UNIVERSIDADE FEDERAL DO PARANÁ
LEANDRO ANDRADE OLIVEIRA
REGIS ANDRE GABOARDI

CFSTORE – RECOMENDAÇÃO DE PRODUTOS

CURITIBA
2016

LEANDRO ANDRADE OLIVEIRA
REGIS ANDRE GABOARDI

CFSTORE – RECOMENDAÇÃO DE PRODUTOS

Trabalho de conclusão de curso proposto para obtenção do nível superior em Tecnologia em Análise e Desenvolvimento de Sistemas, do Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná.

Orientador: Prof. Dr. Rafael R. Wandresen.

CURITIBA
2016

It was the best of times,
It was the worst of times,
It was the age of wisdom,
It was the age of foolishness…

Charles Dickens

RESUMO

O *e-commerce* já é tradicional no cotidiano brasileiro e não há dúvida de que a maior porção da população já teve contato e/ou já fez compras *online*. A concorrência entre o virtual e o físico só aumenta. O virtual, porém, toma grande vantagem sobre o real, pois pode oferecer inúmeros serviços, desde a redução de impostos até a entrega na comodidade do lar. O e-commerce é utilizado em curta e larga distância, em qualquer tipo de plataforma, e assim se faz presente na vida das pessoas mesmo que estas não percebam. Com esta popularidade em mente, empregam-se os algoritmos de recomendação de produtos: ferramentas que analisam o comportamento de um cliente, comparando este comportamento com uma base empírica, para decidir quais produtos este cliente está procurando, para então dar destaque, ou até mesmo oferecê-los de forma direta ao cliente. Através do estudo de ferramentas de recomendação deu-se a escolha de uma, que é aplicada neste projeto; a segunda teve seu desenvolvimento dentro da UFPR. A criação de duas aplicações semelhantes é importante para tornar visível a comparação da ferramenta escolhida com o algoritmo desenvolvido pelos professores da instituição. As aplicações foram criadas seguindo conceitos de *scrum* e *extreme programming*, com atenção às práticas da engenharia de *software* e outras boas práticas presentes ao longo do curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

Palavras-Chave: E-commerce; Compras; Online; Algoritmo de recomendação; Negócios; *SCRUM*; *Extreme Programming*; Engenharia de Software; Comparação.

ABSTRACT

It’s known for a fact that *e-commerce* is traditional into Brazilian’s routine, and there's no doubt that the bigger share of people already had contact with it and/or already shopped *online*. The concurrency between both virtual and real stores increases every day. The virtual though, has a healthy advantage against its rival: it can offer better deals from lowering taxes’ weight to delivering products at the buyer’s doorstep. the *e-commerce* shortens distances and does not entangle itself to a given platform, thus being part of peoples’ lives even when they don’t realize. With all this popularity in mind, the product recommendation algorithms appeared: tools that analyze costumer’s behavior by matching his actions against an empirical base, so it can predict what the costumer is looking for and make any given product more visible or even offer the product in a straighter way. Through the study of such tools of recommendation one was picked and applied in this project; the second were developed inside the university. The creation of two similar applications comes as a way of seeing the differences between the chosen tool and the UFPR professors developed algorithm. Both applications were built following scrum and extreme programming concepts, with and eye in software engineering practices and other good practices taught through the Technology in System Analysis and Development course.

Keywords: Recommendation algorithm; E-commerce; Purchase; Online; Business; SCRUM; Extreme Programming; Software Engineering; Comparison.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API	- Application Programming Interface
CRUD	- Create, Read, Update, Delete (Criar, Ler, Atualizar, Remover)
HTML	- Hyper Text Markup Language
IDE	- Integrated Development Environment
JSP	- JavaServer Pages
POC	- Prova de conceitos
SEPT	- Setor de Educação Profissional e Tecnológica
TADS	- Tecnologia em Análise e Desenvolvimento de Sistemas
UFPR	- Universidade Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

O *e-commerce* já é tradicional no cotidiano brasileiro, não há dúvida de que a maior porção da população já teve contato e/ou já fez compras *online*. A concorrência entre o virtual e o físico só aumenta. Com esta popularidade em mente, empregam-se os algoritmos de recomendação de produtos: ferramentas que analisam o comportamento de um cliente, comparando este comportamento com uma base empírica, para decidir quais produtos este cliente está procurando, para então dar destaque, ou até mesmo oferece-los de forma direta ao cliente.

Nos capítulos seguintes entramos com mais profundidade nos conceitos considerados na produção deste trabalho. Começando pela fundamentação teórica, onde encontram-se mais informações sobre algoritmos e técnicas de recomendação e engenharia de software. No capítulo seguinte (Capítulo 3), estão descritas as metodologias e os pensamentos utilizados para a definição das aplicações como um todo, desde suas interfaces no navegador até a interação com os algoritmos de recomendação. E nos dois últimos capítulos é feita a apresentação do sistema e das considerações finais, respectivamente.

1.1 JUSTIFICATIVA

A CFStore, loja online criada para a simulação dos algoritmos, surgiu como um dos produtos de uma demanda real, proposta inicialmente pelo grupo SOLID (Soluções Computacionais para Análise Inteligente de Dados) do Setor de Educação Profissional e Tecnológica (SEPT), da UFPR (www.tads.ufpr.br). Este grupo tem uma parceria com a Lojas KD, é uma empresa do ramo moveleiro (www.lojaskd.com.br).

O SOLID é um grupo de pesquisa formado por alunos e professores da UFPR, que busca desenvolver soluções relacionadas para os algoritmos de recomendação usados atualmente na loja online das Lojas KD. Essa parceria surgiu do interesse das Lojas KD em melhorar os algoritmos de recomendação de produtos usados em sua loja online.

Neste projeto buscou-se a comparação da aplicação de dois algoritmos de recomendação. Um algoritmo que foi desenvolvido dentro da UFPR, pelo grupo de pesquisa SOLID. O outro foi escolhido a partir do estudo sobre algoritmos de recomendação e da disponibilidade de suas bibliotecas na internet. Esta comparação tornou-se visível com a criação de duas aplicações semelhantes em Java, que funcionam em plataforma Web e seguem uma série de padrões reforçados pelo curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS).

1.2 OBJETIVOS

Esta seção apresenta os objetivos deste trabalho, e mapeia as conquistas que devem ser alcançadas ao final do projeto.

1.2.1 Objetivo Geral

O objetivo deste trabalho, como um todo, é desenvolver um site que simule a venda de móveis e a recomendação de produtos, dando atenção às características da UML, arquiteturas de desenvolvimento, gerenciamento de projetos, e tantas outras práticas do desenvolvimento de software.

O foco central deste trabalho é testar dois algoritmos de recomendação baseados em item. O primeiro é uma recomendação baseada em item usando o Apache Mahout como ferramenta e o segundo é a recomendação de itens usando a técnica do Slope-one. Para dar visibilidade às diferenças dos resultados de cada algoritmo, uma aplicação foi desenvolvida, e em seu estágio final foi replicada para que o segundo algoritmo proposto seja usado. As aplicações criadas são baseadas na loja já existente.

1.2.2 Objetivos Específicos

a) Estudo de algoritmos de recomendação e bibliotecas de programação;
b) Escolha do algoritmo de recomendação;
c) Levantamento de requisitos do sistema e analise da possibilidade de construção destes;
d) Definição de linguagens e ferramentas que são utilizadas ao longo do projeto;
e) Modelagem do sistema, levantando diagramas tais como:
   - Diagrama de arquitetura
   - Diagrama de Casos de Uso
   - Diagrama de Entidade e Relacionamento (DER)
   - Diagrama de Classes
   - Diagrama de Fluxo de Dados (DFD)
f) Criação do *schema* do banco de dados que abriga todos os dados fornecidos para as recomendações de produtos;
g) Criação da *CFStore*, a *Loja do Filtro Colaborativo*;
h) Utilização dos algoritmos de recomendação escolhidos.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve os conceitos e conhecimentos necessários para o entendimento do projeto, assim como foram para sua construção.

2.1 SISTEMAS DE RECOMENDAÇÃO

Os sistemas de recomendação têm como finalidade principal sugerir produtos ou itens, fazendo previsões das preferências de um usuário. É a combinação de várias técnicas computacionais e estatísticas para selecionar itens que possam ser do interesse de um usuário. Sua aplicação é feita nas mais diversas áreas, tendo como principais exemplos sites de e-commerce como Amazon.com e Google e sites de entretenimento como *Spotify* e *Netflix*.

2.1.1 Técnicas de Recomendação

Segundo Ricci**,** Rokach**,** Shapira e Kantor (2011), as principais técnicas de recomendação são:

a) Baseada no conteúdo (Content-based): o sistema aprende a recomendar itens que são baseados em outros que o usuário gostou no passado;
b) Filtragem colaborativa (Collaborative filtering): Nesta técnica, a similaridade de preferências entre dois usuários é calculada baseada no histórico de avaliações dadas por esses usuários. É considerada umas das técnicas mais utilizadas em sistemas de recomendação;
c) Demográfica: esse tipo de recomendação é baseado no perfil demográfico do usuário;
d) Baseado em conhecimento (Knowledge-based): nesta técnica, a avaliação dada pelo usuário pode ser diretamente interpretada como grau de utilidade da recomendação para o usuário;

e) Baseado em comunidade (Community-based): esse tipo de recomendação baseia-se na preferência dos amigos dos usuários;
f) Sistemas de recomendação híbridos: é a combinação das técnicas mencionadas acima.

2.1.2 Recomendações baseadas em Filtro Colaborativo

A técnica de recomendação colaborativa resolve as limitações da recomendação baseada em conteúdo. Itens cujo conteúdo não está disponível ou é difícil de obtê-lo podem ser recomendados a um usuário mesmo assim, através do feedback de outros usuários (Ricci, Rokach, Shapira e Kantor ,2011).

2.1.3 Recomendação por item

Enquanto a recomendação por usuário tem como objetivo encontrar gostos parecidos entre usuários diferentes para poder fazer recomendação, a recomendação por item faz uma indicação de um novo item baseado em um escolhido, não levando em consideração o perfil do usuário. Um exemplo prático dessa técnica pode ser dado através do seguinte raciocínio: um cliente de uma loja de móveis que compra um armário branco, possivelmente vai ter interesse em uma cômoda branca também, baseando-se no histórico de vendas desta loja.

Para este trabalho estamos propondo o teste de 2 tipos de recomendações pode item : *Generic Item Based* utilizando o Apache Mahout e *Slope-one*.

2.1.4 Recomendação Slope-one

Uma outra técnica de recomendação de filtro colaborativo que pode ser aplicada é o slope-one. Segundo Owen, Anil, Dunning e Friedman (2012), esta técnica

consiste em estimar a preferência por novos itens baseando-se na diferença em relação à média de notas dadas pelo usuário para outros itens.

O slope-one exige um grande volume de pré-processamento para calcular essa diferença entre as notas (diffs) como citados acima e demanda grande espaço de armazenamento dessas informações. Por outro lado, após esse pré-processamento, as recomendações e atualizações são mais rápidas em relação à outras técnicas de recomendação.

3 METODOLOGIAS

Neste capítulo são descritas as metodologias adotadas para o desenvolvimento do sistema.

3.1 UNIFIED MODELING LANGUAGE UML

O conhecimento da *UML* é imprescindível para o entendimento dos diagramas deste projeto. Como indicado por Dutoid e Bruegge (2009, p.31), “o objetivo da UML é prover uma notação padrão que pode ser usada por todos os métodos orientados a objetos e para selecionar e integrar os melhores elementos das notações precursoras”.

Neste projeto a UML é utilizada a técnica do *sketch* para se elaborar e discutir diagramas. O sketch tem foco na seletividade, e progressivamente (*forward sketching*), é usado para isolar uma funcionalidade que será desenvolvida em breve, e discutir com um time suas particularidades com o intuito de facilitar a comunicação e o levantamento de alternativas sobre o que está prestes a ser feito. Cada sessão que usa um sketch pode ser bem curta, considerando a complexidade do objeto em discussão, mapeando o desenvolvimento das próximas horas, ou dos próximos dias. (Fowler, 2003)

Identifica-se neste projeto, também, o estilo incremental, descrito por Fowler (2003, p.20): “O estilo iterativo (incremental) divide o projeto em *subsets* por

funcionalidade. Um ano então, é dividido em iterações de três meses. Cada iteração tem todo o ciclo de vida do software; análise, *design*, codificação e testes".

É importante notar que as mesmas técnicas da UML podem levar diferentes nomes, segundo Fowler (2003, p.21): "O desenvolvimento iterativo leva vários nomes: incremental, espiral, evolucionário, *jacuzzi* são alguns que vem à mente. Várias pessoas os distinguem, mas estas distinções não são amplamente aceitas nem tão importantes quanto a separação entre incremental e cascata".

Diagramas de classes são usados para ilustrar a estrutura do sistema. Classes são abstrações que especificam a estrutura normal e os comportamentos de conjuntos de objetos. Objetos são instâncias de classes, são criados, modificados e destruídos durante a execução do sistema. Um objeto possui estados que incluem os valores de seus atributos e sua ligação com outros objetos (Dutoid & Bruegge, 2009).

