



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

3.1	Arquitetura Geral do Projeto	9
3.2	Estrutura por Camadas do Projeto	10
4.1	Modelo Entidade-Associação	14
5.1	Planeamento	28

Lista de Tabelas

4.1	Domínio dos Atributos da Entidade House.	15
4.2	Domínio dos Atributos da Entidade User.	15
4.3	Domínio dos Atributos da Entidade Allergy.	15
4.4	Domínio dos Atributos da Entidade Recipe.	16
4.5	Domínio dos Atributos da Entidade SystemRecipe.	16
4.6	Domínio dos Atributos da Entidade UserRecipe.	16
4.7	Domínio dos Atributos da Entidade SharedRecipe.	17
4.8	Domínio dos Atributos da Entidade List.	17
4.9	Domínio dos Atributos da Entidade SystemList.	17
4.10	Domínio dos Atributos da Entidade UserList.	17
4.11	Domínio dos Atributos da Entidade Category.	18
4.12	Domínio dos Atributos da Entidade Product.	18
4.13	Domínio dos Atributos da Entidade StockItem.	19
4.14	Domínio dos Atributos da Entidade Ingredient.	19
4.15	Domínio dos Atributos da Entidade Storage.	20
4.16	Domínio dos Atributos da Entidade UserHouse.	20
4.17	Domínio dos Atributos da Entidade StockItemStorage.	20
4.18	Domínio dos Atributos da Entidade StockItemMovement.	21
4.19	Domínio dos Atributos da Entidade HouseAllergy.	21
4.20	Domínio dos Atributos da Entidade ListProduct.	21
4.21	Domínio dos Atributos da Entidade StockItemAllergy.	22
4.22	Domínio dos Atributos da Entidade Date.	22
4.23	Domínio dos Atributos da Entidade ExpirationDate.	22

Índice

1	Introdução	1
1.1	Contexto	1
1.2	Metas e Objetivos	2
1.3	Abordagem do Projeto	2
1.4	Estrutura do Relatório	3
2	Sistema de Gestão de Stocks	5
2.1	Sistema Smart Stocks	5
2.2	Requisitos Funcionais e Opcionais	7
3	Solução Proposta	9
3.1	Abordagem	9
3.2	Análise	10
4	Progresso	13
4.1	Planeamento	13
4.2	Web API	13
4.3	Algoritmo de Previsão	13
4.4	Aplicações Móvel e Web	13
4.5	Modelo de Dados	14
4.5.1	Modelo Entidade-Associação	14
4.5.2	Domínio dos Atributos	15
4.5.3	Base de Dados	23
4.5.4	Acesso a Dados	24
4.5.5	Lógica de Negócio	25
5	Conclusões	27
5.1	Planeamento	27
5.2	Sumário	29
5.3	Trabalho Futuro	29

A Terminologia	33
A.1 Conceitos Básicos de Gestão de Stocks	33

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, 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 despesa 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 massiva 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 de um sistema de gestão de stocks na agenda de tarefas de uma organização doméstica poderá ser uma mais valia. Para o concretizar desse 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:

- Rever e sumarizar os conteúdos das aplicações mais populares e com classificação mais elevada.
- Criar o primeiro protótipo de uma aplicação simples e educativa com base em estratégias de usabilidade e manutenção.

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 toda a parte do trabalho desenvolvido para o projeto, ou seja os requisitos do projeto, a solução implementada e acesso a dados.

1.4 Estrutura do Relatório

O relatório está estruturado em 5 capítulos.

O capítulo 2 formula o problema, detalhando os requisitos do projeto e são apresentados os casos de uso.

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

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

É no capítulo 6 que se retiram conclusões face ao trabalho desenvolvido em relação ao trabalho inicialmente previsto. Este planeamento inicial é revelado na secção 4.1. Para finalizar, propõe-se o trabalho a realizar futuramente, na secção 4.3.

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 2.2. Para a leitura deste capítulo é recomendada a leitura do Anexo A, a leigos em gestão de stocks, para uma melhor compreensão de certos termos.

2.1 Sistema Smart Stocks

O 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.

Em seguida listam-se as diversas entidades relevantes que compõem o sistema de informação que permita 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. 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.
- A casa pode ter vários itens em stock presentes.
- 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 identificado univocamente 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 caracterizado 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 Requisitos Funcionais e Opcionais

Requisitos Funcionais

- Informar o utilizador dos produtos existentes, a sua validade e a sua quantidade;
- Alertas sobre os produtos que estão perto da data de validade;
- Geração da lista de compras com os produtos em falta;
- Possibilidade de especificar os produtos a ter sempre em stock bem como as suas quantidades mínimas;
- Listas partilhadas entre utilizadores da mesma casa;
- Criação de Listas;
- Especificação das alergias dos membros da casa.

