Gestão Inteligente de Stocks

Ana Santos
Inês Soares
Nuno Veloso

Orientadores Matilde Pato
Nuno Datia

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

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

Março de 2018

Resumo

Com a evolução da *Internet of Things (IoT)*, é expectável que tarefas básicas do dia-a-dia passem a usar estas tecnologias. Tal avanço minimizaria o tempo desperdiçado pelas pessoas em atividades realizáveis por máquinas. Ao automatizarmos a recolha de dados relacionada com os stocks de produtos em casa, simplificamos a gestão dos mesmos. Desta forma, auxiliamos os utilizadores a manter o stock adequado às suas necessidades, bem como alertá-lo para a proximidade do fim da validade e/ou stock dos produtos. Assim, o nosso 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? Como impedir artigos fora de prazo?". Se entendermos que a nossa casa funciona como uma empresa, onde existem pessoas que podem realizar as mesmas tarefas, e.g. ir às compras seguindo uma lista previamente elaborada, capacitamos qualquer elemento da família para exercer a compra.

Palavras-chave: automatização; gestão; Iot; stocks; tarefas.

Abstract

Nowadays with the constant evolution of the Internet of Things (IoT), it would be expected that basic day-to-day tasks would be under these technologies. Such an advance would minimize the time wasted by people in activities that can be carried out by machinery. By automating the collection of data related to the stocks of products at home, we simplify the management of stocks. In this way, we help users to keep the stock adequate to their needs, as well as to alert them to the proximity of the product term. Thus, our work is aimed at answering questions such as: "In what way can we avoid inconveniences caused at the time of refueling our storages? Or how to control stockpiles of food and other products? How to prevent outdated items?". If we understand that our house functions as a company, where there are people who can perform the same tasks, e.g., go shopping following a prepared list, we enable any member of the family to carry out the purchase.

Keywords: automation; Internet of Things (IoT); management; stocks; tasks.

Lista de Figuras

Lista de Tabelas

Índice

Capítulo 1

Introdução

Atualmente, são várias as aplicações responsáveis por fornecer listas de compras, porém carecem de controlo de stocks e conhecimento dos hábitos dos seus utilizadores. Neste projeto pretende-se desenvolver uma aplicação que permite a gestão e controlo de stocks de produtos alimentares na despensa de qualquer casa. Os padrões de consumo dos utilizadores são registados, permitindo prever a data da rotura de stock de produtos.

1.1 Abordagem

No contexto do projeto assume-se a existência de duas formas de apresentação para os produtos: avulsos e embalados. Os primeiros são conservados em sistemas de arrumação marcados com *tags* programáveis por *smartphones*. Os detalhes dos itens são especificados pelo utilizador e carregados para a *tag*. Enquanto que para os produtos embalados, admite-se que os produtores utilizam *tags*, *Near-Field Communication* (NFC) ou *Radio-frequency Identification* (RFID), para guardar os rótulos de forma digital e em formato standard.

Após a aquisição, os artigos são armazenados em locais que devem dispor de dispositivos de hardware equipados com leitores de *tags*. É recolhida a informação presente na *tag*, identificado o tipo de movimento (entrada ou saída) e enviado para a API *Web*.

1.2 Metas e Objetivos

Têm-se como objetivos, os seguintes pontos:

- Desenho e Implementação das Aplicações Móvel e Web
- Elaboração do Algoritmo de Previsão de Stocks