Outra ferramenta da UML amplamente utilizada neste projeto, são diagramas de caso de uso. Casos de uso, histórias do usuário no *extreme programming*, são detalhamentos tanto em forma gráfica quanto textual, sobre como o usuário interage com o sistema, como convicção de Fowler (2003, p.104): "Casos de uso proveem uma narrativa de como o usuário usa o sistema". Casos de uso são escritos em linguagem natural. Isso permite uma comunicação mais fluida entre desenvolvedores, clientes e usuários, que podem não ter conhecimento sobre as notações da UML. O uso da linguagem natural permite que participantes de outras disciplinas entendam os requisitos do sistema. A linguagem natural também possibilita que os desenvolvedores capturem requisitos especiais, que não seriam identificados por diagramas (Dutoid & Bruegge, 2009).

3.2 STORED PROCEDURES

Stored procedure (SP) é a ferramenta escolhida para que o código deste projeto fique mais enxuto e performático. Um SP é um programa armazenado no banco de dados que permite flexibilidade na análise e desenvolvimento do sistema pois provê independência entre os componentes.

O uso de Stored Procedures, como citado por Harrison e Feuerstein (2006, p4 e p5): "pode proporcionar um banco de dados mais seguro.", "Programas

armazenados podem diminuir o tráfico de dados, pois podem processar os dados dentro do servidor, sem precisar transferir dados pela rede.” e “O uso de programas armazenados pode, em alguns casos, melhorar a portabilidade da aplicação.”.

Stored Procedures agilizam o processamento de dados pois permitem que este processamento ocorra no servidor. Os dados oriundos do usuário precisam ser coletados externamente, mas uma vez tratados e enviados ao programa, qualquer função pode ser concluída. Sem que várias requisições sejam feitas entre o back-end do sistema e o banco de dados.

3.3 MATERIAIS E METODOLOGIAS

Nesta seção, descrevemos os materiais de metodologia e os recursos de software utilizados para a execução do projeto.

3.3.1 Java e Web

Para o desenvolvimento deste projeto optou-se pela utilização da linguagem de programação Java, juntamente com a IDE *NetBeans* e servidor *Tomcat*. Esta decisão foi amplamente influenciada pela forte base que o curso de Tecnologia em Análise e Desenvolvimento de Sistemas dá ao Java, sendo também a linguagem de maior afinidade por parte dos integrantes da equipe.

3.3.2 Modelo MVC

Neste projeto, utilizou-se do *design pattern* MVC – *Model-View-Controller*. É um padrão de desenvolvimento amplamente utilizado no mercado de trabalho, e também tem seu uso encorajado ao longo do curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

O MVC provê três camadas independentes para a aplicação, e a intenção é que estas sejam independentes entre si, como descrito por Steelman e Murach (2008, p.208): “Quando o padrão MVC é utilizado tentamos manter o model, view e controller tão independentes quanto possível. Isso facilita a manutenção da aplicação no futuro.”. Assim, o sistema será de fácil entendimento para terceiros, pois o baixo acoplamento entre as camadas permite que novas funcionalidades tenham um “lugar” determinado no código, e agiliza o processo de manutenção uma vez que as alterações são rapidamente identificadas.

No padrão MVC, a camada de modelo (*model*) contém classes que representam os atributos de negócio e os métodos necessários para os processos de negócio do sistema. A camada de visão (*view*) define como o sistema será representado ao usuário; por ser comum o envio de *HTML* para o navegador, a camada de visão geralmente contém arquivos HTML ou JSP. A camada de controle (*controller*) gerencia o fluxo da aplicação identificando necessidades das requisições recebidas e então executando logicas do sistema, como modificar um objeto da camada de modelo e exibi-lo na camada de visão novamente (Murach & Steelman, 2008).

3.3.3 Metodologias Ágeis

No desenvolvimento deste projeto adotou-se uma mistura de *scrum* com *extreme programming*. Dividindo o desenvolvimento em ciclos de 10 dais, com a entrega de uma porção do sistema ao final de cada ciclo.

Segundo concepção de Roman Pichler (2010, p.99): “O Scrum favorece um fluxo suave e continuo de trabalho(...). Confiança é mais importante que ambição e possibilita previsões mais precisas.”. Com este pensamento, definiu-se que tarefas mais complicadas e que inspiravam menos confiança a equipe, seriam desenvolvidas primeiro e que as tarefas familiares aos membros do time seriam desenvolvidas mais tarde, ao longo do projeto.

O Extreme Programming (XP) foi adotado por fornecer uma base de pensamento simples e que se encaixa bem com o Scrum; O XP é uma “aproximação” de desenvolvimento que defende entregas do sistema em pequenos ciclos de tempo

(*timeboxing*) e, portanto, não se preocupa com a definição de todos os requisitos adiantadamente, isto é normalmente feito no modelo de cascata. (COBB, 2011).

Dos conceitos do XP identificados durante o projeto podem-se citar:

a) Propriedade coletiva do código: cada membro do time é dono e tem liberdade para corrigir bugs, melhorar o design, ou refatorar o código;
b) Integração contínua: cada ciclo tem várias integrações de código em que *commits* são feitos com frequência para evitar duplicação na escritura do código;
c) Melhora do processo: Cada ciclo que altera uma funcionalidade desenvolvida no ciclo anterior, melhora o código existente por meio de uma crítica retrospectiva do processo e para fazer ajustes no processo se necessário.

Este projeto utilizou Casos de Uso no lugar das Histórias do usuário pois os sistemas não possuem um usuário definido, os casos de uso foram levantados pela observação de outros sites, brainstorming e “pensamento especulativo”.

3.3.4 Bootstrap

Boostrap é um framework para desenvolvimento de layouts para sites baseados em HTML. Ele usa a combinação de CSS e JavaScript para criar elementos HTML com design e ações personalizados, facilitando e encurtando o tempo para desenvolver páginas Web através de classes CSS já implementadas na ferramenta.

3.4 MATERIAIS

Nesta seção, serão listados os materiais utilizados para o desenvolvimento do projeto como um todo. Todas as etapas podem ser reproduzidas com estas ferramentas em um meio virtual, salvo os *hardwares* descritos.

3.4.1 Hardwares

Proprietário: Regis Andre Gaboardi

- Marca: *Dell*
- Sistema Operacional: *Windows* 10 x64
- Memória RAM: 8GB
- Processador: Intel Core i5-3210M @ 2.5GHz
- Espaço de armazenamento: 1TB
- Placa de vídeo: AMD RADEON 7730M 2GB

Proprietário: Leandro Andrade Oliveira

- Marca: *Dell*
- Sistema Operacional: *Windows* 10 x64
- Memória RAM: 8GB
- Processador: Intel Core i7 – 5500U @ 2.4GHz
- Espaço de armazenamento: 1TB
- Placa de vídeo: NVIDIA 920M 4GB DDR3

3.4.2 Softwares

Este tópico descreve os softwares e API’s utilizados no desenvolvimento do sistema.

3.4.2.1 Apache Mahout

O *Mahout* é uma biblioteca em Java de *machine learning* criada pelo Apache. É muito utilizado em sistemas de recomendações, *clustering* e classificação. O objetivo do Mahout é ser uma escolha quando for necessária uma ferramenta para

processar uma machine learning utilizando uma quantidade de dados muito grande. (Owen, Anil, Dunning e Friedman, 2012, pg. 1). O Mahout facilita o trabalho de criar uma ferramenta de filtro colaborativo. Neste sistema, ele foi usado para criar uma recomendação baseada em item.

3.4.2.2 NetBeans IDE

O NetBeans foi a IDE escolhida para uso durante a codificação das aplicações desenvolvidas. É uma IDE simples que possibilita algumas facilidades durante a construção de sistemas, como suporte ao TDD e geração automatizada de código, os getters e setters do pacote de modelo foram gerados automaticamente, por exemplo.

É uma ferramenta que permite o desenvolvimento rápido e fácil de aplicações *desktop*, *mobile* e *web*, em várias linguagens, dentre elas o Java. Também dá suporte ao *HTML*, *JavaScript* e *CSS*. A ferramenta é grátis e *open source*, além de ter uma comunidade de usuários pelo mundo (www.netbeans.org).

3.4.2.3 MySQL Workbench

O MySQL foi o banco de dados escolhido para uso neste projeto, é um dos bancos de dados mais conhecidos e populares no mundo da programação. Além da familiaridade dos membros com a ferramenta, podem-se citar outras razões para a escolha, que são: Disponibilidade, Performance, Suporte, Segurança e Portabilidade.

3.4.2.4 Astah Community

O Astah Community é uma ferramenta para modelagem de dados. Foi escolhida pela equipe para o desenho da grande maioria dos diagramas presentes neste documento, salvo os de banco de dados, que foram feitos explorando a

facilidade garantida pelo MySQL Workbench. O Astah é uma ferramenta que oferece um serviço completo, porém pago. Oferece também uma assinatura para estudantes, que não tem custo e que dá suporte a vários tipos de diagramas necessários para o desenvolvimento de um sistema. O Astah oferece também amplo suporte ao usuário, várias versões especializadas em diferentes âmbitos da modelagem de dados e até mesmo guias de modelagem. Tudo isso pode ser encontrado em www.astah.net.

3.5 REQUISITOS

Requisitos são determinações de como o sistema funciona e se comporta. Foram definidos na fase inicial do desenvolvimento por meio de *brainstorming*, o grupo reúne-se para levantar ideias por meio da discussão sobre a ideia que cada integrante tem do sistema.

3.5.1 Requisitos funcionais

Os requisitos funcionais definem como o sistema deve se comportar e as funcionalidades que deve oferecer. A descrição de cada requisito funcional encontra-se no apêndice J.

3.5.2 Requisitos não funcionais

Os requisitos não funcionais definem propriedades do sistema em si, são requisitos que o usuário final não tem controle, mas que influenciam diretamente o sistema como um todo. A descrição de cada requisito não funcional encontra-se no apêndice J.

3.6 PLANO DE ATIVIDADES

O plano de atividades foi definido na fase inicial da análise, considerando que a equipe tem dois integrantes, e baseado no WBS; apêndice H. Também criado no início do projeto.

3.6.1 Responsabilidades

As responsabilidades de cada integrante da equipe durante o desenvolvimento do projeto estão descritas na tabela de responsabilidades (1).

Atividades	Membro da equipe
Definição do Escopo	Leandro e Regis
Plano de Riscos	Regis
Plano de Atividades	Regis
WBS	Leandro e Regis
POC	Regis
Levantamento de Requisitos	Leandro e Regis
Modelagem de Casos de Uso	Regis
Protótipos de Interfaces	Leandro
Diagramas UML	Leandro e Regis
Definição da Metodologia	Leandro e Regis
Desenvolvimento da Metodologia	Leandro e Regis
Definição da Linguagem de Programação	Leandro e Regis
Criação do Esquema do Banco	Leandro
Preenchimento da Base de Dados	Leandro
Desenvolvimento do Sistema	Leandro e Regis
Testes Unitários	Leandro e Regis
Correção de falhas	Leandro e Regis
Apresentação do Projeto	Leandro e Regis

TABELA 1 – TABELA DE RESPONSABILIDADES
FONTE: OS AUTORES, 2016

3.6.2 Cronograma

O *cronograma* é necessário, pois é a ferramenta que define datas e prazos para as tarefas do sistema. Este cronograma foi definido com base na separação de *Sprints*, característicos do *scrum*, feito inicialmente no WBS. Cada um dos Sprints

definidos contém um determinado número de funcionalidades, que os membros da equipe julgam ser viável para o espaço de tempo definido em cada Sprint.

Os ciclos definidos para este projeto são de dez dias corridos.

3.7 PLANO DE RISCOS

O plano de riscos teve seu desenvolvimento ao início do projeto, quando os integrantes do grupo discutem os possíveis problemas que podem aparecer ao longo do desenvolvimento. Assim como o levantamento de requisitos, foi projetado a partir da técnica de *brainstorming*.

A tabela do plano de riscos encontra-se no apêndice G deste documento.

3.8 DESENVOLVIMENTO

O desenvolvimento deste projeto foi inteiramente dividido em *Sprints*, ciclos de 10 dias em que, ao final de cada, foi entregue uma nova versão mais completa do que o sistema existente na entrega anterior.

Não está descrito entre os *Sprints* o tempo de estudo e decisão sobre qual metodologia de recomendação seria utilizada. Esta etapa foi iniciada no dia 18 de agosto, e foi dedicada à leitura e pesquisa sobre ferramentas e tecnologias de recomendação disponíveis, antes que qualquer etapa de desenvolvimento tivesse seu início.

O primeiro Sprint de desenvolvimento; que definiu o escopo do projeto como um todo, foi chamado de “Pré Projeto”, e teve como data inicial o dia 02 de outubro. Nesse Sprint definiram-se as primeiras versões de diagramas e requisitos do sistema. Por meio de brainstorming, rascunhos e especulações, foi construída a ideia do sistema. Neste Sprint foram definidas linguagens, banco de dados, arquiteturas, riscos. Também foi iniciada a criação de uma prova de conceitos (POC), que teve sua apresentação no início do Sprint seguinte. A imagem que representa o Pré Projeto encontra-se no apêndice H.

No dia 12 de outubro, deu-se o início do primeiro Sprint de programação. Neste Sprint ocorreu a criação e importação da base de dados. Os dados foram importados da base fornecida pelas Lojas KD. Assim foi possível usar produtos e dados existentes ao longo de todo o projeto, além de possibilitar a avaliação da recomendação de um produto real. Construiu-se também a busca por produtos, que é a segunda funcionalidade principal do sistema, atrás apenas da própria recomendação. Na criação desta busca, verificou-se dificuldade para retornar resultados coerentes em determinadas buscas devido a formo como os nomes dos produtos foram armazenados no banco de dados, contendo informações desde descrição e especificações do produto até o nome do fornecedor. Assim buscas como “Mesa escritório” retornam somente produtos que tem exatamente este trecho no nome. Optou-se por não fazer essa otimização no nome dos produtos pois seria um processo demasiadamente demorado e que traria outras dificuldades para o processo de busca. Neste Sprint ainda foi criada a função de adição de itens ao carrinho. O Sprint teve final previsto para o mesmo dia que iniciou o Sprint seguinte. A imagem que representa o Sprint 01 encontra-se no apêndice H.