Requisitos Opcionais

- Lista de produtos quase a expirar;
- Lista de produtos indesejados (Lista Negra);
- Lista 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.

Capítulo 3

Solução Proposta

Neste capítulo pretende-se abordar de forma geral a solução implementada para resolver o problema apresentado no capítulo 2.

3.1 Abordagem

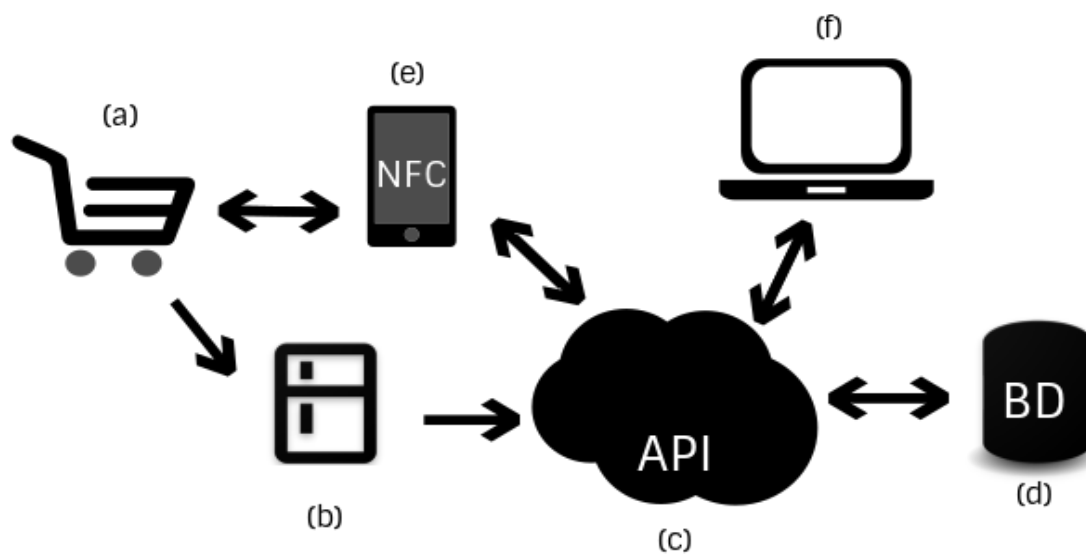


Figura 3.1: Arquitetura Geral do Projeto

Após uma ida às compras, os itens adquiridos, Figura 3.1(a), são armazenados nos seus respetivos locais, Figura 3.1(b). Como forma de automatizar a recolha de informação relativa quer aos artigos obtidos quer às suas características, utilizam-se sensores. A utilização destes sensores 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, Figura 3.1(c). Assim, estes dados são posteriormente tratados e armazenados de forma persistente na Base de Dados (BD), Figura 3.1(d). A API é responsável por retornar dados para as aplicações cliente, Figura 3.1(e, f). É ainda nesta que está presente o algoritmo de previsão de stocks utilizado para fazer estimativas 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*, 3.1(e). Os detalhes dos itens são especificados pelo utilizador e carregados para a *tag*. Já os segundos contêm os seus rótulos digitais com o seu detalhe guardado pelos fornecedores.

3.2 Análise

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 3.2.