1.3 Especificações do Projeto e Resumo da Solução

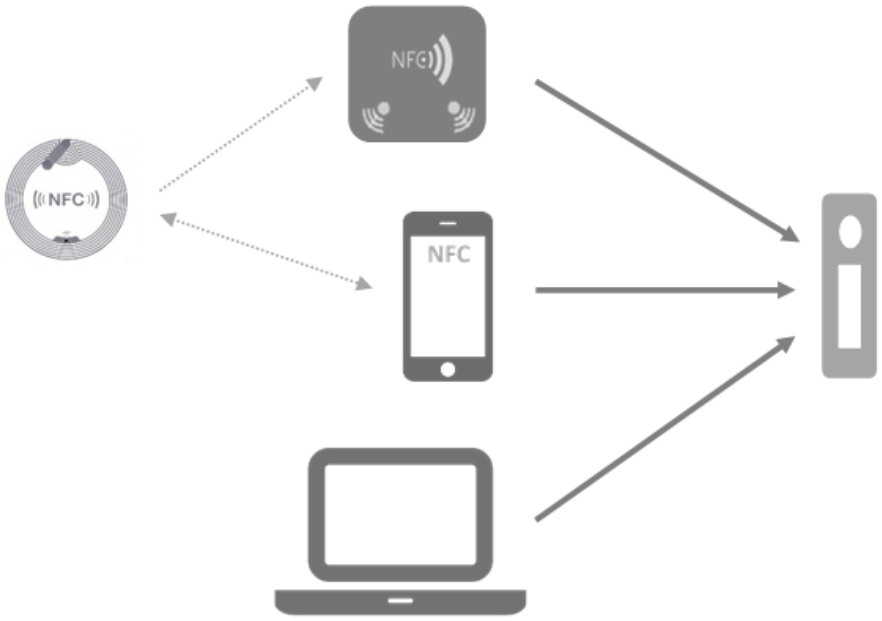
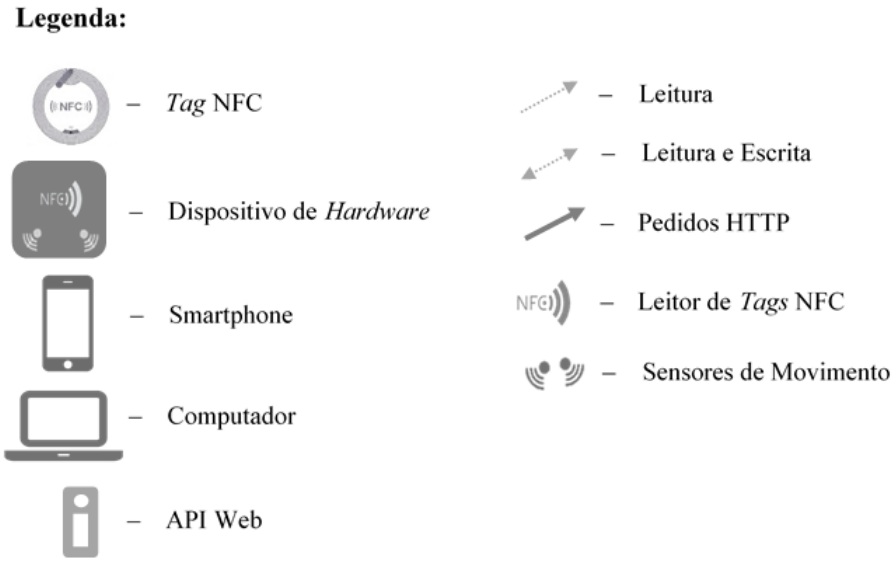


Figura 1.1: Estrutura do Projeto



Com a Figura 1.1 pretende-se não só apresentar os principais componentes do projeto, bem como demonstrar de forma breve a relação dos mesmos. É de destacar que uma *tag* pode ser lida por um dispositivo de *hardware* munido de um leitor de *tags*. Um *smartphone* equipado com tecnologia NFC pode escrever na *tag* NFC, tal é necessário para identificar produtos avulsos presentes num sistema de arrumação. Tanto o dispositivo de *hardware* como as aplicações, móvel e *web*, comunicam com a API *Web*.

O projeto é composto por 2 blocos principais, que se relacionam. A Figura 1.2 representa esses blocos.