O segundo Sprint começou no dia 22 de outubro, e teve fortíssimo esforço para que as recomendações funcionassem com a base existente. Neste Sprint foi inserido no sistema o Apache Mahout, ferramenta que foi escolhida pelos membros da equipe para gerar uma das recomendações que são oferecidas pelo sistema. Devido à complexidade e das dificuldades encontradas para se gerar recomendações, esta foi a única funcionalidade desenvolvida neste ciclo. Também foram efetuados testes unitários no Sprint. A imagem representante do Sprint 02 encontra-se no apêndice H.

No Sprint 03, que se iniciou no dia primeiro de novembro. Foram desenvolvidas as outras funcionalidades esperadas para o lado cliente do sistema. Funcionalidades relacionadas ao cadastro do cliente, controle de permissões e acesso por meio de *login*, alteração de endereço do cliente cadastrado e finalização de compra. Estes requisitos formularam-se com o intuito de dar um ar de completude ao sistema, que realmente é apenas um simulador da loja original. A imagem que representa este Sprint encontra-se no apêndice H.

O quarto Sprint teve início no dia 11 de outubro. Neste Sprint foi feita a replicação da aplicação com a substituição do algoritmo de recomendação (SOLID). Assim como o segundo, este Sprint teve grande esforço para sua conclusão. Houve

dificuldade de integração, pois, neste trabalho, optou-se por utilizar banco de dados MySQL e o algoritmo de recomendações do SOLID faz uso, originalmente, do banco de dados *PostgreSQL*.

Para que o sistema pensado em primeiro lugar consiga “comunicar-se” com o algoritmo do SOLID, foi feita uma transição de dados em duas etapas. Primeiramente os produtos foram todos importados para o MySQL e então foram feitos dois “cruzamentos” de base. Estes cruzamentos foram necessários no primeiro momento em decorrência de uma outra inconsistência das bases de dados. O sistema que usa o Mahout utiliza como identificador dos produtos, o código de cada produto. Já o algoritmo do SOLID utilizou um identificador incremental e sequencial. Após a combinação das bases para identificar qual produto é recomendado em cada página, houve outro cruzamento com a base de imagens, que não foram fornecidas inicialmente pela Lojas KD. Para a recuperação das imagens de cada produto, utilizamos um algoritmo de força bruta, que recuperou e salvou os links de cada imagem presente no site. A imagem que representa este Sprint encontra-se no apêndice H.

O último Sprint foi programado para conter apenas fases de testes e correções. Suas datas inicial e final foram previstas para o dia 21 de novembro e primeiro de dezembro, respectivamente. Por ter sido um Sprint muito trabalhoso, principalmente por que as tarefas desempenhadas demandam de muito processamento e demoram para ser concluídas, o quarto Sprint avançou e fundiu-se com o quinto. Nestas datas já aviam sugestões de correções do próprio SOLID, que foram atendidas em paralelo ao processamento dos dados. O Sprint 5 é representado graficamente no apêndice H.

3.9 CRIAÇÃO DE RECOMENDAÇÕES COM O APACHE MAHOUT

Para a utilização do Apache Mahout, usando um algoritmo de recomendação baseado em item, é necessário executar as ações conforme descritas na figura 1. A figura do fluxo é explicada passo a passo ainda nesta seção.

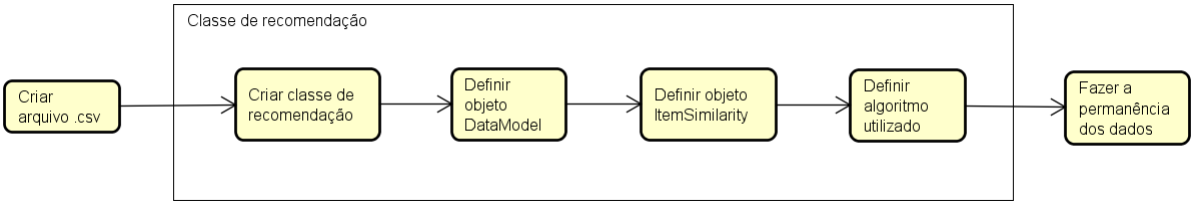

FIGURA 1 - FLUXO DE INSTALAÇÃO DO MAHOUT
FONTE: OS AUTORES, 2016

Para a utilização do Mahout, primeiro é necessário que importe as suas bibliotecas para o projeto.

Tendo as bibliotecas importadas para o projeto, usamos as seguintes etapas para a criação de um sistema de recomendação *item-based*:

a) Criar um *arquivo.csv* com a seguinte estrutura: código do usuário, código do item e avaliação do produto. Todas essas informações foram geradas a partir da tabela de vendas, considerando que um produto vendido receberá a nota máxima de avaliação (5). Os campos do arquivo devem ser separados por “;”.
b) Importar o csv para o projeto;
c) Criar uma classe de recomendação. Essa classe deverá conter os seguintes itens:
   - Um objeto do tipo DataModel referenciando o arquivo csv;

   DataModel dm = new FileDataModel(new File("web/data/cfstore.csv"));
   - Um objeto ItemSimilarity passando como parâmetro o DataModel criado;

   ItemSimilarity is = new LogLikelihoodSimilarity(dm);
   - Um objeto GenericItemBasedRecommender passando os objetos de DataModel e ItemSimilarity como parâmetros;

   GenericItemBasedRecommender recommender = new GenericItemBasedRecommender(dm, is);
   - A quantidade de itens recomendados pode ser ajustada no método mostSimilarItens (long itemID, int howMany) do objeto criado, que passa como parâmetro o id do item e quantas recomendações serão feitas.
d) Salvar a lista de recomendações geradas, em um banco de dados.

4 APRESENTAÇÃO DO SOFTWARE

O software proposto tem por finalidade ilustrar o funcionamento de uma loja de moveis virtual. Simulando o catálogo de produtos de uma loja real e disponibilizando uma ferramenta de recomendação.

4.1 ACESSO AO SISTEMA

Na tela inicial (FIGURA 1), o usuário visualiza no menu superior, as opções de login no sistema, visualizar itens do carrinho e a barra de busca. Ainda é possível que o usuário navegue pelos departamentos utilizando o menu à esquerda ou que visualize mais informações sobre os produtos mostrados nesta página.

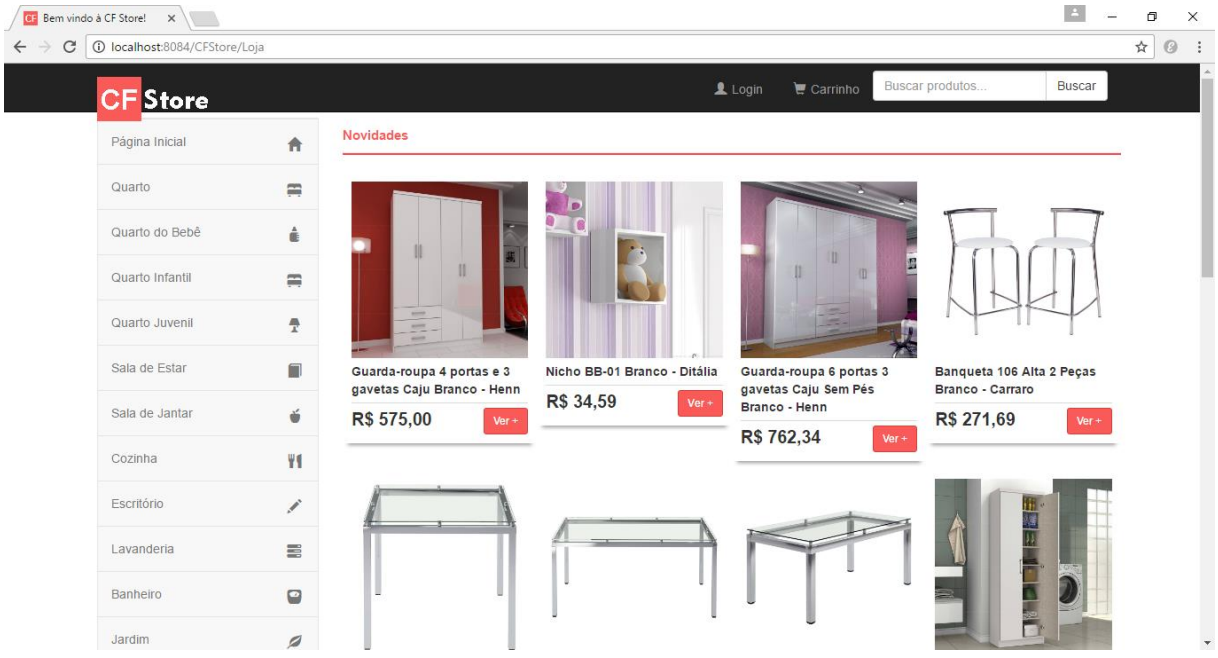

FIGURA 2 – TELA INICIAL DO SISTEMA
FONTE: OS AUTORES, 2016

4.2 BUSCA E VISUALIZAÇÃO DE RESULTADOS

Para realizar a busca de um produto, o usuário deve usar a barra de busca, localizada no canto superior da tela. Essa barra de busca é visível de qualquer tela do sistema. Caso o critério da busca seja atendido, o sistema irá mostrar a tela com os resultados encontrados, conforme figura abaixo (2).

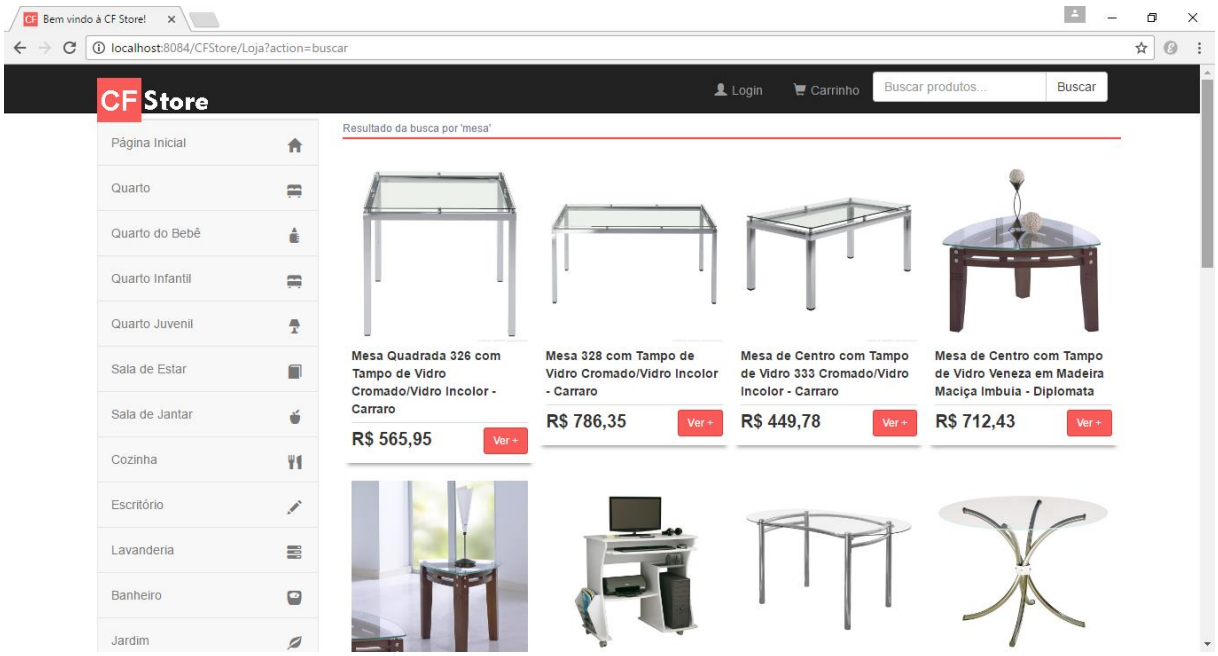

FIGURA 3 – TELA DE BUSCA E RESULTADOS
FONTE: OS AUTORES, 2016

4.3 EXIBIÇÃO DO PRODUTO

A FIGURA 3 apresenta a tela de exibição do produto onde o usuário consegue visualizar detalhes do produto, tendo a opção de adicioná-lo ao seu carrinho. Logo abaixo do produto, são exibidas as recomendações de outros produtos ao usuário.