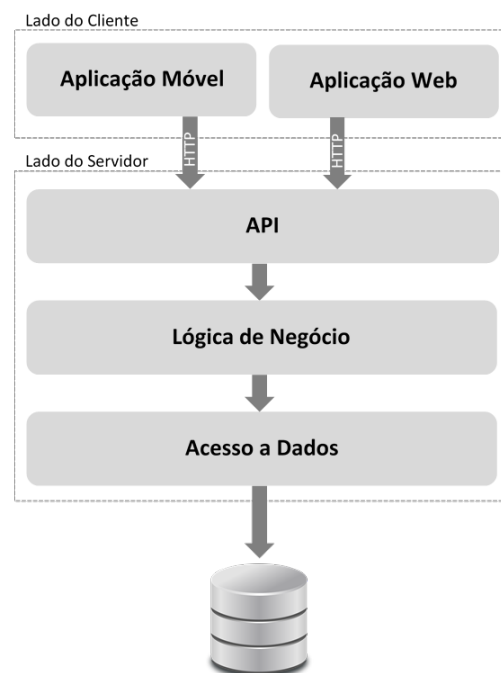


Figura 3.2: Estrutura por Camadas do Projeto

O lado do servidor inclui três camadas e expõe uma API *Web*. A Camada de Acesso a Dados (DAL) é a responsável pelas leituras e escritas exercidas na BD, Sistema de Gestão de Base de Dados (SGBD) *PostgreSQL*. Esta camada é produzida com a linguagem de programação *Java*, usando a *Java Persistent API* (JPA). A Camada da Lógica de Negócio

(BLL) é responsável pela execução das regras de negócio. A implementação desta camada é, também, implementada com linguagem *Java*, usando a JPA. Os *controllers* foram desenvolvidos em *Java* com a *framework* da *Spring*, chamada de *Spring Boot*. A API *Web* disponibiliza recursos em diferentes *hypermedia*, *Siren* e *Json Home*.

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, implementada utilizando a linguagem *JavaScript*, com o auxílio da *framework Express*.

Capítulo 4

Progresso

Neste capítulo apresenta-se o progresso do projeto, assim como, o planeamento inicial e de que forma foi cumprido. São expostas também conclusões face ao desempenho do grupo e trabalho realizado.