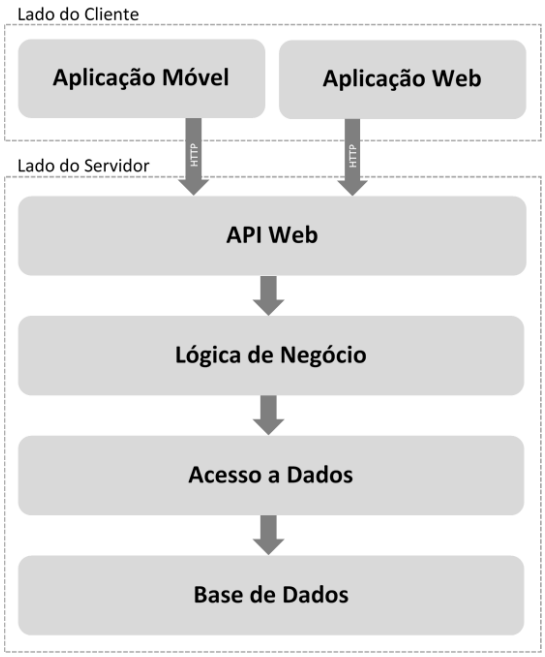


Figura 1.2: Arquitetura do Projeto

O lado do servidor incluí quatro camadas e expõe uma API *Web*. A camada da Base de Dados (BD) é realizada com o Sistema de Gestão de Base de Dados (SGBD) *PostgreSQL*. A Camada de Acesso a Dados (DAL) é responsável pelas leituras e escritas à BD. Esta camada é produzida com a linguagem de programação *Java*, com a *Java Persistent API* (JPA). A Camada da Lógica de Negócio (BLL) é responsável pela gestão dos dados obtidos da BD ou dos *controllers*. A implementação desta camada recorreu à mesma ferramenta que foi usada na DAL. Os *controllers* foram desenvolvidos em *Java* com a *framework* da *Spring*, chamada de *Spring Boot*. A API *Web* disponibiliza recursos em diferentes *hypermedia*.

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 auxilio da *framework Express*.

1.4 Estrutura do Relatório

O relatório está estruturado em 4 capítulos.

O capítulo 2 formula o problema, detalhando os requisitos do projeto. São ainda apresentadas as dificuldades encontradas ao longo do projeto.

No capítulo 3 o problema é solucionado, sendo apresentada, com detalhe, a solução implementada. O capítulo foi dividido pelas várias camadas que compõe o projeto. Na secção 3.1

expõe-se a modelagem dada aos dados. Por conseguinte, a secção 3.2 explicita de que forma esses dados foram armazenados, sendo ainda justificadas as decisões tomadas nesta camada. A secção 3.3 fundamenta a seleção das ferramentas utilizadas para a implementação. A lógica de negócio está presente na secção 3.4.

É no capítulo 4 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

Formulação do Problema

Neste capítulo o problema é descrito de forma detalhada na secção 2.1, bem como os requisitos funcionais e não funcionais na secção 2.2. A secção 2.3 apresenta as dificuldades que surgiram no decorrer do projeto.

2.1 Descrição do Problema

No âmbito do projeto existe a necessidade de criar um sistema de informação que permita gerir os itens em stock de uma dada casa.

Uma casa está associada a um ou mais utilizadores, podendo um utilizador ter várias casas. Cada casa é caracterizada por um identificador único, um nome, atribuído por um utilizador no momento de registo da casa, quantos bebés, crianças, adultos e seniores vivem nessa casa. Cada utilizador é identificado univocamente por um email ou por um nome de utilizador, pelo nome da própria pessoa, idade e uma password. Para cada casa podem existir um ou mais administradores. Um utilizador pode criar as suas listas e as suas receitas. As listas que este cria pode decidir se quer partilhar com os restantes utilizadores da casa a que pertence, nunca podendo partilhar com utilizadores fora da sua casa. Existem listas do sistema que são comuns a todos os utilizadores registados, contudo são particulares a cada casa. As listas partilhadas pelos vários utilizadores dependem dos produtos que cada utilizador tem em casa. As receitas que um utilizador cria podem ser partilhadas com todos os utilizadores registados ou só com determinados utilizadores. Existem ainda um conjunto de receitas que são partilhadas por todos os utilizadores registados.

Cada receita é identificada por um identificador único, um nome, uma preparação, uma dificuldade, um tempo, quantas doses, qual o tipo de cozinha e qual o tipo de prato. Cada receita pode ter vários ingredientes. Para cada receita deve ser possível saber a quantidade dos vários ingredientes que a compõem. Cada lista é identificada por um identificador único e um nome. Uma lista pode ter vários produtos. Para os produtos presentes numa lista pode ser possível saber a sua marca e a quantidade de um produto na lista. Um produto é identificado pelo seu identificador único, contém um nome, se é comestível ou não, e a

validade perecível. Um produto pertence a uma categoria, podendo uma categoria ter vários produtos. Uma categoria é identificada por um identificador único ou um nome. Um produto pode ter vários itens em casa. Uma casa pode ter vários itens presentes na mesma.