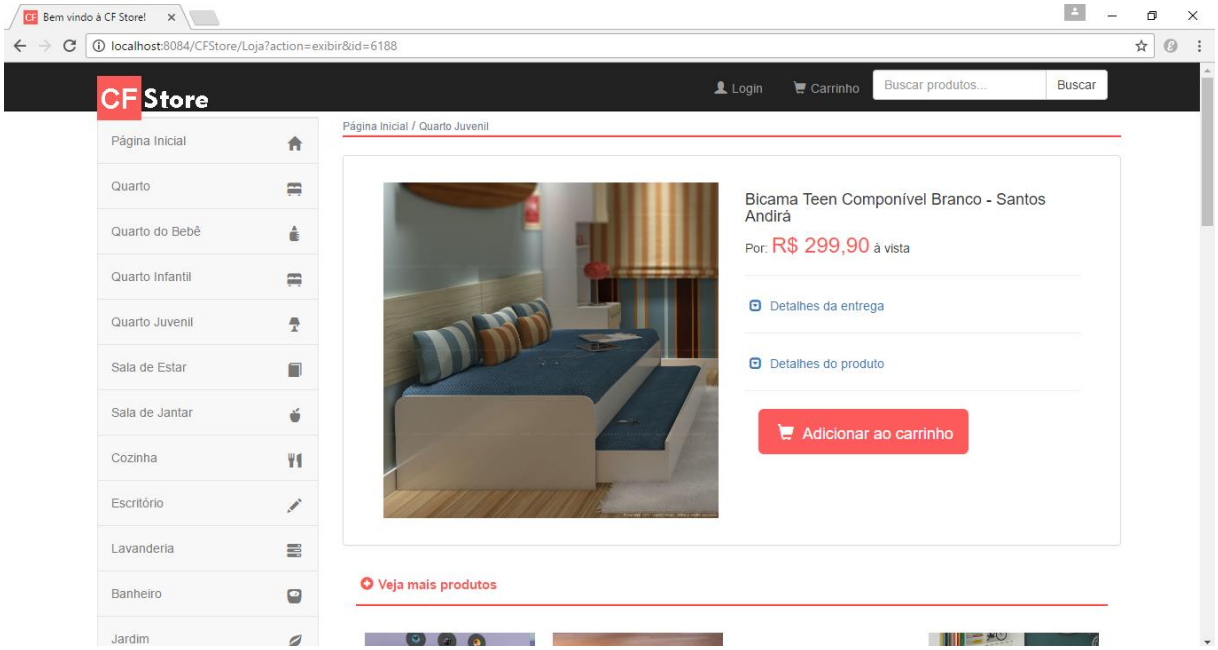

FIGURA 4 – TELA DE EXIBIÇÃO DO PRODUTO
FONTE: OS AUTORES, 2016

4.4 RECOMENDAÇÃO

Ainda na tela de visualização do produto, é possível visualizar as recomendações ao usuário baseada no produto selecionado exibidas logo abaixo da descrição do produto.

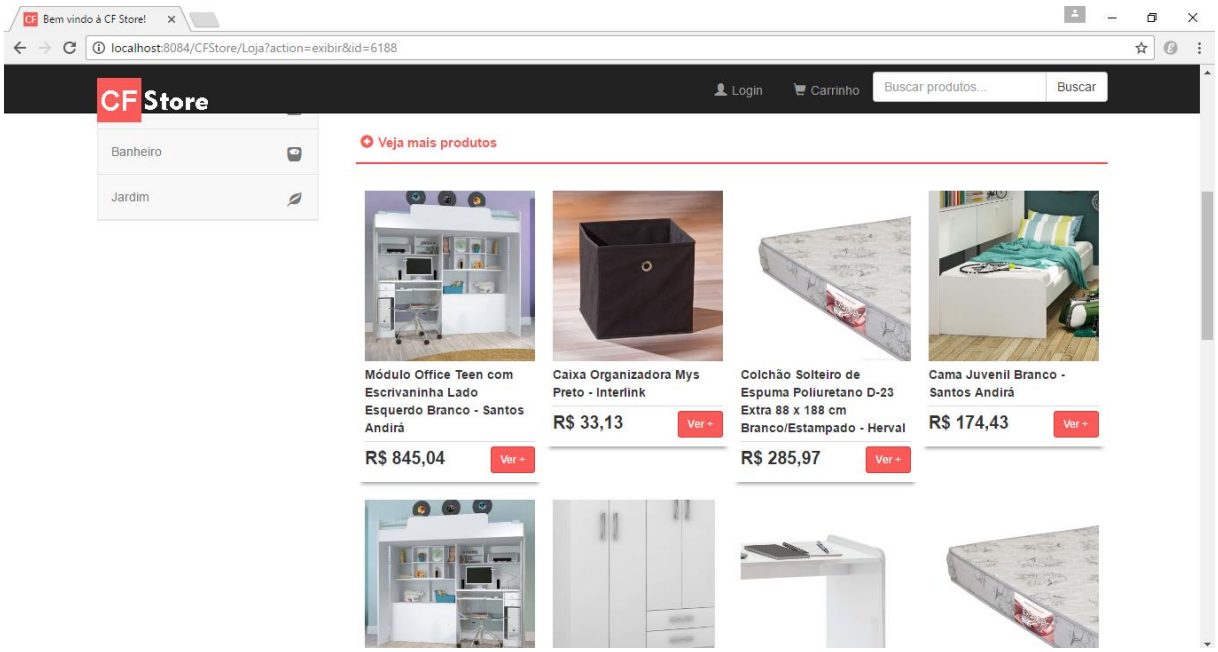

FIGURA 5 – TELA DE EXIBIÇÃO DAS RECOMENDAÇÕES

FONTE: OS AUTORES, 2016

4.5 CARRINHO DE COMPRAS

Esta tela exibe todos os itens adicionados ao carrinho. O usuário poderá excluir um item que foi incluído clicando no botão “Excluir” relacionado ao item. O valor total da compra é exibido ao final da listagem dos itens do carrinho.

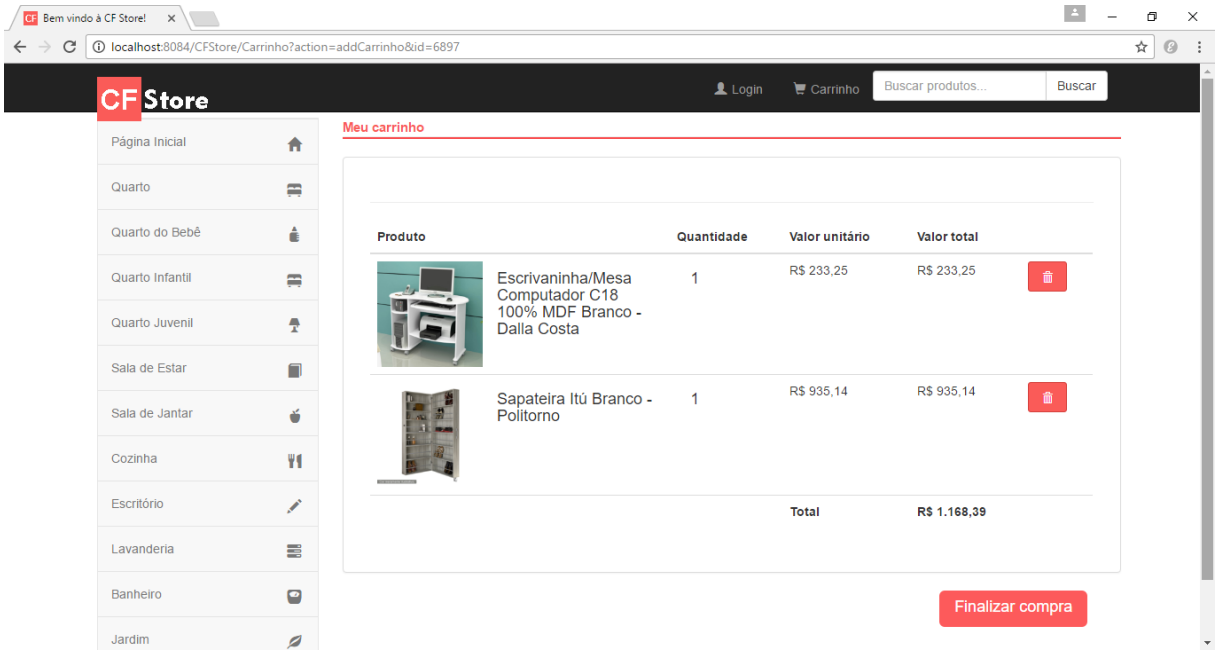

FIGURA 6 – TELA CARRINHO DE COMPRAS DO CLIENTE
FONTE: OS AUTORES (2016)

5 CONSIDERAÇÕES FINAIS

Ao final desse projeto, o sistema desenvolvido foi capaz de demonstrar a utilização de 2 tipos de sistemas de recomendação baseados em item (*item based e slope-one)*. Mostrando-se uma ferramenta importante para auxiliar usuário em sua tomada de decisão na hora de escolher produtos que complementem a sua compra.

5.1 CONCLUSÕES

Ao final do desenvolvimento do projeto, foi possível fazer uma análise dos resultados de cada sistema de recomendação. Esta análise foi feita por meio da comparação direta entre as recomendações geradas para um produto em cada sistema.

Para isso, pesquisamos por um produto no primeiro sistema e gravamos as recomendações geradas. Então no segundo sistema, procuramos pelo mesmo produto para poder comparar as recomendações geradas.

As diferenças nos resultados gerados podem ser conferidas nas figuras 6 e 7 abaixo. Essas figuras apenas um dos exemplos de comparação testados.

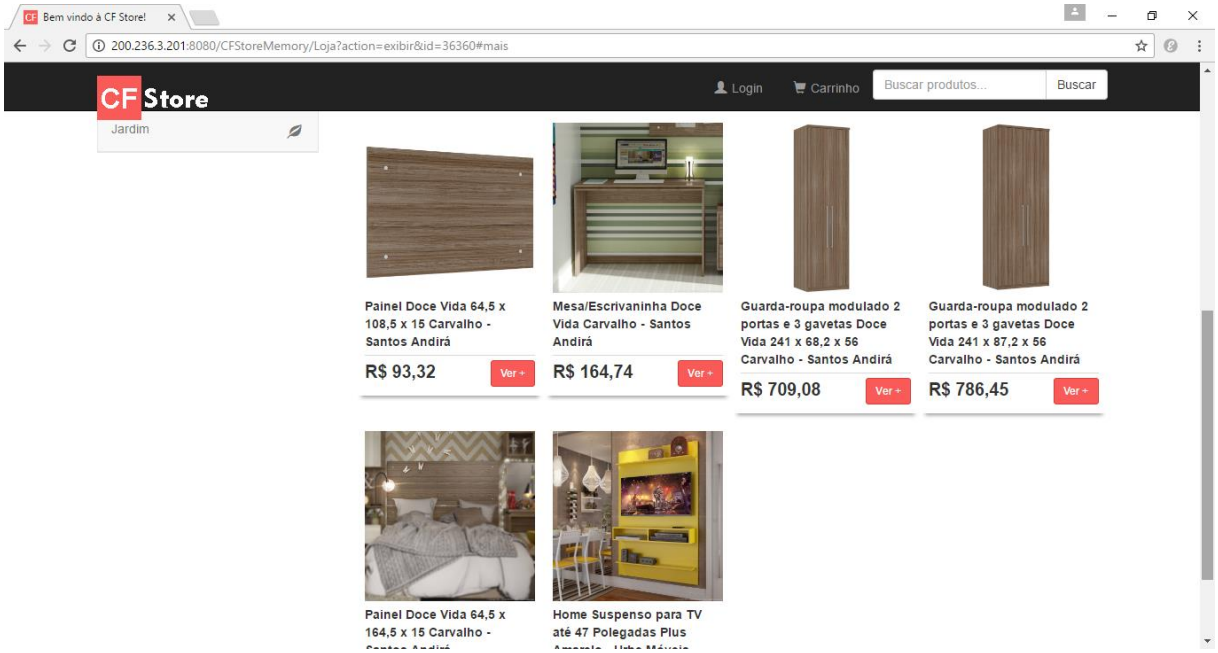

FIGURA 7 – RECOMENDAÇÕES GERADAS PELO ALGORITMO "ITEM BASED"

FONTE: OS AUTORES, 2016

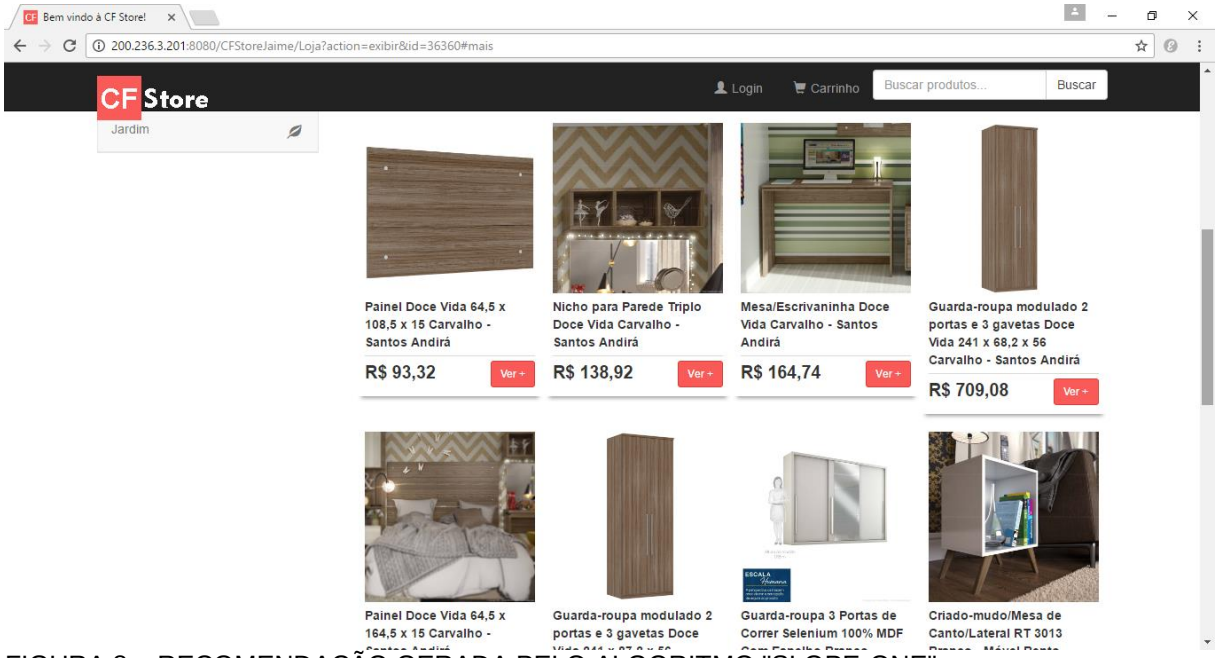

FIGURA 8 – RECOMENDAÇÃO GERADA PELO ALGORITMO "SLOPE-ONE"
FONTE: OS AUTORES, 2016

Com essa comparação direta, é possível concluir que ambas as técnicas para a recomendação são bastante similares. Para alguns itens pesquisados, os resultados produzem uma repetição nos itens apresentados, alterando-se apenas a ordem de alguns; do mais indicado ao menos indicado.

