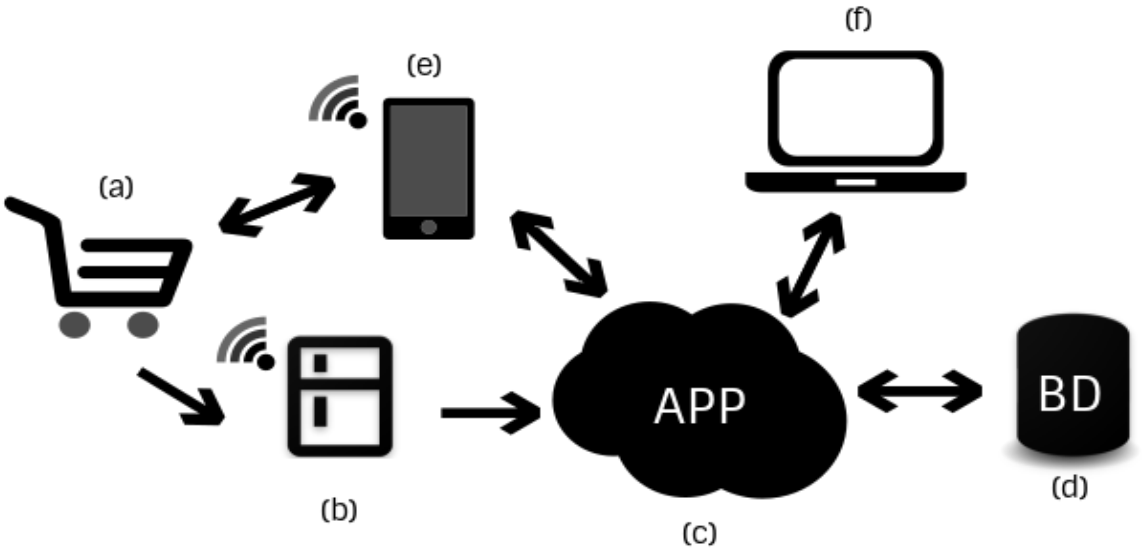

Figura 2.1: Arquitetura Geral do Projeto

Após uma ida às compras, os itens adquiridos, Figura 2.1(a), são armazenados nos seus respetivos locais, Figura 2.1(b). Como forma de automatizar a recolha de informação relativa quer aos artigos obtidos quer às suas caraterísticas, utilizam-se sensores. O uso destes só é possível caso os rótulos dos itens se encontrem em formato digital *standard*, com *tags Near-Field Communication* (NFC) [2] ou *Radio-frequency Identification* (RFID) [3], e os locais de armazenamento contenham os respetivos leitores de *tags*.

Ao guardar os artigos nos locais de armazenamento, os mesmos devem ser lidos pelos leitores, de forma a que a informação presente na *tag* e o tipo de movimento (entrada ou saída) possam ser enviados para a API. Assim, estes dados são posteriormente tratados e armazenados de forma persistente na Base de Dados (BD), Figura 2.1(d). A APP, Figura 2.1(c), é responsável por retornar dados para as aplicações cliente, Figura 2.1(e, f). É ainda nesta que está presente o algoritmo de previsão de stocks utilizado para efetuar a previsão quanto à duração de cada um dos itens em stock.

No contexto da gestão de stocks assume-se a existência de duas formas de apresentação para os itens em stock: avulsos e embalados. Os primeiros são conservados em sistemas de arrumação identificados com *tags* programáveis por *smartphones*, 2.1(e). Os detalhes dos itens são especificados pelo utilizador e carregados para a *tag*. Já os segundos contêm os seus rótulos digitais com o detalhe guardado pelos embaladores.

2.2.2 Arquitetura por Camadas

O sistema de gestão de stocks é composto por 2 blocos principais: o bloco do lado do cliente e o bloco do lado do servidor, que se relacionam. A representação destes blocos é apresentada na Figura 2.2.

A arquitetura do projeto segue o padrão de arquitetura por camadas, dado que este padrão permite individualizar cada camada. Com isto, estas tornam-se independentes umas das outras, fornecendo não só abstração sobre as camadas inferiores, mas também, oferecendo a possibilidade de testar ou substituir cada uma das mesmas, sem que existam alterações significativas nas restantes.