Um item presente numa casa é identificado por um identificador único 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, isto é, se entrou ou saiu de um local de armazenamento. Para cada movimento deve ser possível saber o tipo de movimento (entrada ou saída), a data em que ocorreu o movimento e a quantidade de produtos que ocorrem num movimento. Para cada casa existem vários locais de armazenamento dos itens, por exemplo armários, frigoríficos, etc. Cada local de armazenamento é caraterizado por um identificador único, a temperatura e um nome. Um local de armazenamento pode ter vários itens presentes numa casa e vários movimentos. Para cada local de armazenamento deve ser possível saber a quantidade de cada item.

Para cada casa deve ser possível saber que alergias os seus membros têm e para cada alergia saber o número de membros que têm essa alergia (os membros não precisam necessariamente de estar registados). Deve também ser possível saber os alergénios de um item presente na casa.

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;
- Lista de Compras *Offline*;
- Listas partilhadas entre utilizadores da mesma casa;
- Criação de Listas (As Minhas 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);
- Sugestão de receitas que utilizem os produtos mais perto do fim da validade;
- Inserção de receitas (As Minhas Receitas);
- Inserir refeições extraordinárias de eventos a realizar num futuro próximo, para acrescentar alimentos não básicos à lista de compras;
- Especificação dietas alimentares (Vegetarianos, *Vegans*, etc.) a cada utilizador da casa.

2.3 Dificuldades Encontradas

Para a realização deste projeto encontrámos dificuldades nos aspetos a seguir referidos.

2.3.1 Rótulos em Formato Não Digital

Nos dias de hoje, os produtos não possuem rótulos digitais. Isto é um problema para a concretização do projeto, na medida em que se torna menos eficiente a recolha dos dados presentes nos produtos. Contudo, assumindo que este problema é resolvido fora do âmbito do projeto, apenas é preciso definir um formato standard de como os dados devem ser armazenados nas *tags*, que podem ser NFC ou RFID. Num cenário ideal, este formato deve ser respeitado por todos os embaladores. Assim, os produtos têm um rótulo, código de barras e uma *tag* NFC ou RFID, com a informação necessária. Está fora do âmbito do trabalho implementar o suporte hardware para a leitura das *tags* e qual o sentido do movimento (entrada ou saída). Assume-se que essas informações são disponibilizadas num formato conhecido.

2.3.2 Ausência de Identificador Único nos Itens

Os itens não dispõem de um identificador unívoco. Alguns deles contêm um lote e um número de série. A ausência deste identificador impede a distinção entre itens iguais, o que impossibilita saber se entrou um novo item no local de armazenamento ou se saiu um dos itens presentes. Tal facto torna a gestão dos stocks dependente do dispositivo de hardware para distinguir o tipo de movimento.

Capítulo 3

Solução Proposta

Neste capítulo pretende-se dar ênfase à solução implementada para resolver o problema apresentado no capítulo 2. Este capítulo está divido nas seguintes secções, Modelo de Dados na secção 3.1, Base de Dados na secção 3.2, Acesso a Dados na secção 3.3 e Lógica de Negócio na secção 3.4.