Durante esta fase observou-se, também, que a qualidade das recomendações de cada algoritmo; segundo o julgamento dos integrantes da equipe, não é consistente. Ou seja, hora a melhor recomendação é trazida pelo algoritmo *item based* outrora é trazido pelo algoritmo que usa a técnica *slope-one*.

O teste de qualidade das recomendações feito pelo SOLID foi aberto à comunidade, portanto trará resultados que talvez possam definir qual técnica de recomendação dá os melhores resultados, sendo então uma fonte mais confiável para resolver esta questão isolada.

5.2 DIFICULDADES

O projeto surgiu da demanda de uma loja já existente que cedeu uma base em Excel com cerca de 200.000 registros de vendas. Para conseguir utilizá-la foi feito todo o processo de normalização dos dados, sendo que a informação do cliente não estava totalmente preenchida. Mas é fundamental para a criação das recomendações.

Por não ser de domínio dos integrantes, o assunto *algoritmos de recomendação* veio a ser um fator de superação. Passamos por um período de pesquisa que teve uma curva de aprendizado bastante acentuada, porém, a teoria estudada nem sempre reflete à pratica. O tempo que reservamos para estudos sobre algoritmos de recomendação foi um aspecto determinante para o desenvolvimento. Fazendo com que a construção de código começasse relativamente tarde para que todos os objetivos fossem alcançados de forma confortável.

Um ponto importante para o desenvolvimento foram as imagens de produtos (cerca de 27000 no catálogo da loja). Estas imagens, infelizmente não foram disponibilizadas, e para contornar isso tivemos de escrever um terceiro programa em Java. Só assim conseguimos capturar as imagens da loja online e criar uma base com estas. O que demandou uma quantia de tempo e de processamento muito grande, causando atraso na finalização do sistema, pois a ideia de vendas em um meio *online* funciona primeiramente através das suas imagens.

Durante o processo de análise das recomendações, tivemos problema com a consistência e confiança nos dados, pois não tivemos um veículo de divulgação para que a comunidade pudesse avaliar e dar um *feedback* sobre os resultados de cada algoritmo. Este estudo, porém, foi feito pelos professores do grupo SOLID, mesmo assim, não tivemos tempo hábil de colher ou negociar os resultados desta pesquisa para o enriquecimento deste trabalho. A análise, portanto, foi feita utilizando os mesmos produtos isolados nas duas aplicações, cada aplicação com o seu algoritmo, e comparando os resultados que cada recomendação trouxe para cada algoritmo.

5.3 TRABALHOS FUTUROS

Como uma proposta de melhoria, pode ser desenvolvido um novo sistema de recomendação híbrido baseando-se não apenas no item, mas também no perfil dos usuários, podendo assim trazer mais vendas para a loja de imóveis.

Por ser uma aplicação com o intuito apenas de simular uma loja aos olhos do cliente, não está previsto nesse sistema um meio simples de manutenção. Para isso há a possibilidade da criação de uma área para administradores do sistema. Esta área deve facilitar a manutenção do sistema, disponibilizando uma forma mais prática de se executar funcionalidades previstas inicialmente, mas que saíram do escopo devido ao foco do projeto, e até mesmo outras funcionalidades não previstas.

REFERÊNCIAS

BRUEGGE, Bernd; DUTOID, Allen. **Object-Oriented Software Engineering Using UML, and Java**. 3rd ed. Upper Saddle River: Pearson, 2009.

COBB, Charles. **Making Sense of Agile Project Management**. Hoboken: Wiley, 2011.

FOWLER, Martin. **UML Distilled.** 3rd ed. Boston: Addison-Wesley Professional, 2003.

HARRISON, Guy; FEUERSTEIN, Steven. **MySQL Stored Procedure Programming**. Sebastopol: O'Reilly Media, 2006.

OWEN, Sean; ANIL, Robin; DUNNING, Ted; FRIEDMAN, Ellen. **Mahout in Action.** Manning, 2012.

PICHLER, Roman. **Agile Product Management with Scrum**. Boston: Addison-Wesley, 2010.

RICCI, Francesco; ROKACH, Lior; SAPHIRA, Bracha; KANTOR, Paul B.. **Recommender Systems Handbook.** Springer, 2011.

SOMMERVILLE, Ian. **Software Engineering.** 9th ed. Boston: Pearson, 2010.

STEELMAN, Andrea; MURACH, Joe. **Murach's Java Servlets and JSP.** 2nd ed. Mike Murach & Associates, 2008.

APÊNDICES

APÊNDICE A – Diagrama de casos de uso

O diagrama de casos de uso descreve as funcionalidades do sistema e como os atores irão interagir com ele.

Os casos de uso foram modelados inicialmente como consta na figura 6. Posteriormente, a mudança no escopo do projeto resultou no diagrama de casos de uso da figura 7.

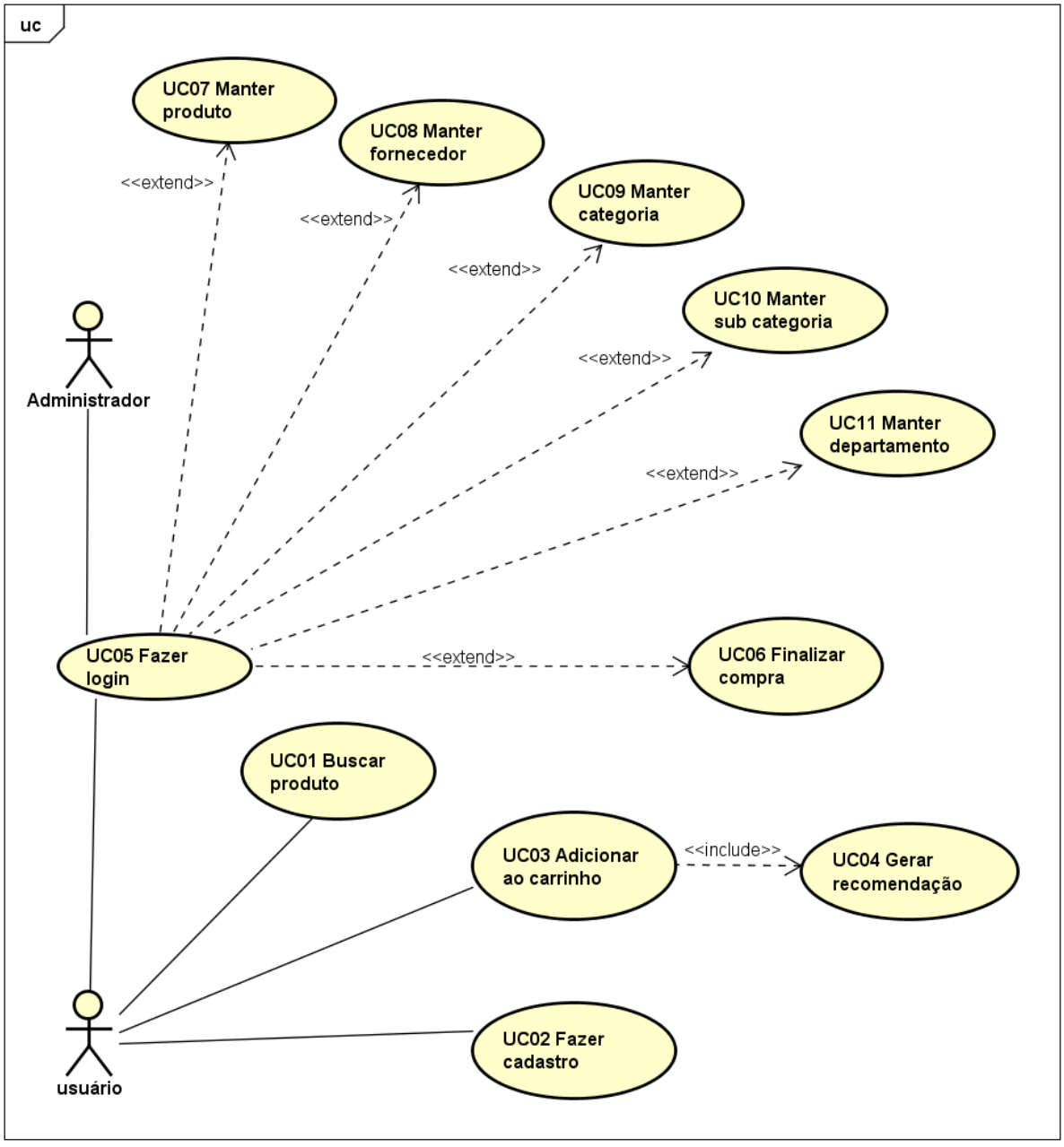

FIGURA 9 – DIAGRAMA DE CASOS DE USO INICIAL
FONTE: OS AUTORES, 2016

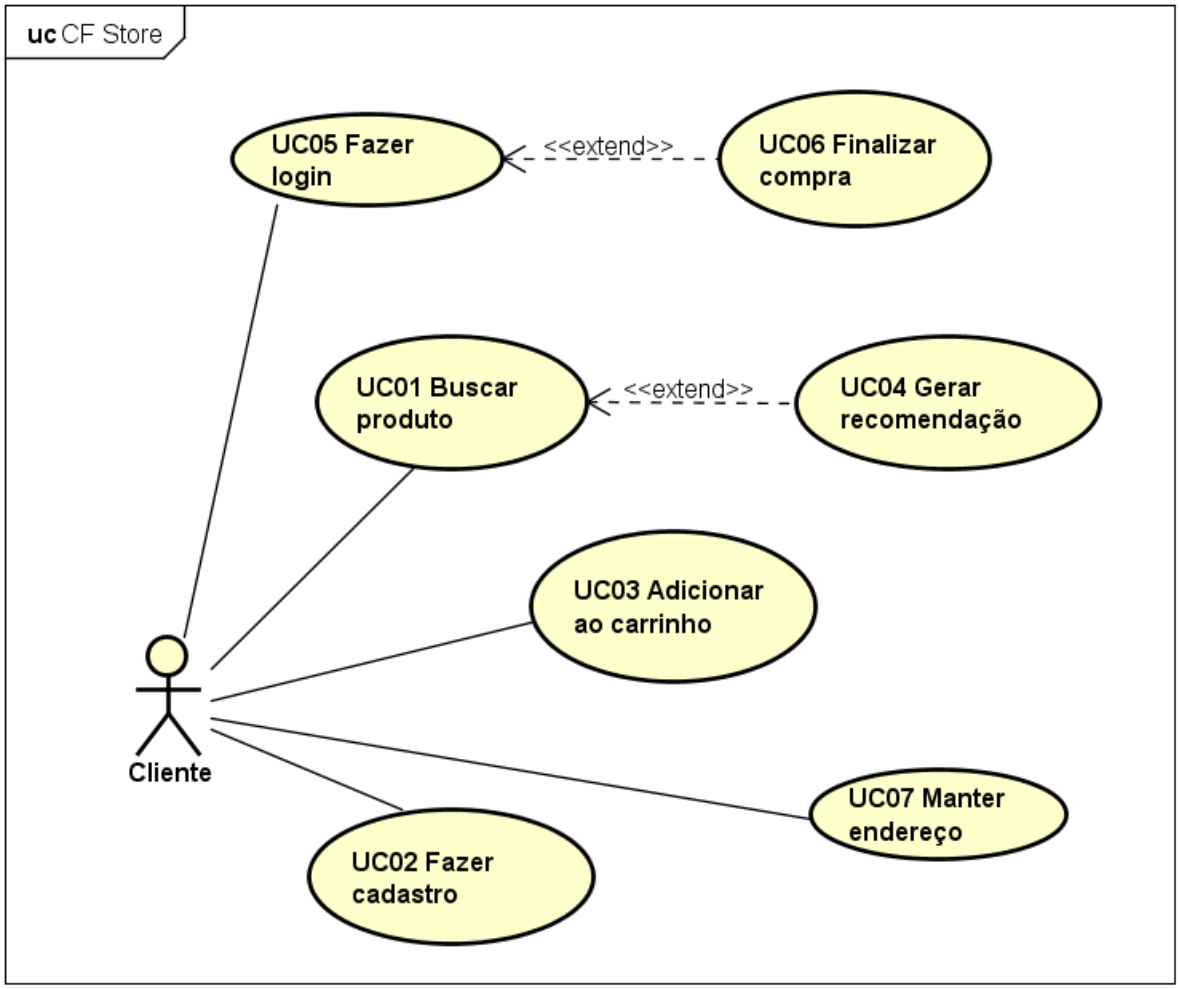

FIGURA 10 – DIAGRAMA DE CASOS DE USO IMPLEMENTAÇÃO
FONTE: OS AUTORES, 2016

APÊNDICE B – DESCRIÇÕES DOS CASOS DE USO

UC01 – BUSCAR PRODUTO

Descrição: Este caso de uso descreve como o usuário faz uma busca por produtos.