4.1 Planeamento

4.2 Web API

4.3 Algoritmo de Previsão

4.4 Aplicações Móvel e Web

4.5 Modelo de Dados

4.5.1 Modelo Entidade-Associação

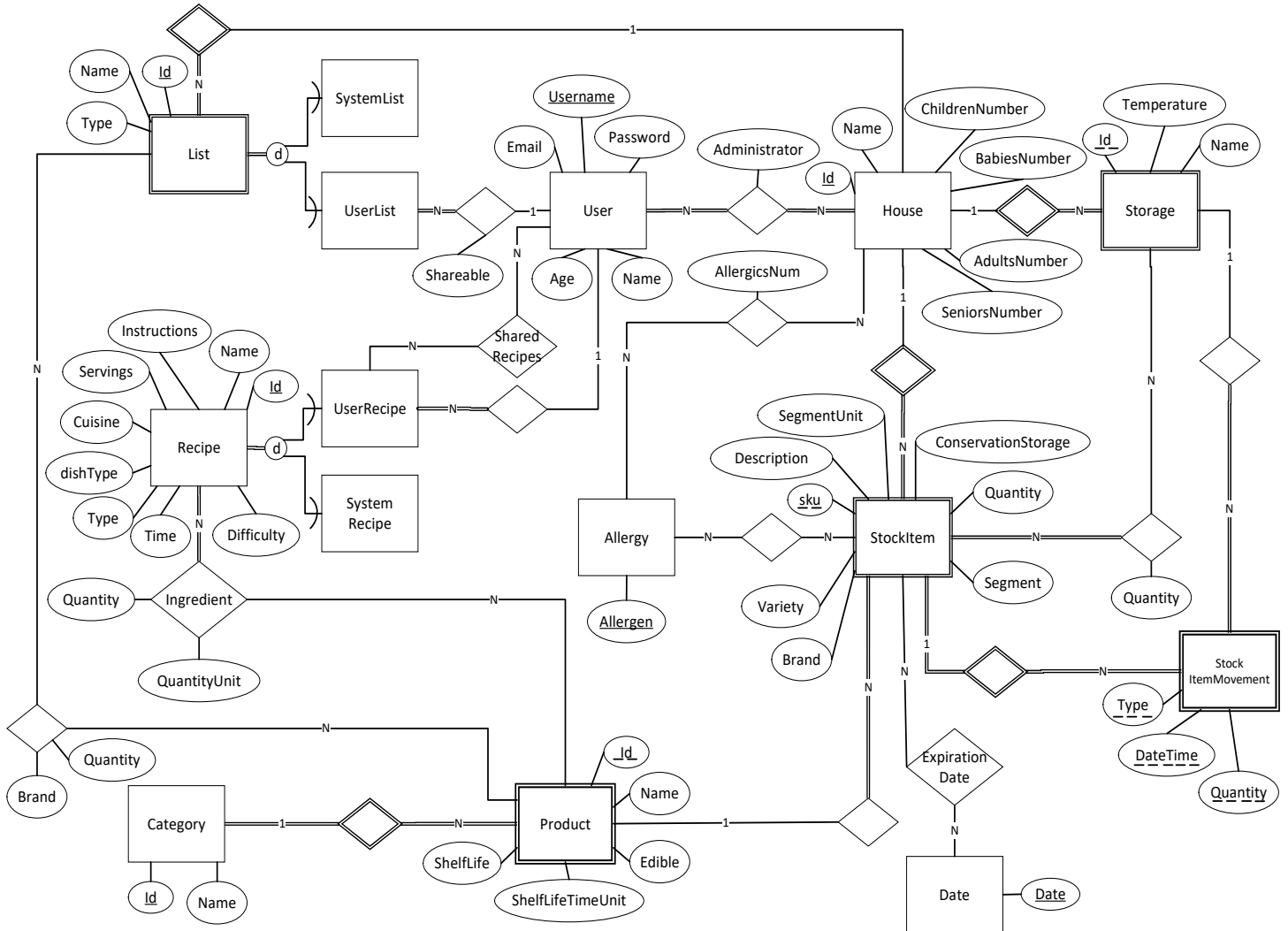


Figura 4.1: Modelo Entidade-Associação

4.5.2 Domínio dos Atributos

Tabela 4.1: 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 4.2: 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 4.3: 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 4.4: Domínio dos Atributos da Entidade Recipe.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
Recipe	recipe_id	Número inteiro auto-incrementado	bigserial	-	não
	recipe_name	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	não
	recipe_instructions	Cadeia de caracteres de comprimento variável	text	-	não
	recipe_difficulty	Cadeia de caracteres de comprimento variável	character varying(9)	recipe_difficulty in ['easy', 'average', 'difficult']	sim
	recipe_time	Número inteiro	smallint	recipe_time > 0	sim
	recipe_servings	Número inteiro	smallint	recipe_servings > 0	sim
	recipe_cuisine	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	sim
	recipe_dishType	Cadeia de caracteres de comprimento variável	character varying(35)	até 35 caracteres	sim
	recipe_type	Cadeia de caracteres de comprimento variável	character varying(7)	recipe_type in ['system', 'user']	não

Tabela 4.5: Domínio dos Atributos da Entidade SystemRecipe.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
System Recipe	recipe_id	Número inteiro	bigint	recipe_id > 0	não

Tabela 4.6: Domínio dos Atributos da Entidade UserRecipe.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
User Recipe	recipe_id	Número inteiro	bigint	recipe_id > 0	não
	user_username	Cadeia de caracteres de comprimento variável	character varying(30)	até 30 caracteres	não

Tabela 4.7: Domínio dos Atributos da Entidade SharedRecipe.

Entidade	Atributo	Domínio	Tipo Variável (PostgreSQL)	Restrições	Nullable
Shared Recipe	recipe_id	Número inteiro	bigint	recipe_id > 0	não
	user_username	Cadeia de caracteres de comprimento variável	character varying(30)	até 30 caracteres	não

Tabela 4.8: 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 4.9: 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 4.10: 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 4.11: 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 4.12: 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	integer	-	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 4.13: 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 decimal	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 4.14: 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 4.15: 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 4.16: 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 4.17: 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 4.18: 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 4.19: 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 4.20: 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 4.21: 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 4.22: Domínio dos Atributos da Entidade Date.

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

Tabela 4.23: 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

4.5.3 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*, característica 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 *IDs*. Tendo em conta que a BD não distribuída não existe o problema descrito.

4.5.4 Acesso a Dados

Uma vez armazenados os dados de forma persistente é indispensável realizar escritas e leituras sobre os mesmos. Para tal, desenvolveu-se a chamada Camada de Acesso a Dados (DAL).

Para implementar esta camada, ponderaram-se duas opções, *Java Persistent API* (JPA) e *Java Database Connectivity Template* (Jdbc Template). Apesar de Jdbc Template permitir um maior controlo do lado do programador, não se fizeram notar discrepâncias significativas, pelo que se optou então por JPA, por questões de familiaridade.

Como o modelo de dados é relativamente extenso (mais de 20 tabelas) o uso de JPA torna-se benéfico. Tal, permite reduzir a extensa repetição de código envolvido para suportar as operações básicas de *Create, Read, Update e Delete* (CRUD) em todas as entidades.