3.1 Modelo de Dados

3.1.1 Modelo Entidade-Associação

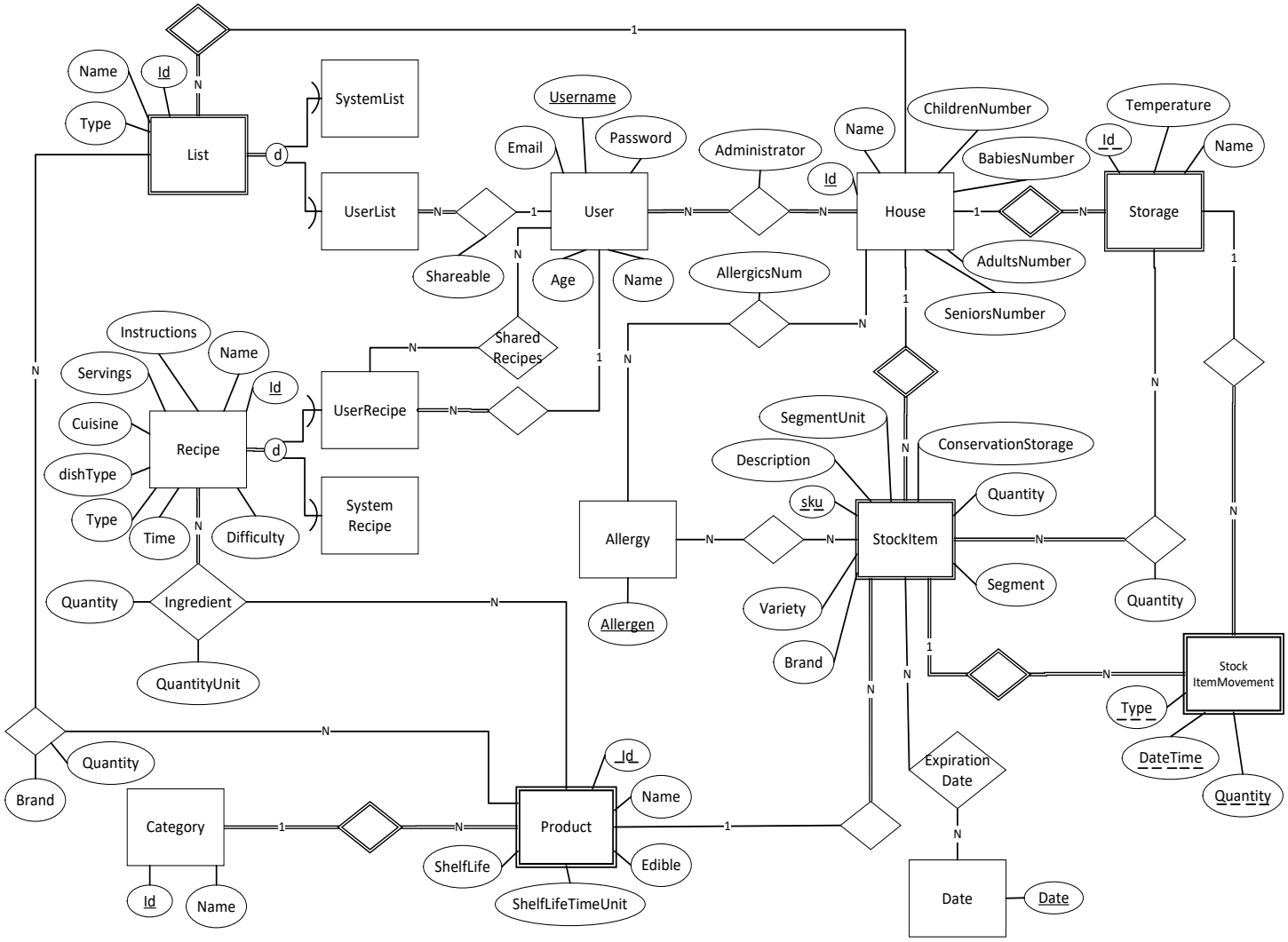


Figura 3.1: Modelo Entidade-Associação

3.1.2 Modelo Relacional

House(house_id, house_name, house_babiesNumber, house_childrenNumber, house_adultsNumber, house_seniorsNumber)

CP: (house_id)

User(user_username, user_email, user_age, user_name, user_password)

CP: (user_username)

OCC: (user_email)

Allergy(allergy_allergen)

CP: (allergy_allergen)

Recipe(recipe_id, recipe_name, recipe_instructions, recipe_difficulty, recipe_time, recipe_servings, recipe_cuisine, recipe_dishType, recipe_type)
CP: (recipe_id)

SystemRecipe(recipe_id)
CP: (recipe_id)
CE: {(recipe_id) ref Recipe}

UserRecipe(recipe_id, user_username)
CP: (recipe_id)
CE: {(recipe_id) ref Recipe, (user_username) ref User}

SharedRecipe(recipe_id, user_username)
CP:(recipe_id, user_username)
CE: {(recipe_id) ref UserRecipe, (user_username) ref User}

List(house_id, list_id, list_name, list_type)
CP: (house_id, list_id)
CE: {(house_id) ref House}

SystemList(house_id, list_id)
CP: (house_id, list_id)
CE: {(house_id, list_id) ref List}

UserList(house_id, list_id, user_username, list_shareable)
CP: (house_id, list_id)
CE: {(house_id, list_id) ref List, (user_username) ref User}

Category(category_id, category_name)
CP: (category_id)
OCC: (category_name)

Product(category_id, product_id, product_name, product_edible, product_shelfLife, product_shelfLifeTimeUnit)
CP: (category_id, product_id)
CE: {(category_id) ref Category}

StockItem(house_id, stockItem_sku, category_id, product_id, stockItem_brand, stockItem_segment, stockItem_variety, stockItem_quantity, stockItem_segmentUnit, stockItem_description, stockItem_conservationStorage)
CP: (house_id, stockItem_sku)

OCC: (house_id, category_id, product_id, stockItem_brand, stockItem_segment, stockItem_variety)

CE: {(house_id) ref House, (category_id, product_id) ref Product}

Ingredient(recipe_id, category_id, product_id, ingredient_quantity, ingredient_quantityUnit)

CP: (recipe_id, category_id, product_id)

CE: {(recipe_id) ref Recipe, (category_id, product_id) ref Product}

Storage(house_id, storage_id, storage_name, storage_temperature)

CP:(house_id, storage_id)

CE: {(house_id) ref House}

UserHouse(house_id, user_username, userHouse_administrator)

CP: (house_id, user_username)

CE: {(house_id) ref House, (user_username) ref User}

StockItemStorage(house_id, stockItem_sku, storage_id, stockItemStorage_quantity)

CP: (house_id, stockItem_sku, storage_id)

CE: {(house_id, stockItem_sku) ref StockItem, (house_id, storage_id) ref Storage}

StockItemMovement(house_id, stockItem_sku, storage_id, stockItemMovement_type, stockItemMovement_dateTime)

CP: (house_id, stockItem_sku, storage_id, stockItemMovement_type, stockItemMovement_dateTime)

CE: {(house_id, stockItem_sku) ref StockItem, (house_id, storage_id) ref Storage}

HouseAllergy(house_id, allergy_allergen, houseAllergy_alergicsNum)

CP: (house_id, allergy_allergen)

CE: {(house_id) ref House, (allergy_allergen) ref Allergy}

ListProduct(house_id, list_id, category_id, product_id, listProduct_brand, listProduct_quantity)

CP: (house_id, list_id, category_id, product_id)

CE: {(house_id, list_id) ref List, (category_id, product_id) ref Product}

StockItemAllergy(house_id, stockItem_sku, allergy_allergen)

CP: (house_id, stockItem_sku, allergy_allergen)

CE: {(house_id, stockItem_sku) ref StockItem, (allergy_allergen) ref Allergy}

Date(date_date)

CP: (date_date)

ExpirationDate(house_id, stockItem_sku, date_date)

CP: (house_id, stockItem_sku, date_date)

CE: {(house_id, stockItem_sku) ref StockItem, (date_date) ref Date}

Restrições de Integridade

RI1: house_id é auto-incrementado;

RI2: house_name é uma cadeia de caracteres obrigatória de comprimento máximo 35, pode incluir letras, números, pontos finais e underscores;

RI3: house_characteristics é um objeto JSON obrigatório;

RI4: user_username é uma cadeia de caracteres obrigatória de comprimento máximo 30, pode incluir letras, números, pontos finais e underscores;

RI5: user_email é uma cadeia de caracteres obrigatória de comprimento máximo 254, pode incluir letras, números, pontos finais, underscores e um arroba;

RI6: user_age é um número inteiro obrigatório contido no intervalo [0, 150];

RI7: user_name é uma cadeia de caracteres obrigatória de comprimento máximo 70, sendo apenas composta por letras;

RI8: user_password é uma cadeia de caracteres obrigatória de comprimento máximo 50, pode incluir letras, números e caracteres especiais;

RI9: allergy_allergen é uma cadeia de caracteres obrigatória de comprimento máximo 75;

RI10: recipe_id é auto-incrementado;

RI11: recipe_name é uma cadeia de caracteres obrigatória de comprimento máximo 35, pode incluir letras, números, pontos finais e underscores;

RI12: recipe_difficulty é opcional e pode tomar um dos valores [‘easy’, ‘average’, ‘difficult’];

RI13: recipe_time é um número inteiro opcional superior a 0;

RI14: recipe_servings é um número inteiro opcional superior a 0;

RI15: recipe_cuisine é uma cadeia de caracteres opcional de comprimento máximo 35;

RI16: recipe_dishType é uma cadeia de caracteres opcional de comprimento máximo 35;

RI17: recipe_type é obrigatório e tem de tomar um dos valores [‘system, ‘user];

RI18: list_id é único dentro da casa;

RI19: list_name é uma cadeia de caracteres obrigatória de comprimento máximo 35, pode incluir letras, números, pontos e underscores;

RI20: list_type é obrigatório e tem de tomar um dos valores [‘system, ‘user];

RI21: category_id é auto-incrementado;

RI22: category_name é uma cadeia de caracteres obrigatória de comprimento máximo 35, sendo apenas composta por letras;

RI23: product_id é único dentro da categoria;

RI24: product_name é uma cadeia de caracteres obrigatória de comprimento máximo 35, sendo apenas composta por letras;

RI25: product_edible é obrigatório e tem de tomar um dos valores ['true', 'false'];

RI26: product_shelfLife é um número inteiro obrigatório superior a 0;

RI27: product_shelfLifeTimeUnit é obrigatório e tem de tomar um dos valores [‘day’, ‘week’, ‘month’, ‘year’];

RI28: stockItem_sku é uma cadeia de caracteres obrigatória de comprimento máximo 128, gerada pela composição de category_id, product_id, stockItem_brand, stockItem_segment e stockItem_variety;

RI29: stockItem_brand é uma cadeia de caracteres obrigatória de comprimento máximo 35;

RI30: stockItem_segment é uma cadeia de caracteres obrigatória de comprimento máximo 35;

RI31: stockItem_variety é uma cadeia de caracteres obrigatória de comprimento máximo 35;

RI32: stockItem_quantity é um número inteiro obrigatório superior a 0;

RI33: stockItem_segmentUnit é obrigatório e tem de tomar um dos valores [‘kg’, ‘dag’, ‘hg’, ‘g’, ‘dg’, ‘cg’, ‘mg’, ‘kl’, ‘hl’, ‘dal’, ‘l’, ‘dl’, ‘cl’, ‘ml’, ‘oz’, ‘lb’, ‘pt’, ‘fl oz’, ‘units’];

RI34: stockItem_conservationStorage é uma cadeia de caracteres obrigatório de comprimento máximo 128;

RI35: ingredient_quantity é um número inteiro obrigatório superior a 0;

RI36: ingredient_quantityUnit é obrigatório e tem de tomar um dos valores [‘kg’, ‘dag’, ‘hg’, ‘g’, ‘dg’, ‘cg’, ‘mg’, ‘kl’, ‘hl’, ‘dal’, ‘l’, ‘dl’, ‘cl’, ‘ml’, ‘oz’, ‘lb’, ‘pt’, ‘fl oz’, ‘units’];

RI37: storage_id é único dentro da casa;

RI38: storage_name é uma cadeia de caracteres de comprimento máximo 35;

3.1.3 Domínio dos Atributos

Tabela 3.1: 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 3.2: 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 3.3: 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 3.4: 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 3.5: 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 3.6: 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 3.7: 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 3.8: 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 3.9: 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 3.10: 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 3.11: 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 3.12: 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 3.13: 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 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 3.14: 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 3.15: 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 3.16: 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 3.17: 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 3.18: 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 3.19: 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 3.20: 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 3.21: 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 3.22: Domínio dos Atributos da Entidade Date.

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

Tabela 3.23: 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

3.2 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.

3.2.1 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.

3.3 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).

3.3.1 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 outro tipos de objeto usados para representar as entidades recebidas e enviadas para o exterior.

3.4 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.

3.4.1 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 4

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.

4.1 Planeamento

Na Figura 4.1 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 4.1, 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 dividas 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.

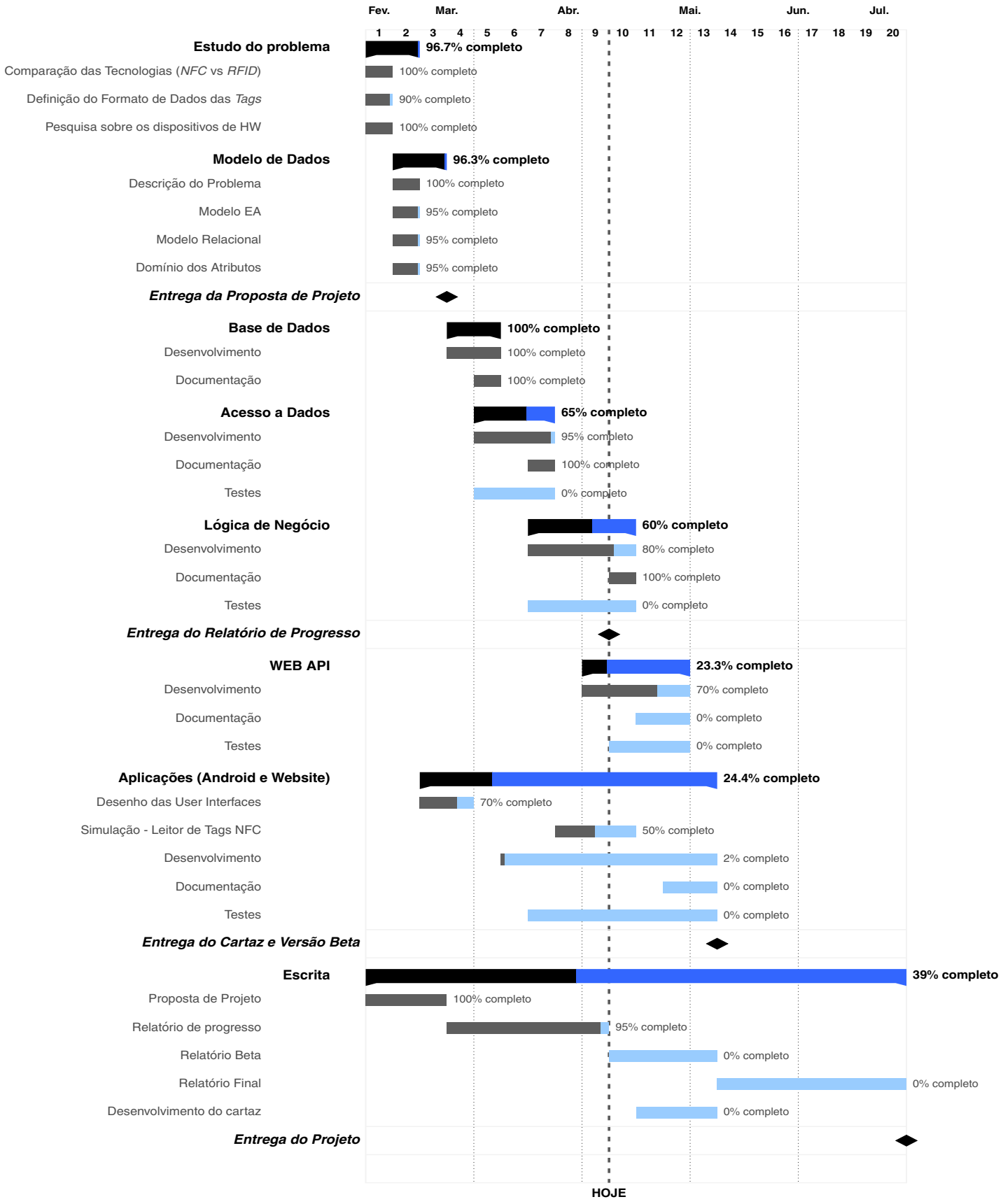


Figura 4.1: Planeamento

4.2 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.

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

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

[3] 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].

[4] 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].

[5] 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].

[6] 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 [1].

***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 [2].

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 [3] e [4].

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 [5].

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 [6]

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 [5].

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 [5].

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