Pré-condições:

1. Não há.

Pós-condições:

Após o fim normal deste caso de uso o sistema deve:

1. Mostrar uma lista de produtos que atendem aos critérios da busca feita pelo usuário.

Ator Principal:

Usuário do sistema.

Regras de negócio

1. Não há.

Fluxo de Eventos Principal

1. O sistema apresenta a tela inicial para o usuário.
2. O usuário informa um critério de busca na barra de busca da tela inicial e clica no botão “Buscar” ou pressiona a tecla <Enter>.
3. O sistema redireciona para a tela com os resultados encontrados.

Fluxos Alternativos

A01: O sistema não possui nenhum item que atende ao critério digitado.

1. O sistema apresenta a tela inicial para o usuário.
2. O usuário informa um critério de busca na barra de busca da tela inicial e clica no botão “Buscar” ou pressiona a tecla <Enter>.

3. O sistema não encontra nenhum item que atende a este critério.
4. O sistema redireciona para uma tela de erro informando que não foram encontrados produtos que atendam os critérios da busca.

A02: O usuário não informa o critério de busca.

1. O sistema apresenta a tela inicial para o usuário.
5. O usuário não informa um critério na barra de busca da tela inicial e clica no botão “Buscar” ou pressiona a tecla <Enter>.
2. O sistema redireciona para uma tela de erro informando que não foi digitado um critério para busca.

UC02 – FAZER CADASTRO

Descrição: Este caso de uso descreve como o usuário faz o seu cadastro na loja CFStore.

Pré-condições

1. Não há.

Pós-condições

Após o fim normal deste caso de uso o sistema deve:

1. Autenticar o novo usuário cadastrado e enviar um e-mail de boas-vindas ao mesmo.

Ator Principal

Usuário do sistema.

Regras de negócio

1. O novo usuário não pode possuir dados já cadastrados.

Fluxo de Eventos Principal

1. O sistema apresenta a tela de cadastro para o usuário.
2. O usuário preenche o formulário de cadastro com todos os dados obrigatórios.
3. O usuário clica no botão “Cadastrar”.

Fluxos Alternativos

1. Não há.

Fluxos de Exceção

1. O sistema apresenta a tela de cadastro para o usuário.
2. O usuário preenche o formulário de cadastro, porém, sem informar todos os dados obrigatórios necessários.
3. O usuário clica no botão “Cadastrar”.
4. O sistema notifica o usuário do não preenchimento dos campos obrigatórios.

UC03 – ADICIONAR AO CARRINHO

Descrição: Este caso de uso descreve como o usuário adiciona um produto ao seu carrinho.

Pré-condições

1. Não há.

Pós-condições

Após o fim normal deste caso de uso o sistema deve:

1. Salvar os itens com seus respectivos valores e quantidades no novo pedido gerado.

Ator Principal

Usuário do sistema.

Regras de negócio

1. Não há.

Fluxo de Eventos Principal

1. O sistema apresenta a tela inicial ao usuário.
2. O usuário clica no botão “Ver +” de um produto qualquer.
3. Na tela do detalhe, o usuário informa a quantidade de itens e clica no botão “Adicionar ao carrinho”.
4. O sistema exibe uma mensagem notificando que o item foi adicionado ao carrinho de compras.

Fluxos Alternativos

1. Não há.

UC04 – GERAR RECOMENDAÇÃO

Descrição: Este caso de uso descreve como será gerada a recomendação de novos produtos a partir das vendas que foram realizadas.

Pré-condições

1. Não há.

Pós-condições

Após o fim normal deste caso de uso o sistema deve:

1. Salvar no banco de dados as recomendações correspondentes a cada produto.

Ator Principal

Administrador.

Regras de negócio

1. Não há.

Fluxo de Eventos Principal

1. O administrador do sistema executa a classe de Recomendação.
2. As recomendações geradas são salvas no banco de dados da aplicação.

Fluxos Alternativos

1. Não há.

UC05 – FAZER LOGIN

Descrição: Este caso de uso descreve como um usuário cadastrado pode fazer *login* no sistema.

Pré-condições

1. O usuário deve possuir um cadastro ativo.

Pós-condições

Após o fim normal deste caso de uso o sistema deve:

1. Autenticar o usuário no sistema.

Ator Principal

Qualquer usuário cadastrado no sistema.

Regras de negócio

1. O usuário deve estar previamente cadastrado no sistema.

Fluxo de Eventos Principal

1. O sistema apresenta a tela de login ao usuário
2. O usuário preenche os campos usuário e senha com seu e-mail e senha, respectivamente.
3. O sistema valida as informações fornecidas pelo usuário.
4. O sistema apresenta a tela inicial ao usuário

Fluxos Alternativos

1. Não há.

Fluxos de Exceção

1. O usuário preenche os campos de usuário e/ou senha incorretos.
2. O sistema valida as informações fornecidas pelo usuário.
3. O sistema notifica o usuário que houve erro na autenticação pois o campo usuário ou senha está incorreto.

UC06 – FINALIZAR COMPRA

Descrição: Este caso de uso descreve como um usuário cadastrado concluir a compra dos itens do carrinho.

Pré-condições

1. O usuário deve possuir um cadastro ativo e deve estar autenticado no sistema.

Pós-condições

Após o fim normal deste caso de uso o sistema deve:

1. Enviar um e-mail ao cliente com a confirmação da compra.

Ator Principal

Qualquer usuário cadastrado no sistema.

Regras de negócio

1. O usuário deve estar previamente cadastrado no sistema e deve ter itens inseridos no seu carrinho de compras.

Fluxo de Eventos Principal

1. O sistema apresenta a tela de carrinho ao usuário
2. O clica no botão “Finalizar compra”.
3. O sistema valida as informações fornecidas pelo usuário.
4. O sistema apresenta a tela de confirmação da compra ao usuário.

Fluxos Alternativos

1. Não há.

Fluxos de Exceção

1. O usuário não tem itens em seu carrinho.
2. O sistema informa ao usuário que o carrinho está vazio.

UC07 – MANTER ENDEREÇO

Descrição: Este caso de uso descreve como um usuário cadastrado pode alterar os dados de endereço.

Pré-condições

1. O usuário deve possuir um cadastro ativo e deve estar autenticado no sistema.

Pós-condições

Após o fim normal deste caso de uso o sistema deve:

1. Mostrar uma mensagem de confirmação com o novo endereço.

Ator Principal

Qualquer usuário cadastrado no sistema.

Regras de negócio

1. O usuário deve estar previamente cadastrado no sistema.

Fluxo de Eventos Principal

1. O sistema apresenta a tela de login ao usuário
2. O usuário efetua o login.
3. O sistema apresenta a tela dos dados do usuário.
4. O usuário altera as informações referentes ao endereço.
5. O usuário clica em “Salvar alterações”.
6. O sistema exibe mensagem de sucesso na alteração.