Aqui, o requisito é o acesso aos dados na BD e o suporte para as operações CRUD em quase todas as tabelas. Desta forma criou-se uma interface *Repository* com métodos que garantem não só essas operações, como outras para facilitar a obtenção de dados de determinada maneira. Existe ainda a possibilidade de criar *queries*, definindo métodos nas interfaces JPA. O uso de JPA obriga a representar o esquema/modelo da BD em classes *Java, Plain Old Java Objects* (POJO).

Implementação

No acesso a dados, são utilizados dois padrões de desenho: Padrão *Repository* e Padrão *Unit Of Work*. Esta componente é, salvo exceções, gerada através da JPA.

Cada entidade presente na BD é mapeada numa classe em Java, que representa o modelo da mesma. Esta classe tem várias anotações da JPA para referir a Chave-Primária, Chave-Estrangeira, relações entre entidades, etc. Em conjunto estas classes Java formam o modelo utilizado entre as camadas internas do lado do servidor. Mais à frente serão apresentados outros tipos de objeto usados para representar as entidades recebidas e enviadas para o exterior.

4.5.5 Lógica de Negócio

É fundamental fazer cumprir as regras, restrições e toda a lógica da gestão dos dados para o correto funcionamento das aplicações. 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.

Implementação

Foram criados serviços para as principais entidades, que dispõem de diversas funcionalidades. É de salientar que um serviço está fortemente ligado a um ou mais repositórios.

Capítulo 5

Conclusões

Neste capítulo apresentam-se as conclusões relativas ao desempenho e trabalho realizado pelo grupo. São efetuadas comparações face ao planeamento inicial previsto e ao que realmente sucedeu, como forma de analisar e apreciar o trabalho realizado.

5.1 Planeamento

Na Figura 5.1 está representado o planeamento ao longo do projeto, igualmente, estão as percentagens de trabalho realizado em cada um dos principais pontos. As datas críticas do projeto estão também presentes na Figura 5.1, de modo a que se possa observar da melhor maneira as tarefas e prazos a cumprir.

Tentou-se, ao máximo, que o maior número de tarefas pudesse ser executado em paralelo, para assim aumentar a eficiência e melhor gerir os recursos disponíveis. Essencialmente as tarefas foram divididas de acordo com a camada/bloco onde se encontravam, sendo requisito, o seu desenvolvimento, teste e documentação.

Note-se que a partir da semana catorze até à semana da entrega final, as tarefas a realizar dependerão vivamente da qualidade da entrega da versão beta. Servirá também para melhoramentos, visto que a melhoria se trata de um processo contínuo. Eventualmente os requisitos funcionais não entregues na versão beta, serão executados também no decorrer desse período.