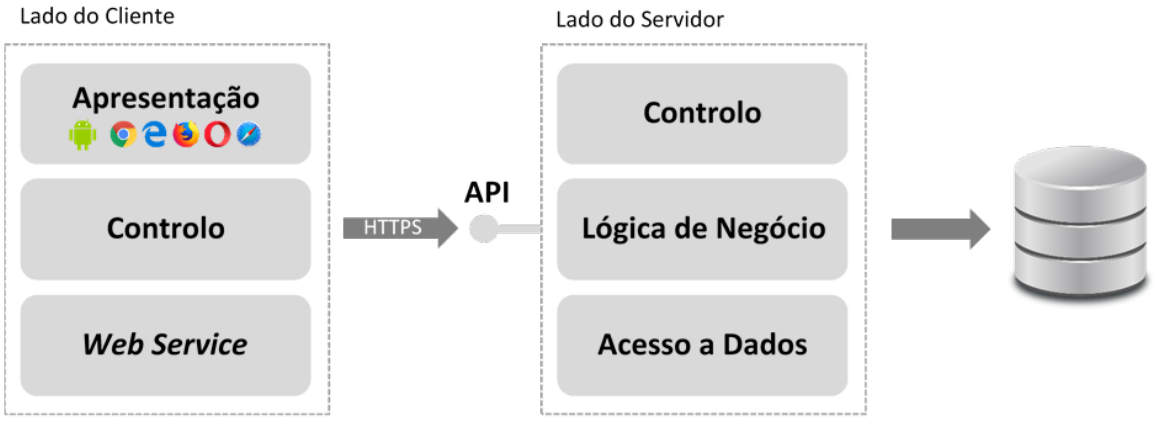

Figura 2.2: Arquitetura por Camadas do Projeto

No lado do cliente têm-se três camadas: a camada Apresentação que é responsável por representar os dados solicitados pelo utilizador; o Controlo que está encarregue de despoletar ações na camada do *Web Service* de forma a satisfazer as solicitações do utilizador; e assim o *Web Service* interage com a API *Web*.

As camadas que compõem o lado do servidor são: o Controlo que processa pedidos e retorna uma resposta; a camada da Lógica de Negócio que é responsável por satisfazer as regras de negócio; e por fim o Acesso a Dados que efetua leituras e escritas sobre a BD.

Tecnologias Inerentes à Solução

O lado do servidor incluí três camadas e expõe uma API *Web*. A Camada de Acesso a Dados (DAL) é produzida com a linguagem de programação *Java*, usando a *Java Persistent API* (JPA), e é responsável pelas leituras e escritas exercidas sobre a Base de Dados (BD). A BD é externa ao servidor, utilizando para isso o Sistema de Gestão de Base de Dados (SGBD) *PostgreSQL*. A Camada da Lógica de Negócio (BLL) é responsável pela aplicação das regras de negócio. A implementação desta camada é também realizada com linguagem *Java*. Os *controllers* foram desenvolvidos em *Java* com a *framework* da *Spring*, chamada de *Spring Boot*. A API *Web* disponibiliza recursos em diferentes *hypermedias*. Para a implementação do algoritmo de previsão de stocks usou-se a linguagem *R*.

Do lado do cliente existem dois modos de interação, por uma aplicação móvel e outra por uma aplicação web. A aplicação móvel disponível para a plataforma *Android*, desenvolvida em linguagem *Kotlin*. A aplicação web é disponibilizada para a maioria dos browsers, e é implementada utilizando a linguagem *JavaScript* no ambiente *Node.js*, com o auxilio da *framework Express*.

Capítulo 3

Implementação do Sistema de Gestão de Stocks

O presente capítulo retrata a implementação do sistema de gestão de stocks, Smart Stocks, que se subdivide nas seguintes secções: 3.1) Aplicações Móvel e Web, 3.2) Web API, 3.3) Algoritmo de Previsão de Stocks, 3.4) Modelo de Dados.

3.1 Aplicações Cliente

Tabela 3.1: Comparação com Duas Aplicações

	Smart Stocks	**OutOfMilk**	**Bring!**
Aplicação Móvel	✓	✓	✓
Aplicação Web	✓	✓	✓
Listas de Compras	✓	✓	✓
Listas de Tarefas	✗	✓	✗
Partilha de Listas	✓	✓	✓
Adicionar Itens às Listas	✓	✓	✓
Adicionar Detalhes aos Itens	✗	✓	✗
Produtos Organizados por Categorias	✓	✓	✓
Integração com Dispositivos Inteligentes	✓	✗	✗
Sincronização em Tempo Real	✓	✓	✓
Gestão Automática da Despensa	✓	✗	✗
Gestão de Múltiplas Casas por Utilizador	✓	✗	✗
Leitura de Rótulos Digitais	✓	✗	✗
Previsão de Stocks	✓	✗	✗

3.2 Servidor

A API *Web* é a interface exposta pela aplicação baseado em *Hypertext Transfer Protocol* (HTTP) [4]. Esta disponibiliza informação e funcionalidades fornecidas pelo sistema Smart Stocks, através de *endpoints* públicos.

A decisão de utilizar a *framework Spring Boot* para implementar o servidor deve-se ao facto de ser uma ferramenta *open source*, capaz de criar rapidamente aplicações com auto-configuração, através de anotações. Outra vantagem é a integração com tecnologias de persistência de dados, como a *Java Persistent API* (JPA). Por fim, por questões de conhecimento e de experiência anterior com esta *framework*.

O servidor é composto por várias camadas - Acesso a Dados, Lógica de Negócio e Controlo.

3.2.1 Acesso a Dados

Uma vez armazenados os dados de forma persistente é indispensável realizar escritas e leituras sobre os mesmos. Como forma de abstrair as restantes camadas do SGBD, é necessário definir interfaces entre as diferentes camadas. Para tal, desenvolveu-se a Camada de Acesso a Dados (DAL).

Como o modelo de dados é relativamente extenso o uso de *Java Persistent API* (JPA) com *Hibernate* torna-se benéfico uma vez que permite reduzir a repetição de código envolvido para suportar as operações básicas de *Create, Read, Update e Delete* (CRUD) em todas as entidades.

No acesso a dados, são utilizados dois padrões de desenho: Padrão *Repository* [5] e Padrão *Unit Of Work* [6]. A DAL é, salvo exceções, gerada através da JPA com *Hibernate*.

O principal requisito é o acesso aos dados da BD e o suporte para as operações CRUD nas tabelas. Desta forma criou-se interfaces *repositories* com métodos que garantem não só essas operações, como outras para facilitar a obtenção de dados de determinada forma. O uso de JPA com *Hibernate* obriga a representar o modelo da BD em classes *Java*, *Plain Old Java Objects* (POJO).

Cada entidade presente na BD é mapeada numa classe em Java, que representa o modelo de domínio da mesma. Esta classe tem várias anotações da JPA com *Hibernate* para referir a Chave-Primária, Chave-Estrangeira, associações entre entidades, etc. Em conjunto estas classes Java formam o modelo utilizado entre as camadas internas do lado do servidor.

3.2.2 Lógica de Negócio

É fundamental fazer cumprir as regras, restrições e toda a lógica da gestão dos dados para o correto funcionamento do sistema. Assim este controlo foi depositado na Camada da Lógica de Negócio (BLL) e também no modelo desenvolvido. Esta decisão permite não só concentrar a gestão dos dados como também controlar numa camada intermédia os dados a obter, atualizar, remover ou inserir, antes de realizar o acesso/escrita dos mesmos.

que estabelece um conjunto de operações disponíveis e coordena a resposta do aplicativo em cada operação.

Cada serviço estabelece um conjunto de operações disponíveis [7]. Desta maneira, consegue-se isolar as diversas operações referentes a cada entidade. Assim, para a implementação desta

camada criaram-se serviços para cada entidade. Estes expõem funcionalidades e aplicam as regras de negócio necessárias. É de salientar que um serviço está fortemente ligado a um ou mais *repositories*, e estes podem estar associados a um ou mais serviços.

3.2.3 Controlo

Os Controlos são responsáveis por processar os pedidos aos diferentes *endpoints* e produzir uma resposta. Os formatos de resposta utilizam *hypermedia* [8], são de um de três tipos, *Json Home* [9], *Siren* [10] e *Problem Details* [11]. A escolha do uso de *hypermedia* apoia-se em questões evolutivas da API em termos de hiperligações, ou seja, caso os *endpoints* dos recursos sejam alterados a aplicação cliente não sofre alterações.

3.2.4 Segurança

Sendo o Smart Stocks um sistema de gestão de stocks domésticos é de extrema importância assegurar a confidencialidade e segurança dos dados de cada casa. Como tal, o acesso a recursos ou a manipulação dos mesmos só pode suceder de forma autenticada e autorizada. Ao assumir a utilização do protocolo *Hypertext Transfer Protocol Secure* (HTTPS) [12], nesta primeira fase, permitiu escolher como forma de autenticação o *Basic Scheme* [13] .

3.3 Algoritmo de Previsão de Stocks

Uma vez que para uma boa gestão de stocks existe a necessidade de prever a duração de cada um dos itens em stock, com base nos historial de consumo e reposição da casa. De forma a garantir este requisito inerente ao sistema Smart Stocks, introduziu-se um algoritmo de previsão de stocks.

Para a realização deste algoritmo, teve-se a preocupação de analisar os vários tipos de modelo existentes e adotar o mais adequado ao sistema. Durante a pesquisa, deparou-se com a existência de dois tipos de modelo de previsão, sendo eles, Modelos Quantitativos e Qualitativos. Os primeiros baseiam-se em análises numéricas dos dados históricos, enquanto que os qualitativos privilegiam dados subjetivos baseados na opinião de especialistas. Como se deseja realizar este algoritmo com base no histórico de consumo e reposição, optou-se pelo Modelo Quantitativo.

Dos sub-modelos disponíveis elegeu-se um modelo de séries temporais visto que este pode incluir nos seus dados um conjunto de elementos, por exemplo, sazonalidade, tendências, influências cíclicas ou comportamento aleatório.

3.3.1 Implementação

Aplicou-se o método da média móvel ponderada para um período mínimo de 3 semanas. Este procedimento começa por atribuir pesos aos dados consoante a semana a que se referem,

sendo que as mais recentes têm maior peso.

Como se tratam de dados diários e para evitar a descompensação de valores provocada por dias com consumo excecionais amortecem-se os valores. Assim os resultados tornam-se mais homogéneos e com um comportamento mais sazonal.

Com base nestes dois aspetos construiu-se um dia típico, que servirá de base para uma previsão. Seguiu-se o seguinte procedimento:

1º Passo

A primeira fase consiste em aplicar o método da média móvel ponderada aos dados, para isso usou-se a seguinte expressão:

$$Pdiax' = T1 \times Rdia1 + T2 \times Rdia2 + T3 \times Rdia3$$

Onde os valores de Tx correspondem às taxas definidas como pesos para os dados diários ($Rdiax$) da semana x. As taxas escolhidas foram:

$$T1 = 10\%,\ T2 = 30\%\ e\ T3 = 60\%$$

2º Passo

Aos valores obtidos anteriormente aplicou-se um método de amortização de forma a homogeneizar e harmonizar a previsão. Assim, o valor obtido para um dia da semana resulta da soma do seu valor e do valor do dia da semana anterior e da seguinte, multiplicados por taxas, ou seja:

$$Pdiax'' = Tant \times PdiaxAnt' + Tdiax \times Pdiax' + Tseg \times PdiaxSeg'$$

Onde os valores de $Tant$, $Tdiax$ e $Tseg$ correspondem às taxas definidas como pesos para os dias da semana anterior, atual e seguinte, respetivamente. Os valores $PdiaxAnt'$, $Pdiax'$ e $PdiaxSeg'$ são os valores previstos no 1º passo do método de previsão para os dias da semana anterior, atual e seguinte, respetivamente. As taxas escolhidas foram:

$$Tant = Tseg = 25\%\ e\ Tdiax = 50\%.$$

3.4 Modelo de Dados

3.4.1 Modelo Entidade-Associação

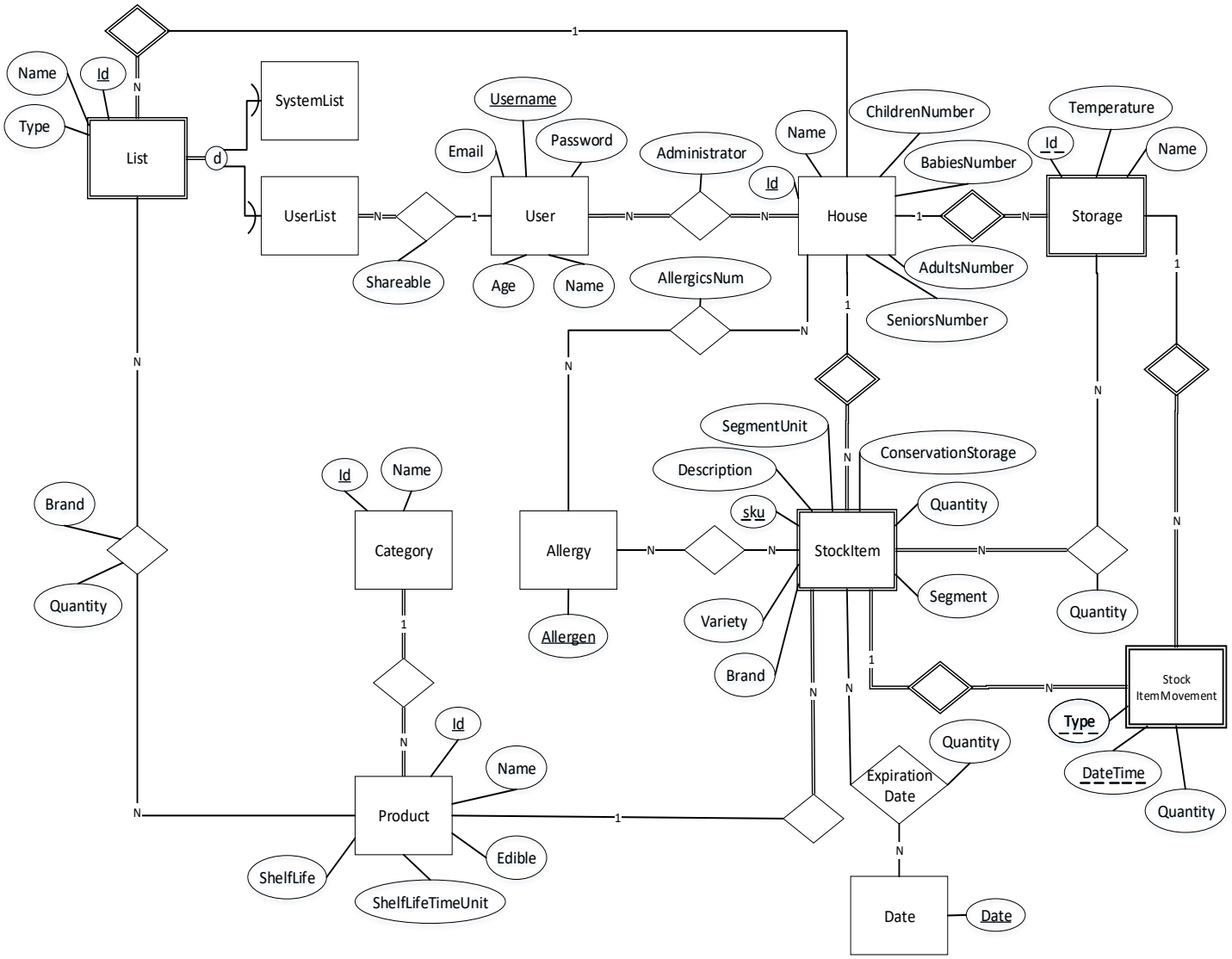

Figura 3.1: Modelo Entidade-Associação

3.4.2 Domínio dos Atributos

Tabela 3.2: Domínio dos Atributos da Entidade House.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
House	house_id	Número inteiro auto-incrementado	bigserial	-	não
	house_name	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não
	house_characteristics	Objeto JSON	json	-	não

Tabela 3.3: Domínio dos Atributos da Entidade User.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
User	user_username	Cadeia de caracteres de comprimento variável	character varying(30)	até 30 caracteres	não
	user_email	Cadeia de caracteres de comprimento variável	character varying(254)	até 254 caracteres	não
	user_age	Número inteiro	smallint	user_age in [0, 150]	não
	user_name	Cadeia de caracteres de comprimento variável	character varying(70)	até 70 caracteres	não
	user_password	Cadeia de caracteres de comprimento variável	character varying(50)	até 50 caracteres	não

Tabela 3.4: Domínio dos Atributos da Entidade Allergy.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
Allergy	allergy_allergen	Cadeia de caracteres de comprimento variável	character varying(75)	até 75 caracteres	não

Tabela 3.5: Domínio dos Atributos da Entidade List.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
List	house_id	Número inteiro	bigint	house_id > 0	não
	list_id	Número inteiro auto-incrementado	smallint	-	não
	list_name	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não
	list_type	Cadeia de caracteres de comprimento variável	character varying(7)	list_type in ['system', 'user']	não

Tabela 3.6: Domínio dos Atributos da Entidade SystemList.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
System List	house_id	Número inteiro	bigint	house_id > 0	não
	list_id	Número inteiro	smallint	list_id > 0	não

Tabela 3.7: Domínio dos Atributos da Entidade UserList.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
User List	house_id	Número inteiro	bigint	house_id > 0	não
	list_id	Número inteiro	smallint	list_id > 0	não
	user_username	Cadeia de caracteres de comprimento variável	character varying(30)	até 30 caracteres	não
	list_shareable	Booleano	boolean	-	sim

Tabela 3.8: Domínio dos Atributos da Entidade Category.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
Category	category_id	Número inteiro auto-incrementado	serial	-	não
	category_name	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não

Tabela 3.9: Domínio dos Atributos da Entidade Product.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
Product	category_id	Número inteiro	integer	category_id > 0	não
	product_id	Número inteiro auto-incrementado	serial	-	não
	product_name	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não
	product_edible	Booleano	boolean	-	não
	product_shelfLife	Número inteiro	smallint	product_shelfLife > 0	não
	product_shelfLifeTimeUnit	Cadeia de caracteres de comprimento variável	character varying(5)	product_shelfLifeTimeUnit in ['day', 'week', 'month', 'year']	não

Tabela 3.10: Domínio dos Atributos da Entidade StockItem.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
StockItem	house_id	Número inteiro	bigint	house_id > 0	não
	stockItem_sku	Cadeia de caracteres de comprimento variável	character varying(128)	até 128 caracteres	não
	category_id	Número inteiro	integer	category_id > 0	não
	product_id	Número inteiro	integer	product_id > 0	não
	stockItem_brand	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não
	stockItem_segment	Número décimal	real	stockItem_segment > 0	não
	stockItem_variety	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não
	stockItem_quantity	Número inteiro	smallint	stockItem_quantity > 0	não
	stockItem_segmentUnit	Cadeia de caracteres de comprimento variável	character varying(5)	stockItem_segmentUnit in ['kg', 'dag', 'hg', 'g', 'dg', 'cg', 'mg', 'kl', 'hl', 'dal', 'l', 'dl', 'cl', 'ml', 'oz', 'lb', 'pt', 'fl oz', 'units']	não
	stockItem_description	Cadeia de caracteres de comprimento variável	text	-	sim
	stockItem_conservationStorage	Cadeia de caracteres de comprimento variável	character varying(128)	até 128 caracteres	não

Tabela 3.11: Domínio dos Atributos da Entidade Ingredient.

Entidade	**Atributo**	**Domínio**	**Tipo Variável (PostgreSQL)**	**Restrições**	**Nullable**
Ingredient	recipe_id	Número inteiro	integer	recipe_id > 0	não
	category_id	Número inteiro	integer	category_id > 0	não
	product_id	Número inteiro	integer	product_id > 0	não
	ingredient_quantity	Número inteiro	integer	ingredient_quantity > 0	não
	ingredient_quantityUnit	Cadeia de caracteres de comprimento variável	character varying(5)	ingredient_quantityUnit in ['kg', 'dag', 'hg', 'g', 'dg', 'cg', 'mg', 'kl', 'hl', 'dal', 'l', 'dl', 'cl', 'ml', 'oz', 'lb', 'pt', 'fl oz', 'units']	não

Tabela 3.12: Domínio dos Atributos da Entidade Storage.

Entidade	**Atributo**	**Domínio**	**Tipo Variável (PostgreSQL)**	**Restrições**	**Nullable**
Storage	house_id	Número inteiro	bigint	house_id > 0	não
	storage_id	Número inteiro auto-incrementado	smallint	-	não
	storage_name	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não
	storage_temperature	Intervalo de números decimais	numrange	-	não

Tabela 3.13: Domínio dos Atributos da Entidade UserHouse.

Entidade	**Atributo**	**Domínio**	**Tipo Variável (PostgreSQL)**	**Restrições**	**Nullable**
UserHouse	house_id	Número inteiro	bigint	house_id > 0	não
	user_username	Cadeia de caracteres de comprimento variável	character varying(30)	até 30 caracteres	não
	userHouse_administrator	Booleano	boolean	-	sim

Tabela 3.14: Domínio dos Atributos da Entidade StockItemStorage.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
StockItemStorage	house_id	Número inteiro	bigint	house_id > 0	não
	stockItem_sku	Cadeia de caracteres de comprimento variável	character varying(128)	até 128 caracteres	não
	storage_id	Número inteiro	smallint	storage_id > 0	não
	stockItemStorage_quantity	Número inteiro	smallint	stockItemStorage_quantity > 0	não

Tabela 3.15: Domínio dos Atributos da Entidade StockItemMovement.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
StockItemMovement	house_id	Número inteiro	bigint	house_id > 0	não
	stockItem_sku	Cadeia de caracteres de comprimento variável	character varying(128)	até 128 caracteres	não
	storage_id	Número inteiro	smallint	storage_id > 0	não
	stockItemMovement_type	Booleano	boolean	-	não
	stockItemMovement_dateTime	Data e Horas	timestamp	-	não
	stockItemMovement_quantity	Número inteiro	smallint	stockItemMovement_quantity > 0	não

Tabela 3.16: Domínio dos Atributos da Entidade HouseAllergy.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
HouseAllergy	house_id	Número inteiro	bigint	house_id > 0	não
	allergy_allergen	Cadeia de caracteres de comprimento variável	character varying(75)	até 75 caracteres	não
	houseAllergy_alergicsNum	Número inteiro	smallint	houseAllergy_alergicsNum > 0	não

Tabela 3.17: Domínio dos Atributos da Entidade ListProduct.

Entidade	**Atributo**	**Domínio**	**Tipo Variável (PostgreSQL)**	**Restrições**	**Nullable**
ListProduct	house_id	Número inteiro	bigint	house_id > 0	não
	list_id	Número inteiro	smallint	list_id > 0	não
	category_id	Número inteiro	integer	category_id > 0	não
	product_id	Número inteiro	integer	product_id > 0	não
	listProduct_brand	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	sim
	listProduct_quantity	Número inteiro	smallint	listProduct_quantity > 0	não

Tabela 3.18: Domínio dos Atributos da Entidade StockItemAllergy.

Entidade	**Atributo**	**Domínio**	**Tipo Variável (PostgreSQL)**	**Restrições**	**Nullable**
StockItemAllergy	house_id	Número inteiro	bigint	house_id > 0	não
	stockItem_sku	Cadeia de caracteres de comprimento variável	character varying(128)	até 128 caracteres	não
	allergy_allergen	Cadeia de caracteres de comprimento variável	character varying(75)	até 75 caracteres	não

Tabela 3.19: Domínio dos Atributos da Entidade Date.

Entidade	**Atributo**	**Domínio**	**Tipo Variável (PostgreSQL)**	**Restrições**	**Nullable**
Date	date_date	Data (AAAA/MM/DD)	date	-	não

Tabela 3.20: Domínio dos Atributos da Entidade ExpirationDate.

Entidade	**Atributo**	**Domínio**	**Tipo Variável (PostgreSQL)**	**Restrições**	**Nullable**
ExpirationDate	house_id	Número inteiro	bigint	house_id > 0	não
	stockItem_sku	Cadeia de caracteres de comprimento variável	character varying(128)	até 128 caracteres	não
	date_date	Data (AAAA/MM/DD)	date	-	não
	date_quantity	Número inteiro	smallint	date_quantity > 0	não

3.4.3 Base de Dados

Os dados são armazenados de forma persistente numa Base de Dados (BD). A BD implementada é relacional uma vez que não se preveem alterações durante o uso, ou seja, as tabelas são de certa forma estáticas, não necessitando portanto do dinamismo oferecido por uma BD documental, por exemplo.

A escolha de qual o melhor Sistema de Gestão de Base de Dados (SGBD) assentava em três possibilidades, *SQL Server*, *PostgreSQL* e *MySQL*. O primeiro apesar de ser uma ferramenta com a qual o grupo estava familiarizado foi automaticamente excluída visto que um dos requisitos exigidos era ser *open source*, caraterística não presente nesta ferramenta. De seguida, ambas as ferramentas são *open source* e têm uma elevada compatibilidade com os principais fornecedores de serviços *cloud*. Pelo que a verdadeira distinção se prende com os factos:

- O *PostgreSQL* é compatível com as propriedades *Atomicity, Consistency, Isolation, Durability* (ACID), garantindo assim que todos os requisitos sejam atendidos;
- O *PostgreSQL* aborda a concorrência de uma forma eficiente com a sua implementação de *Multiversion Concurrency Control* (MVCC), que alcança níveis muito altos de concorrência;
- O *PostgreSQL* possui vários recursos dedicados à extensibilidade. É possível adicionar novos tipos, novas funções, novos tipos de índice, etc.

Assim sendo, foi escolhido o Sistema de Gestão de Base de Dados Relacional de Objetos (SGBDRO) *PostgreSQL*, como já anteriormente mencionado, na secção 1.3 do capítulo 1.

Implementação

Na BD foram desenvolvidas funções que garantem a consistência dos dados, por um lado na inserção de entidades cujos *IDs* sejam incrementais ou gerados consoante o desejado, por outro lado na remoção de entidades que se relacionam com outras.

Decidiu-se usar funções na BD em vez de criar métodos em *Java*, pois se imaginarmos um cenário onde a aplicação servidora esteja distribuída, existe um problema no controlo da concorrência na geração dos *ID*s. Tendo em conta que a BD não distribuída não existe o problema descrito.

Referências

[1] Eurostat. Internet use by individuals. http://ec.europa.eu/eurostat/documents/2995521/7771139/9-20122016-BP-EN.pdf, Dezembro 2016. [Online, Acedido a 23/05/2018].

[2] What is nfc? - nfc forum — nfc forum. https://nfc-forum.org/what-is-nfc/. [Online, Acedido a 15/04/2018].

[3] Rfid readers and rfid tags for any specialty rfid needs - rfid, inc. http://rfidinc.com/. [Online, Acedido a 15/04/2018].

[4] R. Fielding and J. Reschke. Rfc 7231 - hypertext transfer protocol (http/1.1): Semantics and content. https://tools.ietf.org/html/rfc7231, Junho 2014. [Online, Acedido a 15/04/2018].

[5] Edward Hieatt and Rob Mee. P of eaa: Repository. https://martinfowler.com/eaaCatalog/repository.html. [Online, Acedido a 25/05/2018].

[6] Martin Fowler. P of eaa: Unit of work. https://martinfowler.com/eaaCatalog/unitOfWork.html. [Online, Acedido a 25/05/2018].

[7] Randy Stafford. P of eaa: Service layer. https://www.martinfowler.com/eaaCatalog/serviceLayer.html. [Online, Acedido a 28/05/2018].

[8] Mike Stowe. Api best practices: Hypermedia (part 4.1) — mulesoft blog. https://blogs.mulesoft.com/dev/api-dev/api-best-practices-hypermedia-part-1/. [Online, Acedido a 28/05/2018].

[9] Home documents for http apis draft-nottingham-json-home-06. https://tools.ietf.org/html/draft-nottingham-json-home-06, Fevereiro 2017. [Online, Acedido a 15/04/2018].

[10] Kevin Swiber. Siren: a hypermedia specification for representing entities. https://github.com/kevinswiber/siren. [Online, Acedido a 15/04/2018].

[11] M. Nottingham, Akamai, and E. Wilde. Rfc 7807 - problem details for http apis. https://tools.ietf.org/html/rfc7807, Março 2016. [Online, Acedido a 15/04/2018].

[12] E. Rescorla and E. A. Schiffman. Rfc 2660 - the secure hypertext transfer protocol. https://tools.ietf.org/html/rfc2660, Agosto 1999. [Online, Acedido a 15/04/2018].

[13] J. Reschke. Rfc 7617 - the 'basic' http authentication scheme. https://tools.ietf.org/html/rfc7617, Setembro 2015. [Online, Acedido a 15/04/2018].

[14] Business Dictionary. What is inventory? definition and meaning - businessdictionary.com. http://www.businessdictionary.com/definition/inventory.html. [Online, Acedido a 04/26/2018].

[15] Investopedia. Stock keeping unit (sku). https://www.investopedia.com/terms/s/stock-keeping-unit-sku.asp. [Online, Acedido a 26/04/2018].

[16] Business Dictionary. What is standard stock item? definition and meaning - businessdictionary.com. http://www.businessdictionary.com/definition/standard-stock-item.html. [Online, Acedido on 26/04/2018].

[17] Gazelle Point of Sale Support. Stock item and non-stock item : Gazelle point-of-sale support. http://support.phostersoft.com/support/solutions/articles/17907-stock-item-and-non-stock-item. [Online, Acedido a 26/04/2018].

[18] Thad Scheer. Category, segment, and brand – what's the difference? – sphere of influence : Analytics studio. https://sphereoi.com/studios/category-segment-and-brand-whats-the-difference/. [Online, Acedido on 25/03/2018].

[19] Investopedia. Brand. https://www.investopedia.com/terms/b/brand.asp. [Online, Acedido a 26/04/2018].

Anexo A

Terminologia

A.1 Conceitos Básicos de Gestão de Stocks

Inventário - Um catálogo detalhado ou uma lista de bens ou propriedades tangíveis, ou os atributos ou qualidades intangíveis. Ler mais em [14].

***Stock Keeping Unit* (SKU) (Unidade de Manutenção de Stock, em Português)** - Um código de identificação de um produto e serviço para uma loja ou produto, muitas vezes retratado como um código de barras legível por máquinas que ajuda a rastrear o item para inventários. Ver exemplo A.1.1. Ler mais em [15].

Exemplo 1
Por exemplo, um armário pode ter pacotes de leite magro da marca X, 2 pacotes de leite magro da marca Y e 1 pacote de leite meio gordo da marca X. Logo, o armário contém 3 SKU, uma vez que um SKU se distingue pelo tamanho, cor, sabor, marca, etc.

Stock Item (Item em Stock, em Português) - Refere-se aos itens que se mantêm em stock físico na loja. O item de stock tem uma quantidade associada. Cada vez que uma venda é feita para aquele item, a sua quantidade será deduzida. Artigo aprovado para aquisição, armazenamento e emissão, e geralmente mantido à mão. Ler mais em [16] e [17].

Product Category (Categoria de Produtos, em Português) - Taxonomias de classificação que subdividem um Setor ("yet another market construct") nos diferentes tipos de produtos para os quais existe demanda. Quanto mais especializada for uma categoria, mais especializado é o produto. Ler mais em [18].

Nota: Neste projeto apenas se consideram as categorias de maior dimensão, são elas, por exemplo, Laticínios, Bebidas, Frescos, Congelados, entre outras.

Brand (Marca, em Português) - Um símbolo de identificação, marca, logótipo, nome, palavra e/ou frase que as empresas usam para distinguir os seus produtos dos outros. Ler mais em [19]

Segmentation (Segmento, em Português) - Quando os estrategistas de marca falam sobre segmento,referem-se à segmentação do consumidor/audiência. A maneira antiga de

abordar isso era através da demografia (idade, sexo, etnia, faixa de renda, urbano-rural, etc.). Agora a segmentação é VALS (valores, atitudes e estilo de vida). Ler mais em [18].

Nota: Neste projeto o segmento é a quantidade presente numa embalagem, i.e., para um pacote de leite de 1L, o segmento é 1L.

Variety (Variedade, em Português) - A variedade é confusa porque pode ser difícil de entender onde a especialização da segmentação termina e a especialização em prol da Variedade começa. A variação é sobre a personalização de um produto para se adequar ao caráter do consumidor individual. Ver exemplo A.1.2. Ler mais em [18].

Exemplo 2
Note-se um pacote de leite com as caraterísticas, quantidade líquida igual a 1L, da marca X e do tipo UHT magro. Então, identificar-se-ia da seguinte forma:

- Categoria: Laticínios
- Produto: Leite
- Marca: X
- Segmento: 1L
- Variedade: UHT Magro