Fluxos Alternativos

1. Não há.

Fluxos de Exceção

1. Não há

APÊNDICE C – DIAGRAMA DE CLASSE

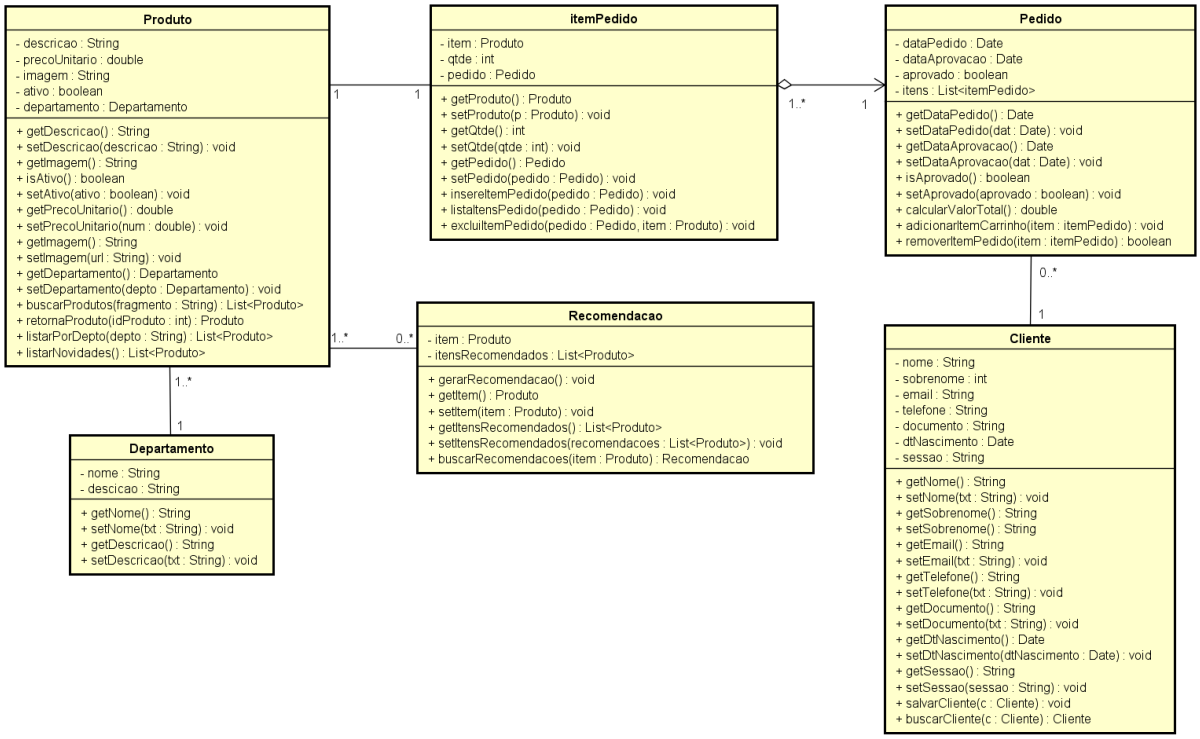

FIGURA 11 – DIAGRAMA DE CLASSES
FONTE: OS AUTORES, 2016

APÊNDICE D – DIAGRAMA DE ARQUITETURA

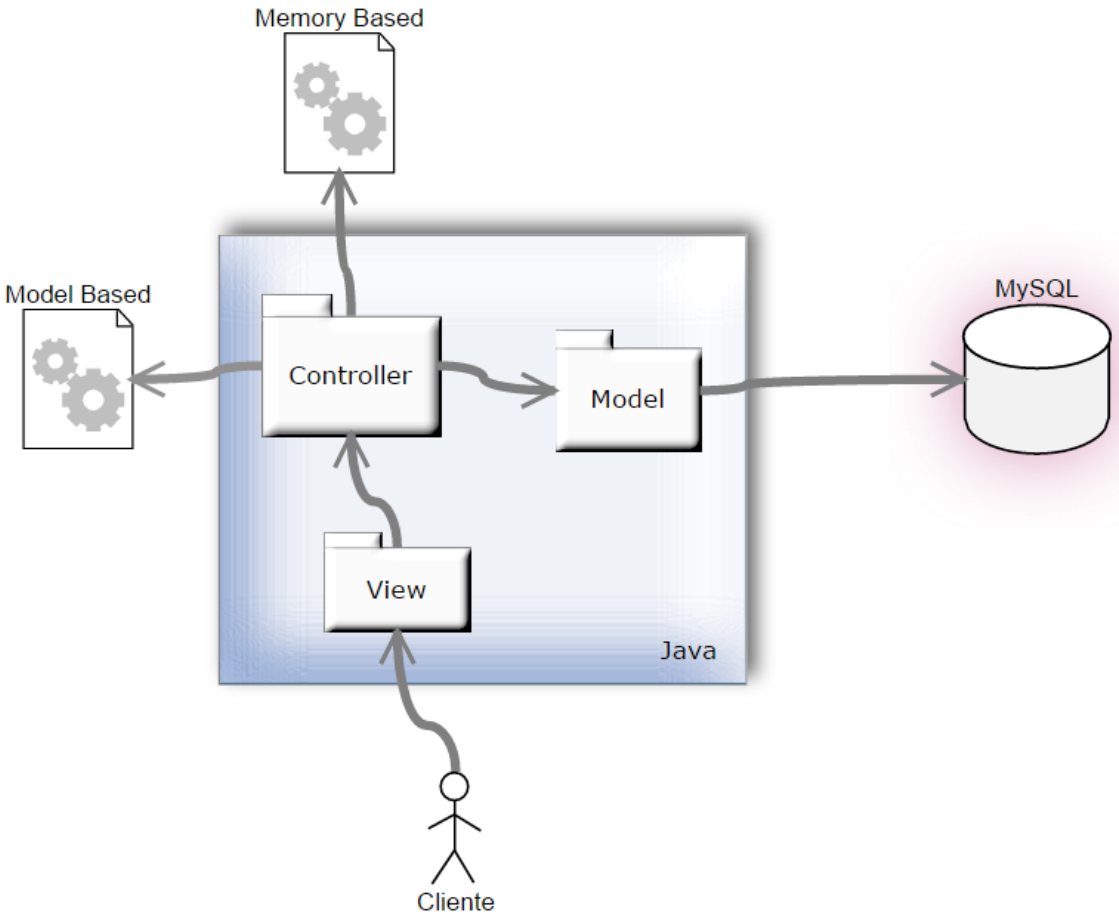

FIGURA 12 – DIAGRAMA DE ARQUITETURA
FONTE: OS AUTORES, 2016

APÊNDICE E – DIAGRAMA DE SEQUÊNCIA

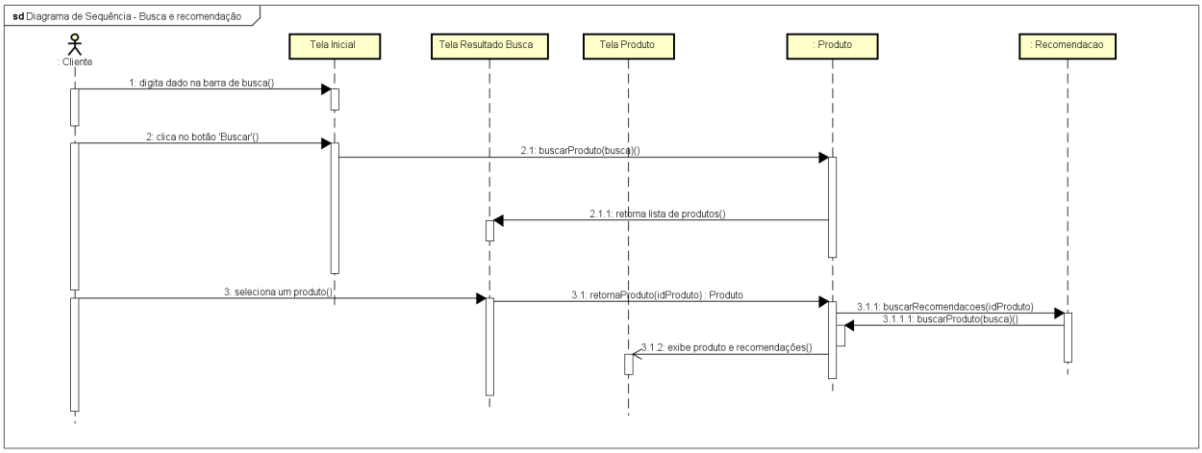

FIGURA 13 – DIAGRAMA DE SEQUENCIA
FONTE: OS AUTORES, 2016

APENDICE F – DIAGRAMA DE ENTIDADE RELACIONAMENTO

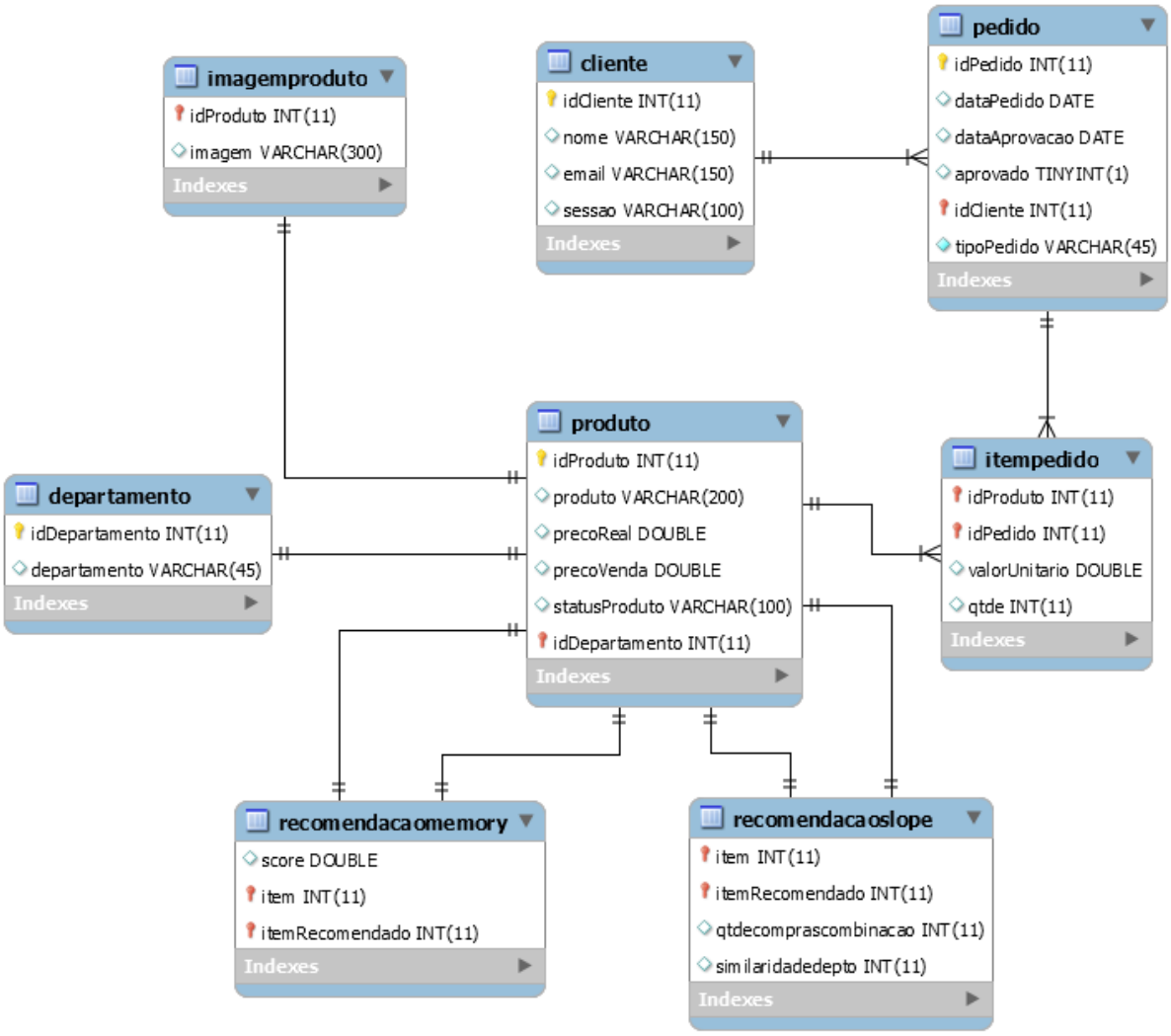

FIGURA 14 – DIAGRAMA DE ENTIDADE RELACIONAMENTO
FONTE: OS AUTORES, 2016

APENDICE G – PLANO DE RISCOS

ID	Risco	Data limite	Consequência	Ação	Monitor	Probabilidade	Impacto
1	Dificuldade com algoritmos de recomendação	01/10	Atraso no início do desenvolvimento	Aumentar tempo e/ou recursos	Leandro	Media	Médio
2	Incompatibilidade de tecnologias	01/10	Atraso no início do desenvolvimento e/ou retrabalho	Prova de conceitos	Regis	Baixa	Leve
3	Dificuldades quanto à aplicação de SPs	12/10	Retrabalho	Prova de conceitos	Regis	Media	Médio
4	Danificação/problemas de hardware	N/A	Queda de produtividade, atraso ou impossibilidade de conclusão	Investimento imediato/uso de recursos da UFPR	Todos	Alto	Grave
5	Saída de integrante	11/11	Impossibilidade de conclusão	Remanejamento de integrante	Todos	Baixa	Grave
6	Alteração de escopo	21/11	Atraso na entrega/Retrabalho	Aumentar tempo e/ou recursos	Todos	Media	Grave