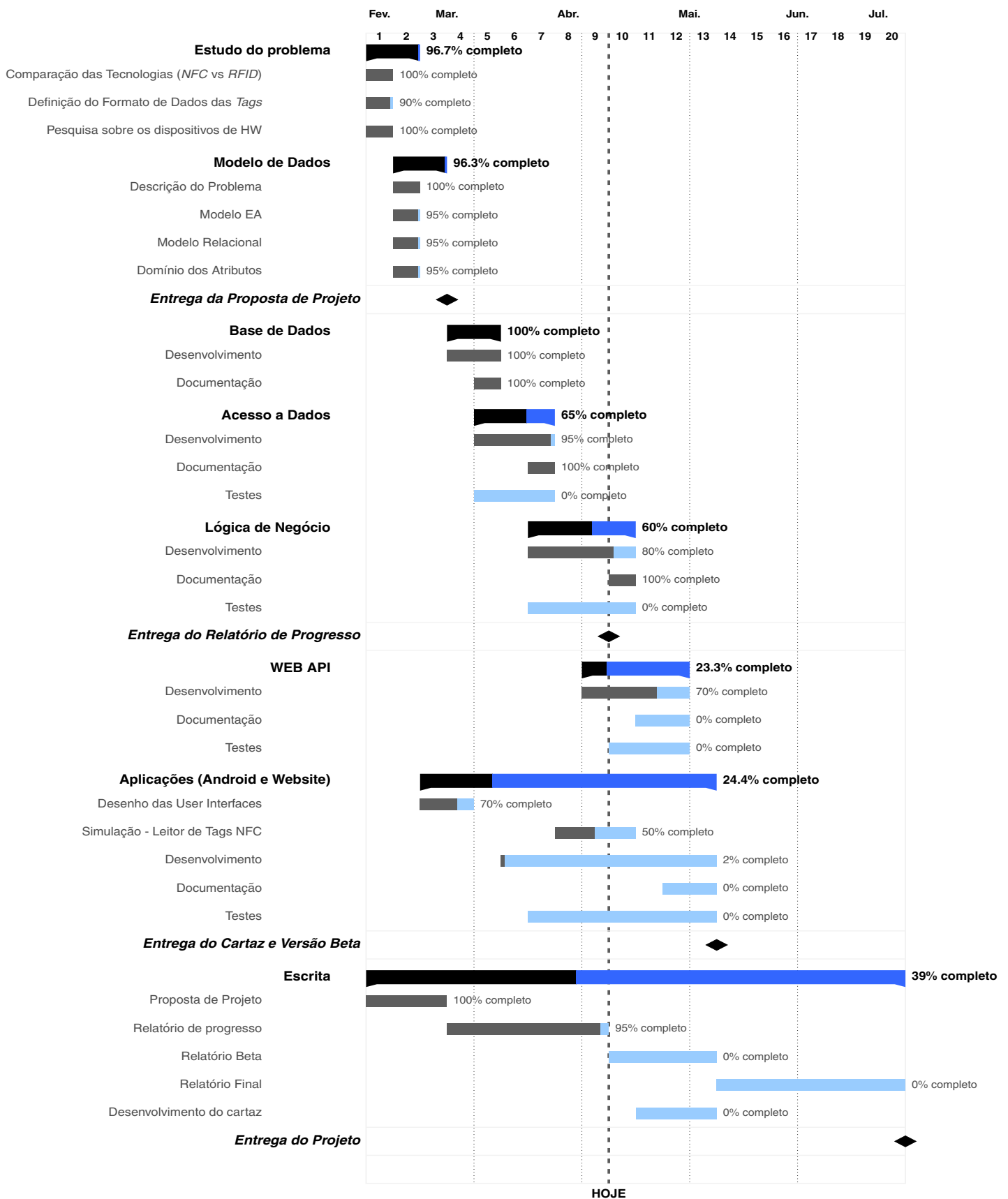


Figura 5.1: Planejamento

5.2 Sumário

Nas semanas iniciais foi realizada pesquisa de forma a melhor entender conceitos, dificuldades e potenciais resoluções e/ou abordagens.

De seguida definiu-se o problema e como seria solucionado, tendo também sido apresentada a proposta de projeto publicamente.

A partir das seguintes datas, começou-se a efetiva implementação das várias camadas. Até à data foram desenvolvidas as camadas: Base de Dados, Acesso a Dados, Lógica de Negócio e iniciada a API *Web*. De forma geral os requisitos foram cumpridos. Foi notado um ligeiro atraso de, aproximadamente, uma semana em relação ao planeamento esperado.

5.3 Trabalho Futuro

Existe ainda trabalho crucial por realizar, nomeadamente a API *Web* e as aplicações móvel e *web*. Seria deveras importante recuperar o atraso que se fez sentir até agora, pelo que será necessário um esforço adicional pelo grupo de trabalho.

Referências

- [1] Eurostat. Internet use by individuals. <http://ec.europa.eu/eurostat/documents/2995521/7771139/9-20122016-BP-EN.pdf>, December 2016. [Online, Accessed on 05/23/2018].
- [2] What is nfc? - nfc forum — nfc forum. <https://nfc-forum.org/what-is-nfc/>. [Online, Accessed on 04/15/2018].
- [3] Rfid readers and rfid tags for any specialty rfid needs - rfid, inc. <http://rfidinc.com/>. [Online, Accessed on 04/15/2018].
- [4] Business Dictionary. What is inventory? definition and meaning - businessdictionary.com. <http://www.businessdictionary.com/definition/inventory.html>. [Online, Acedido a 04/26/2018].
- [5] Investopedia. Stock keeping unit (sku). <https://www.investopedia.com/terms/s/stock-keeping-unit-sku.asp>. [Online, Accessed on 04/26/2018].
- [6] Business Dictionary. What is standard stock item? definition and meaning - businessdictionary.com. <http://www.businessdictionary.com/definition/standard-stock-item.html>. [Online, Accessed on 04/26/2018].
- [7] 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, Accessed on 04/26/2018].
- [8] 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; Accessed on 03/25/2018].
- [9] Investopedia. Brand. <https://www.investopedia.com/terms/b/brand.asp>. [Online, Accessed on 04/26/2018].

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 [4].

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 [5].

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 [6] e [7].

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 [8].

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 [9]

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 [8].

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 [8].

Exemplo 2

Note-se um pacote de leite com as características, 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