TABELA 2 – PLANO DE RISCOS
FONTE: OS AUTORES, 2016

APENDICE H – WBS

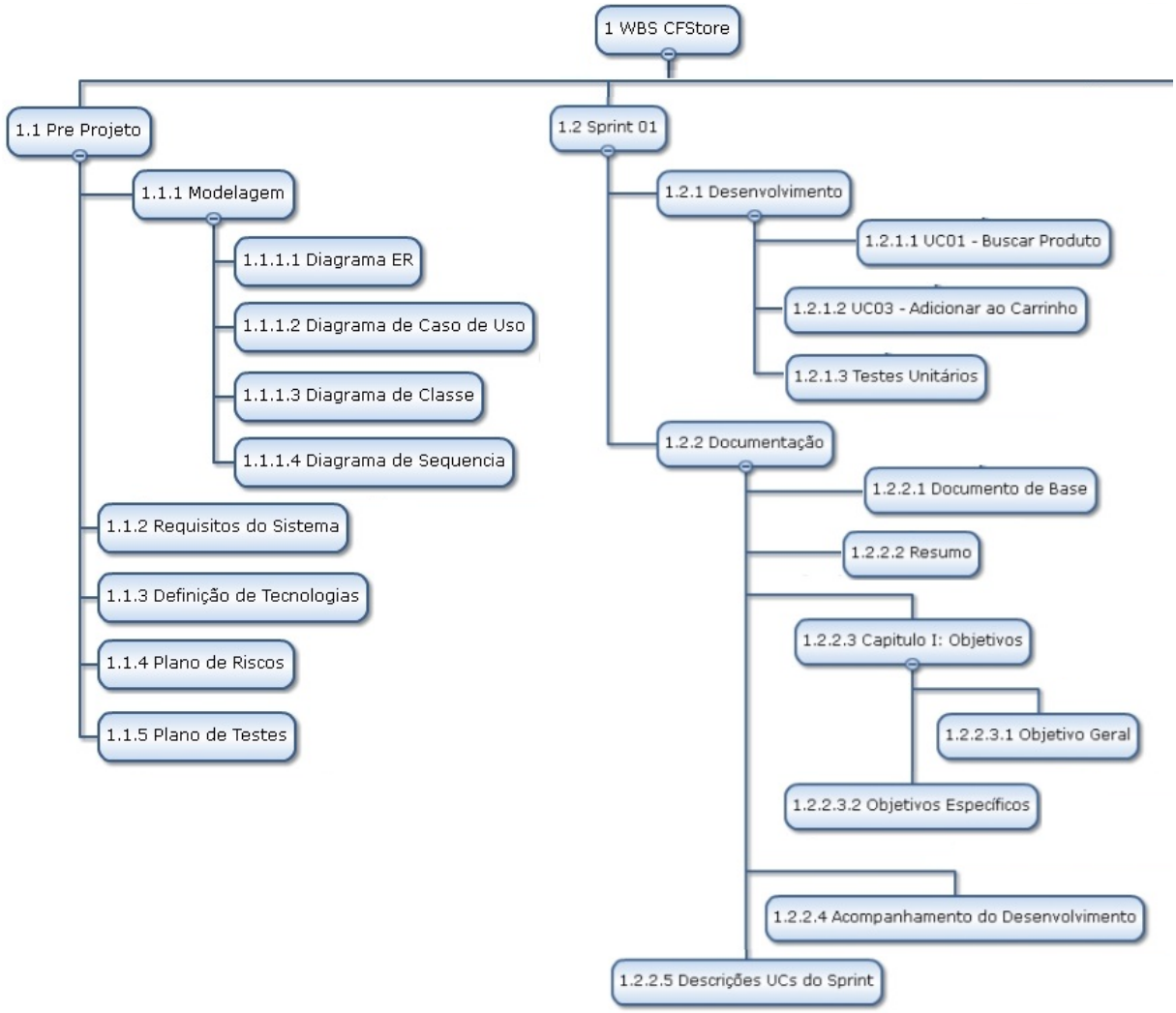

FIGURA 15 – WBS (PARTE 1)
FONTE: OS AUTORES, 2016

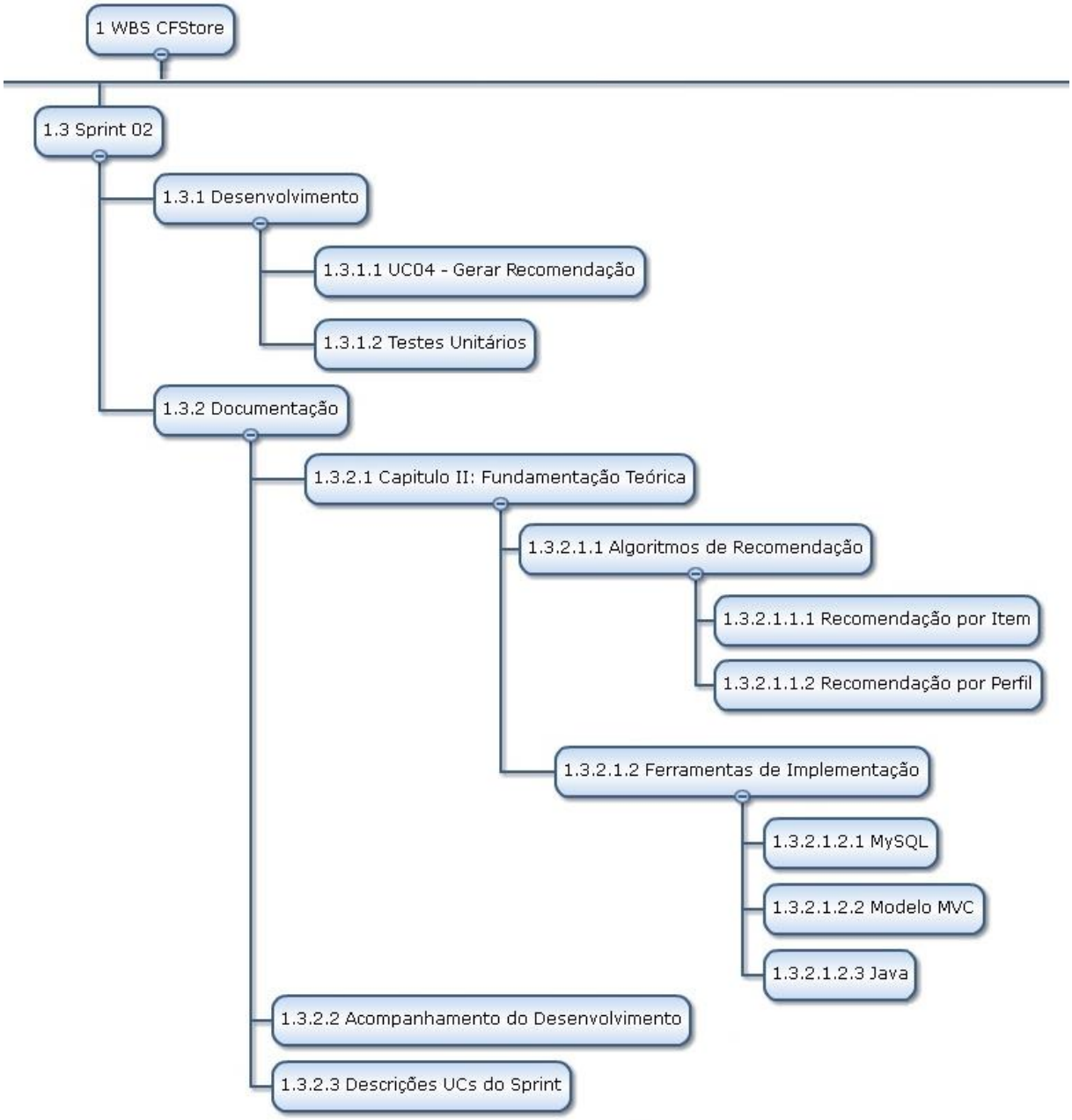

FIGURA 16 – WBS (PARTE 2)
FONTE: OS AUTORES, 2016

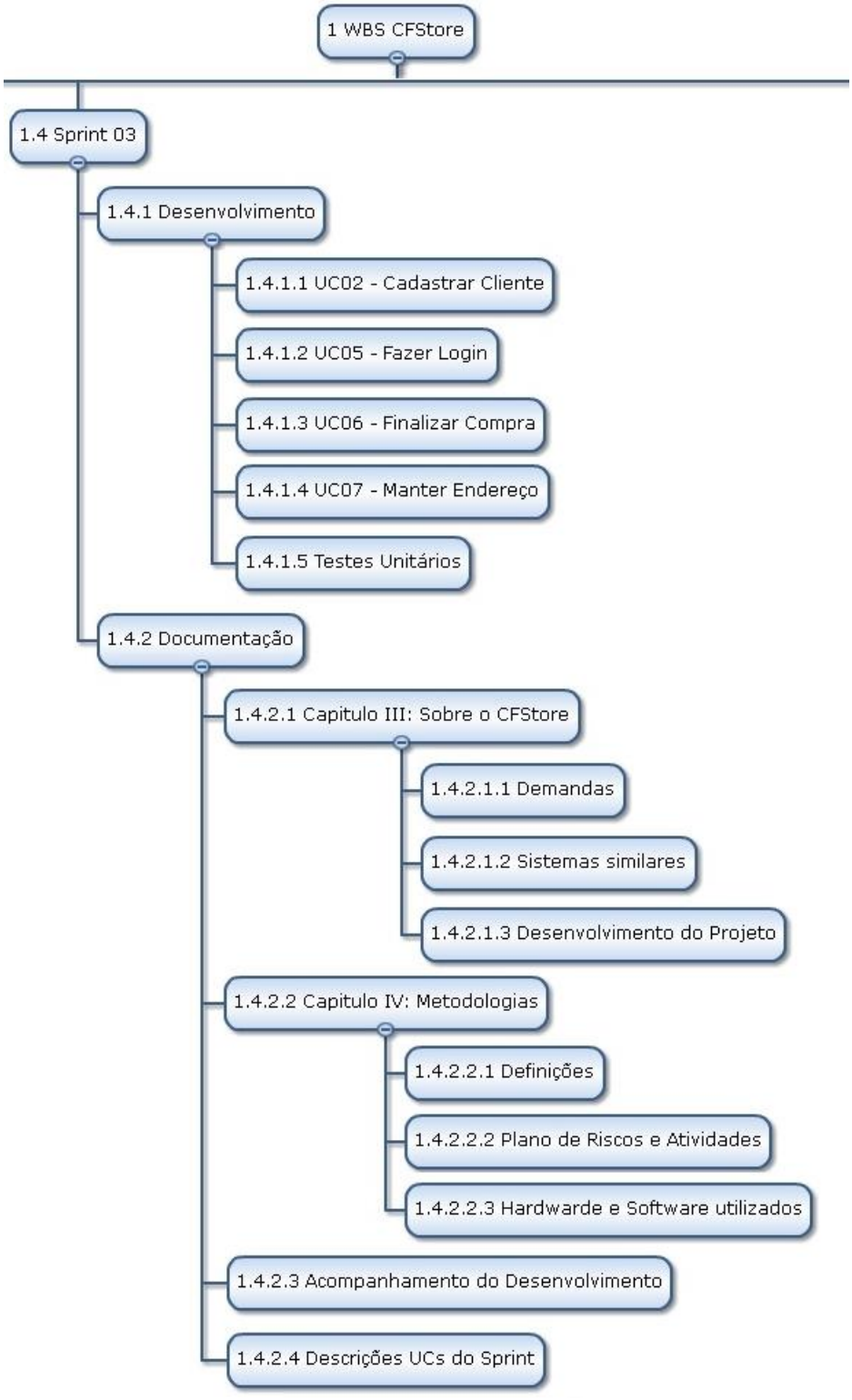

FIGURA 17 – WBS (PARTE 3)
FONTE: OS AUTORES, 2016

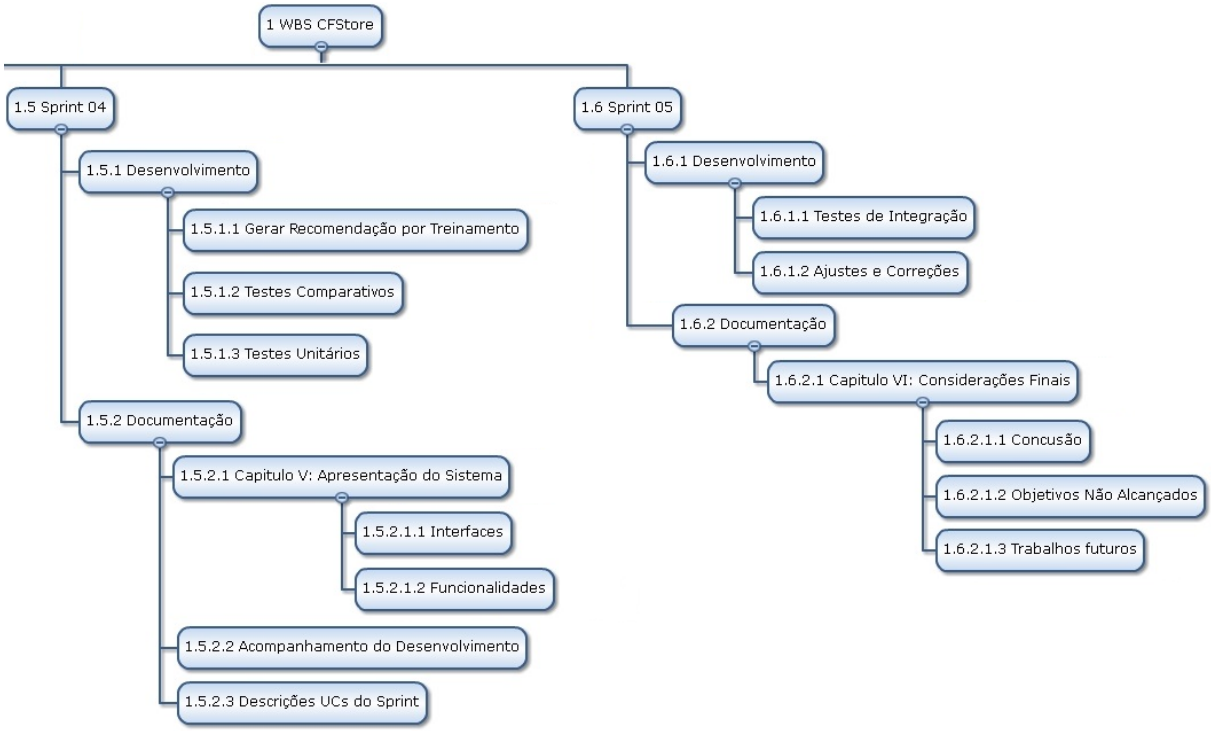

FIGURA 18 – WBS (PARTE 4)
FONTE: OS AUTORES, 2016

APENDICE J – REQUISITOS

Tipo de requisito: Funcional

Código do requisito: RF1

Nome do requisito: Buscar produto

Prioridade: Alta

Descrição: O usuário poderá pesquisar produtos de acordo com qualquer propriedade do produto, tal como cor, quantidade de portas, setor, entre outros.

Dependências: Não há.

Histórico:

Tipo de requisito: Funcional

Código do requisito: RF2

Nome do requisito: Fazer cadastro

Prioridade: Media

Descrição: O usuário deve cadastrar seus dados e uma senha, para poder finalizar compras.

Dependências: RF7.

Histórico:

Tipo de requisito: Funcional

Código do requisito: RF3

Nome do requisito: Adicionar ao carrinho

Prioridade: Alta

Descrição: O usuário pode adicionar produtos ao carrinho como bem entender, este requisito também considera a função de retirada de produtos do carrinho, se

Dependências: Não há

Histórico:

Tipo de requisito: Funcional

Código do requisito: RF4

Nome do requisito: Gerar recomendação

Prioridade: Alta

Descrição: No momento que o usuário visualizar o detalhe de um produto, o sistema recupera as recomendações para o produto em questão.

Dependências: Não há

Histórico:

Tipo de requisito: Funcional

Código do requisito: RF5

Nome do requisito: Fazer login

Prioridade: Média

Descrição: O usuário autentica-se no serviço para poder finalizar compras. O login também identifica as permissões de cada usuário autenticado.

Dependências: RF2

Histórico:

Tipo de requisito: Funcional

Código do requisito: RF6

Nome do requisito: Finalizar compra

Prioridade: Baixa

Descrição: O usuário pode finalizar uma compra. Neste momento o sistema dispara um e-mail de notificação para o endereço cadastrado.

Dependências: RF2, RF3

Histórico:

Tipo de requisito: Funcional

Código do requisito: RF7

Nome do requisito: Manter endereço

Prioridade: Media

Descrição: Ao cadastrar seus dados, o usuário deve informar também um endereço, que pode ser modificado mais tarde, caso haja necessidade.

Dependências: RF2

Histórico:

Tipo de requisito: Funcional

Código do requisito: RF8

Nome do requisito: Manter produto

Prioridade: Baixa

Descrição: O usuário autenticado como administrador do sistema pode efetuar funções de CRUD de produtos.

Dependências: RF5

Histórico: 17/11/2016 – Requisito cancelado devido a alteração do escopo.

Tipo de requisito: Funcional

Código do requisito: RF9

Nome do requisito: Manter fornecedor

Prioridade: Baixa

Descrição: O usuário autenticado como administrador do sistema pode efetuar funções de CRUD de fornecedores.

Dependências: RF5

Histórico: 17/11/2016 – Requisito cancelado devido a alteração do escopo.

Tipo de requisito: Funcional

Código do requisito: RF10

Nome do requisito: Manter categoria

Prioridade: Baixa

Descrição: O usuário autenticado como administrador do sistema pode efetuar funções de CRUD de categoria de produto.

Dependências: RF5

Histórico: 17/11/2016 – Requisito cancelado devido a alteração do escopo.

Tipo de requisito: Funcional

Código do requisito: RF11

Nome do requisito: Manter subcategoria

Prioridade: Baixa

Descrição: O usuário autenticado como administrador do sistema pode efetuar funções de CRUD de subcategoria de produto.

Dependências: RF5

Histórico: 17/11/2016 – Requisito cancelado devido a alteração do escopo.

Tipo de requisito: Funcional

Código do requisito: RF12

Nome do requisito: Manter departamento

Prioridade: Baixa

Descrição: O usuário autenticado como administrador do sistema pode efetuar funções de CRUD de departamento.

Dependências: RF5

Histórico: 17/11/2016 – Requisito cancelado devido a alteração do escopo.

Tipo de requisito: Não funcional

Código do requisito: RNF1

Nome do requisito: Linguagem de programação

Prioridade: Alta

Descrição: O sistema será desenvolvido na linguagem de programação Java, com a IDE *NetBeans*.

Dependências: Não há

Histórico:

Tipo de requisito: Não funcional

Código do requisito: RNF2

Nome do requisito: Banco de dados

Prioridade: Alta

Descrição: O sistema será desenvolvido utilizando *Stored Procedures* no banco de dados MySQL, utilizando-se da ferramenta *MySQL Workbench*.

Dependências: Não há

Histórico:

Tipo de requisito: Não funcional

Código do requisito: RNF3

Nome do requisito: Arquitetura

Prioridade: Alta

Descrição: O sistema será desenvolvido usando a arquitetura MVC.

Dependências: Não há

Histórico:

Tipo de requisito: Não funcional

Código do requisito: RNF4

Nome do requisito: Criptografia

Prioridade: Média

Descrição: O sistema será desenvolvido utilizando-se de criptografia MD5 para as funções de autenticação.

Dependências: Não há

Histórico:

Tipo de requisito: Não funcional

Código do requisito: RNF5

Nome do requisito: Validação de campos

Prioridade: Alta

Descrição: As informações provindas do usuário, nos formulários do sistema, devem ser validadas para garantir sua integridade.

Dependências: Não há

Histórico: