



Projeto Integrador de Competência em Eng. Software

Engenharia de Software
Universidade Cruzeiro do Sul (UNICSUL)
45 pag.

Projeto Integrador Transdisciplinar em Engenharia de Software I.



Conteudista: Prof. Me. Artur Marques

Revisão Textual: Prof.^a Ma. Sandra Regina Fonseca Moreira

≡ Material Teórico

🔗 Material Complementar

DESAFIO

≡ Situação-Problema 1

≡ Situação-Problema 2

≡ Situação-Problema 3

≡ Problema em Foco


ATIVIDADE

≡ Atividade de Entrega

Material Teórico

Olá, estudante!

Vamos iniciar a disciplina abordando os conceitos necessários para que você possa realizar a atividade através de cada situação-problema mais à frente.

 **Atenção, estudante!** Aqui, reforçamos o acesso ao conteúdo *online* para que você assista à videoaula. Será muito importante para o entendimento do conteúdo.

Introdução

A ideia do desenvolvimento deste conteúdo didático é fazer com que você possa se recordar do que você já aprendeu de forma a mantê-lo(a) focado no desafio e ter praticamente tudo do precisa para desenvolvê-lo num lugar só.

Você também encontrará leituras e material complementares que, sinceramente te alerto, serão essenciais no aprofundamento e desenvolvimento desse desafio. Muitas pessoas e discentes

comentam que muitos RH de empresas pedem experiência aos candidatos. Com certeza, é bastante estranho pedir por algo dessa natureza a quem está tentando entrar, ou desenvolver, uma carreira. Porém, com esses desafios você tem a oportunidade de mostrar que desenvolveu seu aprendizado e que está pronto para qualquer desafio de seleção de pessoas para concorrer de igual para igual com qualquer outro candidato.

Vamos aproveitar para recordar agora?!

Requisitos Ágeis

Primeiramente, algumas constatações. Muitas vezes, no início de minha carreira vi projetos com dificuldades porque seus requisitos eram mal elaborados, muitas vezes sem sentido, e eu me sentia um completo inútil. Eu olhava para os "requisitos" e eles não eram nada mais do que um quadro com uma infinidade de notas e postites. Eu era desenvolvedor e tinha muitas dúvidas e, em geral, todas as pessoas envolvidas, clientes, usuários, desenvolvedores, analistas de negócios, arquitetos de solução e inclusive os testadores, não têm um bom entendimento do sistema como um todo e quais são as várias *personas* que usam o sistema. Claro que faltava, também creio eu, a todas essas pessoas, se apropriarem de verdade do conhecimento e dos processos do negócio. Minha sorte foi que, o tempo e a dedicação sempre trazem frutos, para o bem ou para o mal. No meu caso, foi para o bem!

Então respire fundo e se prepare para pensar e analisar muito em profundidade, pois analistas de sistemas criados no raso jamais enfrentaram grandes tempestades e ondas avassaladoras, acredite. Aqui serão apresentadas dicas preciosas para você desenvolver seu trabalho com menos incerteza, vindas de um "velho marujo".

Um bom requisito deve dizer a cada membro do público exatamente qual é a funcionalidade esperada e nunca gerar uma miríade de perguntas de todos os envolvidos. Frequentemente, é difícil solicitar informações a um cliente, mas documentar para desenvolvedores nunca deve ser tão difícil. E nesse caso você vai possuir múltiplos "chapéus"; ora será analista de negócios, ora desenvolvedor, ora etc.

Bons requisitos precisam de:

- Histórias de usuários;
- Testes de Aceitação do Usuário;
- Fluxo de Trabalho;
- Detalhes dos Requisitos;
- *Wireframes*.

Sem nenhuma das seções mencionadas, os requisitos começam a perder valor e sentido. Muitas vezes, quando sou chamado para dar consultoria, vejo times ágeis e especificações de sistemas como um "punhado de gente bem-intencionada que não consegue escrever ou muitas vezes ler o livro que ganharam". Daí fica realmente difícil construir algo se nem entendemos o português que está escrito, não é mesmo?! Cada seção de análise traz muito para a mesa e muitas vezes são julgadas como uma "perda de tempo", porém, quando vamos codificar, fazem toda a diferença.

Olhe só:

Histórias de Usuários

Indica todos os cenários dos usuários envolvidos. O padrão mundial disso é:

Como ALGUM PAPEL,
EU QUERO FAZER ALGUMA COISA, PARA
QUE POSSA OBTER ALGUNS BENEFÍCIOS.

Ou seja,

Como GERENTE DE CONTAS A RECEBER, EU QUERO ter uma relação de devedores por safra (meses vencidos 0d, 60, 90d, 180, 360d), PARA QUE POSSA encaminhar às empresas de cobrança por seguimento e especialidade delas.

Por exemplo,

Entendeu o que esperamos de você. Dezenas de cartões de história do usuário completos, claros, concisos, especificando a ação e o que se espera, incluindo para que serve.

Por favor, pense como um profissional, e você vai pensar, "mas sou aluno", sim todos nós somos alunos, incluindo o autor desse material, mas atitude é tudo e é isso que queremos de você. Atitude positiva e construtiva!

As histórias de usuários são essenciais para definir exatamente quem fará o quê e por quais razões.

Fluxo de Trabalho

Isso deve incluir uma imagem das telas envolvidas (você fará isso na IHC e no protótipo das telas; riqueza de detalhes é fundamental). Os estados de erro (incluindo as telas de mensagem de erros) e as alterações de visualização com base na função devem ser documentados. Aqui, uma imagem vale mais que mil palavras, pois os detalhes do fluxo através do recurso podem ser bastante complexos, e é difícil explicar os detalhes na próxima seção (aqui, visão é tudo, me desculpem os sinestésicos e auditivos).

Segue uma lista de alguns para você usar na construção de sua aplicação. Eles ajudarão a explicar o sistema através de fluxos e vão apoiar no desenvolvimento. Suas versões demo vão permitir que você utilize pelo tempo desse desafio, tranquilamente, portanto, não "durma no ponto". São todos amigáveis e não precisam de curso para usar. Use e abuse:

<u>Gliffy</u>	<u>Creately</u>	<u>Lucidchart</u>	<u>Canva</u>	<u>Diagrams</u>
-------------------------------	---------------------------------	-----------------------------------	------------------------------	---------------------------------

Detalhamento dos Requisitos

Esses são os detalhes do recurso. Documente todas as telas e todos os campos, rótulos, validações, mensagens e ações. Esta é essencialmente a especificação funcional dos detalhes da(s) tela(s) envolvida(s). Por estar no contexto do *wireframe* (próxima seção), é mais conciso. Você pode simplesmente fazer referência ao nome do campo, em vez de declarar detalhadamente tudo sobre o campo. Você pode manter os detalhes no comprimento do campo, obrigatório etc.

Casos de Uso

No início de um projeto, você precisa de vários dias para elicitar e, por que não, prever os requisitos de alto nível e entender o escopo, nesse caso, é o que você acha que o sistema deve fazer.

Seu objetivo é ter uma ideia do que é o projeto, não documentar em detalhes a operacionalidade do sistema.

A documentação pode vir mais tarde, se você realmente precisar dela. Para seu modelo de requisitos inicial, minha experiência é que você precisa de alguma forma de:

- **Modelo de uso:** permite que você explore como os usuários trabalharão com seu sistema. Pode ser uma coleção de casos de uso essenciais ou uma coleção de histórias de usuário;
- **Modelo de domínio inicial:** identifica os tipos de entidade de negócios fundamentais e os relacionamentos entre eles. Os modelos de domínio podem ser descritos como uma coleção de cartões, um diagrama de classe macro, ou até mesmo um modelo de dados geral. Este modelo de domínio conterá apenas informações suficientes: as principais entidades do domínio, seus principais atributos e os relacionamentos entre essas entidades. Seu modelo não precisa ser completo, ele só precisa cobrir informações

suficientes para que você se sinta confortável com os conceitos de domínio primário. (coloquei literatura de referência na sessão de Material Complementar e deve ser mandatoriamente lida, por gentileza);

- **Modelo de interface do usuário:** para projetos intensivos de interface de usuário, você deve considerar o desenvolvimento de alguns esboços de tela, ou até mesmo um protótipo de interface de usuário. (conforme coloquei no tópico *Wireframe* desse material).

Mas afinal, que nível de detalhes você realmente precisa?

Minha experiência é que você precisa de artefatos de requisitos que sejam bons o suficiente para lhe dar esse entendimento e nada mais, portanto, pense simples, porém direto.

Veja esse exemplo de um caso de uso geral:

Nome do caso de uso geral (macro): Inscrever-se numa Disciplina

Curso Básico de Ação (trata-se de um caso de uso normal):

- 1 Aluno digita seu nome e número de aluno;
- 2 O sistema verifica se o aluno está qualificado para se inscrever no curso. Se não for elegível, o aluno será informado e o caso de uso será encerrado;
- 3 O sistema exibe uma lista de disciplinas do curso disponíveis (naquele mês);
- 4 O aluno escolhe uma disciplina ou decide não se inscrever;
- 5 O sistema valida se o aluno está qualificado para se inscrever naquela disciplina (pode haver disciplinas anteriores como pré-requisito). Se não for elegível, o aluno é convidado a escolher outra disciplina;



6

O sistema valida se a disciplina se encaixa na trilha de aprendizagem do aluno;

7

O sistema calcula e exibe as taxas para que o aluno pague por aquela disciplina;

8

O aluno verifica o custo e indica que deseja se inscrever ou não;

9

O sistema inscreve o aluno na disciplina e cobra por ela;

10

O sistema imprime o comprovante de inscrição na disciplina quando o departamento de contas a receber der baixa no crédito.

A descrição acima contém informações suficientes para você entender o que o caso de uso escrito faz e, na verdade, pode conter informações demais para este ponto do ciclo de vida, já que apenas o nome do caso de uso pode ser suficiente para que o time de desenvolvimento ou outra parte interessada entenda os fundamentos do que se pede.

Porém, podemos detalhar isso, num caso de uso mais aprofundado, a que chamaremos de caso de uso expandido.

Nome: Inscrever-se numa Disciplina

Identificador: #C 49

Descrição: Inscrever um aluno existente em uma disciplina existente desde que ele seja elegível.

Pré-condições: O aluno precisa estar regularmente matriculado e pagando pontualmente a Universidade.

Pós-condições: O aluno matriculado na disciplina escolhida porque é elegível, pagante e se ainda houver vagas.

Caso Básico de Ação:

- 1 O caso de uso começa quando um aluno deseja se inscrever em uma disciplina;
- 2 O aluno insere seu nome e RGM no sistema por meio da TELA101 - TELA DE LOGIN;
- 3 O sistema verifica se o aluno está qualificado para se inscrever na disciplina de acordo com a regra de negócios RN32 - Determinar Elegibilidade para Matrícula na Disciplina. <Curso Alternativo Alfa>;
- 4 O sistema exibe a TELA102 - MENU DE DISCIPLINAS, que indica a lista de disciplinas disponíveis;
- 5 O aluno indica a disciplina em que deseja se inscrever. <Curso Alternativo Beta: O aluno decide não se matricular>;
- 6 O sistema valida se o aluno está qualificado para se inscrever na disciplina de acordo com a regra de negócios RN73 - Determinar a elegibilidade do aluno para se inscrever na disciplina. <Curso alternativo Delta>;
- 7 O sistema valida se a disciplina se encaixa na programação/trilha de aprendizagem existente do aluno de acordo com a regra de negócios RN97 - Validar Programação/Trilha da Disciplina do Aluno;
- 8 O sistema calcula as taxas da disciplina com base nas taxas publicadas no catálogo das disciplinas, os descontos de estudante aplicáveis e os impostos. Aplica as regras de negócios RN 143 - Calcular Taxas de Alunos, RN 107 - Calcular Descontos para Alunos e RN59 Calcular Impostos;
- 9 O sistema exibe as taxas via TELA189 - Exibir a tela de taxas da disciplina;
- 10 O sistema pergunta ao aluno se ainda deseja se inscrever na disciplina;

11

O aluno indica que deseja se inscrever na disciplina;

12

O sistema inscreve o aluno na disciplina;

13

O sistema informa ao aluno que a inscrição foi bem-sucedida por meio da TELA68 - Resumo da Matrícula da Disciplina;

14

O sistema emite a fatura/boleto para o aluno pagar pela disciplina, de acordo com a regra de negócios RN71 - Faturar o aluno pela disciplina;

15

O sistema pergunta ao aluno se ele deseja um extrato impresso da matrícula que será disponibilizado após confirmação do pagamento;

16

O aluno indica que deseja uma declaração impressa;

17

O sistema imprime um pdf da declaração de inscrição TELA39 - Relatório de resumo de inscrição (acessível somente após confirmação do pagamento);

18

O caso de uso termina quando o aluno pega a declaração impressa.

Curso alternativo Alfa: O aluno não é elegível para se inscrever na disciplina.

Alfa3: O algoritmo determina que o aluno não está qualificado para se inscrever na disciplina.

Alfa4: O algoritmo informa ao aluno que ele não pode se inscrever.

Alfa5: O caso de uso termina.

Curso alternativo Beta: O aluno decide não se inscrever em uma disciplina disponível.

Beta5: O aluno visualiza a lista de disciplinas e não encontrou aquela em que ele deseja se inscrever.

Beta6: O caso de uso termina.

Curso alternativo Delta: O aluno não possui os pré-requisitos para alguma disciplina.

Delta6: O algoritmo determina que o aluno não é elegível para se inscrever na disciplina que ele escolheu.

Delta7: O algoritmo informa ao aluno que ele não possui os pré-requisitos.

Delta8: O algoritmo informa ao aluno sobre os pré-requisitos de que ele precisa para se tornar elegível para aquela disciplina.

Delta9: O caso de uso continua, e é desviado para a linha 4 do curso básico de ação deste mesmo caso de uso.

Bem, você percebeu como é importante ser analítico, crítico e lógico nessa nossa profissão. A clareza e declaração sem ambiguidade são fundamentais quando estamos elicitando requisitos e/ou desenvolvendo casos de uso.

Temos aqui a descrição do mesmo caso de uso totalmente documentado. Este é um exemplo maravilhoso de um caso de uso bem construído, mas apresenta muito mais detalhes do que você possivelmente precisa no primeiro momento.

Se você realmente precisa desse nível de detalhe e, na prática, raramente o faz, pode capturá-lo quando realmente precisar.

Comunicação é tudo e, nesse caso, é preciso feedback com a equipe, porque quanto mais tempo você passar sem pedir feedback para o cliente/usuário, maior será o perigo de se modelarem coisas que não refletem o que as partes interessadas realmente precisam.

Dessa forma, na fase inicial do planejamento, use casos de uso simples como o do primeiro exemplo para completar ou ilustrar os cartões de história do usuário para dar mais clareza.

Nas fases posteriores, declare analiticamente como no segundo exemplo, afinal, fica difícil para um desenvolvedor saber o que ele tem que fazer sem as regras de negócios ou as telas da interface do usuário (camada de apresentação) que foram criadas. Por isso vemos tanta coisa sem sentido por aí, porque deixaram a critério do desenvolvedor algo que não é missão dele fazer. Lembre-se, [interface do usuário, *front end* e *back end*] são grandes atividades que se deve trabalhar seguindo padrões e metodologias porque uma depende da outra.

Caso você esteja no nível mais inicial possível, ainda pode recorrer ao bom e velho, e sem dúvida alguma, muito útil, diagrama de caso de uso.

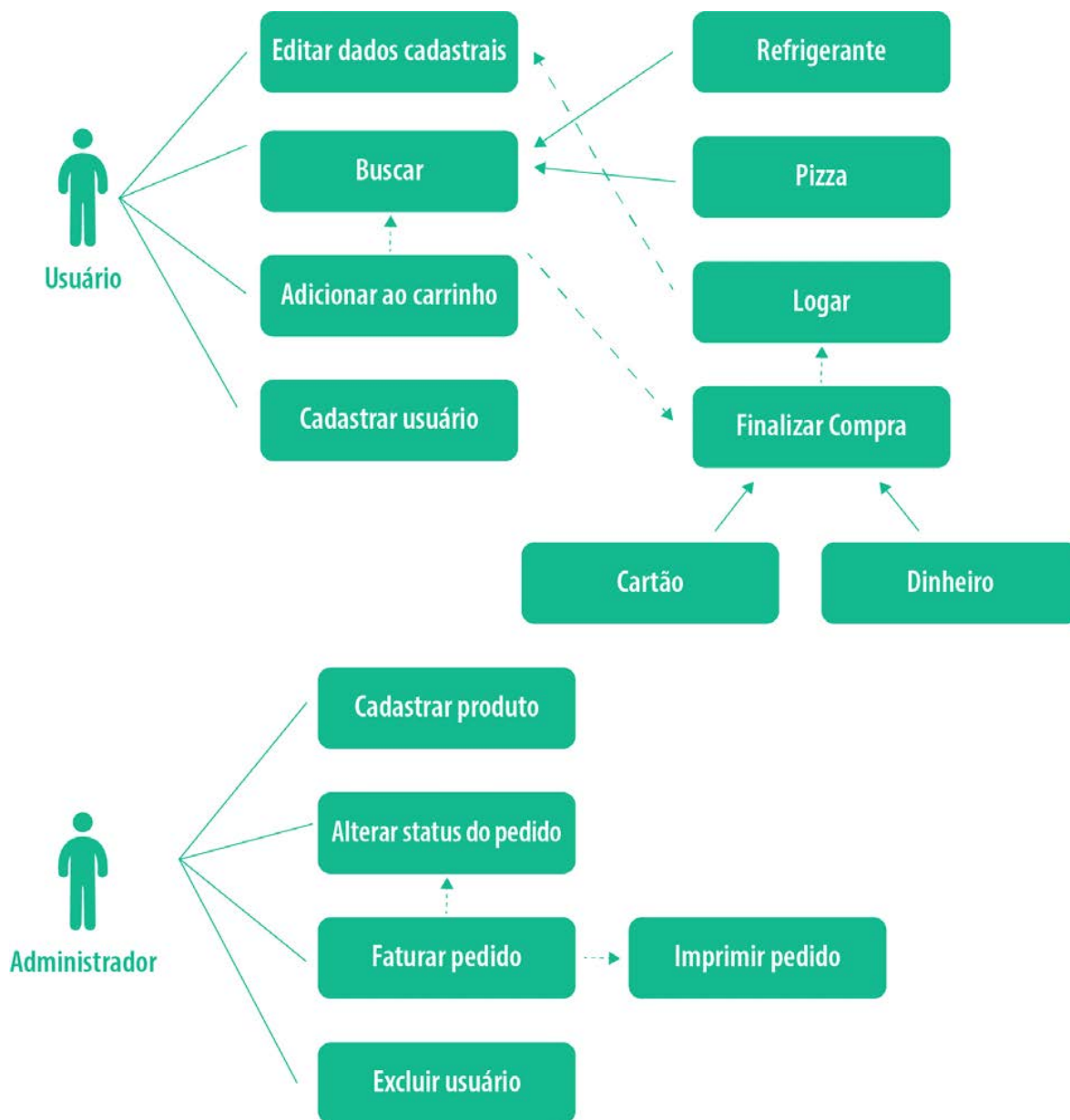


Figura 1 – Diagrama de Caso de Uso macro de uma pizzaria

Recomendo que você crie, inicialmente, todos os diagramas de caso de uso, como o demonstrado acima, para os cartões de história do usuário como requisitos e verifique se não falta nada, se há lógica e complementaridade e principalmente se o "bicho tem cabeça tronco e membros".


Tome cuidado porque há desafios no processo de levantar requisitos e convertê-los em artefatos. Abaixo listo alguns, mas dê muita atenção a todos eles para não cometer essas faltas.

- Acesso limitado às partes interessadas do projeto (clientes e usuários);
- Partes interessadas do projeto geograficamente dispersas (estão distantes em outras cidades, países ou regiões de difícil acesso. Nesse caso use reuniões remotas síncronas ou documentos como questionários);
- As partes interessadas do projeto não sabem o que querem (mais comum do que parece, nesse caso, seja um guia ou "terapeuta");
- As partes interessadas do projeto mudam de ideia (negocie, eles fazem isso por causa do item anterior, eles vão descobrindo conforme pensam no próprio negócio, coisa que a rotina não deixa);
- Prioridades conflitantes (saiba negociar com o *sponsor*, usuário ou *product owner*);
- Muitos participantes do projeto desejam ter voz;
- As partes interessadas do projeto prescrevem soluções de tecnologia (faça-os aterem-se à área de domínio deles, ou seja, negócios);
- As partes interessadas do projeto são incapazes de ver além da situação atual (em tecnologia, chamamos a isso de paralisia de paradigma, crença limitante ou a popular);
- As partes interessadas do projeto têm medo de ser fixadas (é uma questão cultural, às vezes, é necessário tempo e confiança, afinal, as partes interessadas devem ter responsabilidades pelo bom andamento do projeto);
- As partes interessadas do projeto não entendem os artefatos de modelagem. (conte uma história, pessoas se encantam com histórias, a técnica de *story telling* veio para ficar);
- Os desenvolvedores não entendem o domínio do problema (é importante os desenvolvedores estudarem o negócio em profundidade, não existe mais desenvolvedor "trancado numa sala"; eles precisam ter contato com a operação para amadurecerem e melhorarem seu entendimento sobre o negócio e sobre o mundo);

- As partes interessadas do projeto estão excessivamente focadas em um tipo de requisito (reuniões do tipo *Delphy* ou *Brainstorming* ajudam a tirar essas travas);
- As partes interessadas do projeto exigem formalidade significativa em relação aos requisitos (evite essa armadilha, partes interessadas muitas vezes não conhecem toda a solução e querem garantias impossíveis, pedir por detalhamento é um tipo de transferência de responsabilidade. Num projeto ágil, os requisitos e a própria arquitetura vão se revelando aos poucos);
- Os desenvolvedores não entendem os requisitos (convide os desenvolvedores a passarem um tempinho na operação, de preferência fazendo o papel do usuário, uma semana é o suficiente para eles entenderem tudo).

Diagrama de Classes

Vamos recordar os diagramas de classes?! Bem, eles mostram as classes do sistema, seus inter-relacionamentos, incluindo herança, agregação e associação e as operações e atributos das classes. Eles são usados para uma ampla variedade de propósitos, incluindo modelagem conceitual, de domínio e modelagem de projeto detalhado.



"Arquiteturas como diagramas de classe / pacote: A arquitetura é uma apresentação estrutural de todo o sistema. Geralmente, é descrito por diagramas de classe ou pacote, normalmente para mostrar camadas globais (camadas). Por exemplo, em um aplicativo com IU e banco de dados, as camadas são geralmente definidas horizontalmente da IU para o banco de dados e um caso de uso as percorre para atingir seu objetivo. Outros padrões de arquitetura como "*MVC*" (*Model-View-Controller*) também podem ser escolhidos como uma arquitetura global. A Figura.4 é um exemplo de arquitetura desenhada como um diagrama de pacote baseado na arquitetura MVC. Todos na equipe devem compreender as funções e significados dos componentes da arquitetura para que os membros da equipe possam escrever códigos que se encaixem no lugar certo na arquitetura de forma consistente. "Dependências" são frequentemente expressas neste diagrama entre pacotes para evitar acoplamentos indesejados ou dependências circulares. Do ponto de vista arquitetônico,

as dependências circulares entre pacotes são o problema pior e resultam em testes mais difíceis e um tempo de construção mais longo.

Modelos de domínio como diagramas de classe ou ER (entidade-relacionamento) / diagramas: Um Modelo de Domínio descreve a taxonomia de conceito do espaço do problema no qual o aplicativo funciona. No nível de comunicação humana, o vocabulário desse modelo de domínio deve se tornar a "linguagem ubíqua" usada em toda a comunidade de partes interessadas, incluindo usuários, especialistas de domínio, analistas de negócios, testadores e desenvolvedores. No nível de programação, o Modelo de Domínio também é essencial para selecionar nomes de construções de programação, como classes, dados, métodos e outras convenções. Uma grande parte da taxonomia conceitual (frequentemente chamada de "entidades") é mapeada em uma estrutura de dados persistente no banco de dados e geralmente tem uma vida útil mais longa do que o próprio aplicativo. Normalmente, o modelo de domínio (ou entidades) reside no pacote "M" na arquitetura lógica se você escolher uma arquitetura "MVC" para sua aplicação. Um diagrama ER é mais adequado para expressar um modelo de domínio porque está vinculado mais diretamente a bancos de dados relacionais. Observe também que esse modelo de domínio cresce com o tempo. Como o domínio está no cerne da compreensão e comunicação do problema, manter os modelos de domínio em crescimento na equipe (ou mais amplo, na comunidade) é muito importante. "

- JOBA, 2016, p.5-6

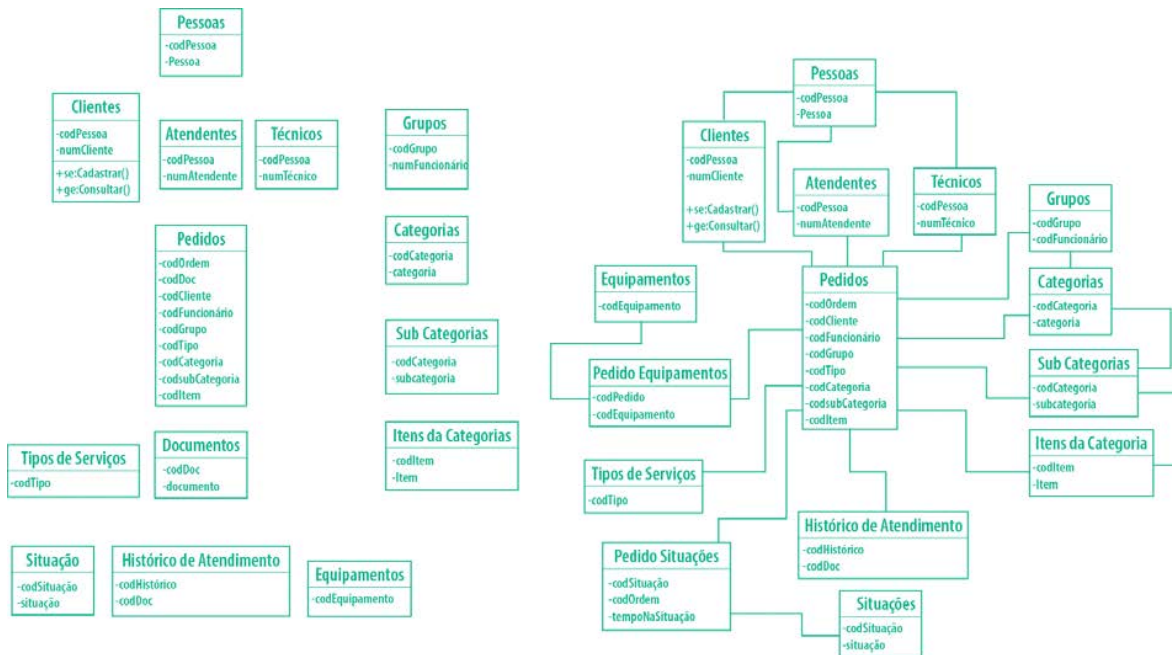


Figura 2 – Exemplo de Classes levantadas e ao lado Diagrama de Classes Completo

Backlog de Produto

Para recordarmos, um *backlog* de produto é uma lista priorizada de entregas e isso inclui novos recursos, que devem ser implementados como parte de um projeto ou desenvolvimento de produto de *software*. É um artefato de tomada de decisão que ajuda a estimar, refinar e priorizar tudo o que você pode querer concluir no futuro.

Isso ajuda a garantir que a equipe esteja trabalhando nos recursos mais importantes e valiosos, corrigindo os bugs mais importantes ou fazendo outro trabalho importante e crítico para o desenvolvimento do produto.

O *backlog*, portanto, é extremamente útil em situações em que você não consegue fazer tudo o que está sendo solicitado, ou em contextos em que mesmo uma pequena quantidade de planejamento ajudará muito. Muitos pensam nesta "lista" como uma lista de tarefas pendentes e a definem exatamente dessa forma, como uma lista de coisas que você deve fazer para entregar seu produto

de *software* ao mercado. Na verdade, não é necessariamente uma lista de tarefas pendentes. Pense nisso como uma lista de desejos.

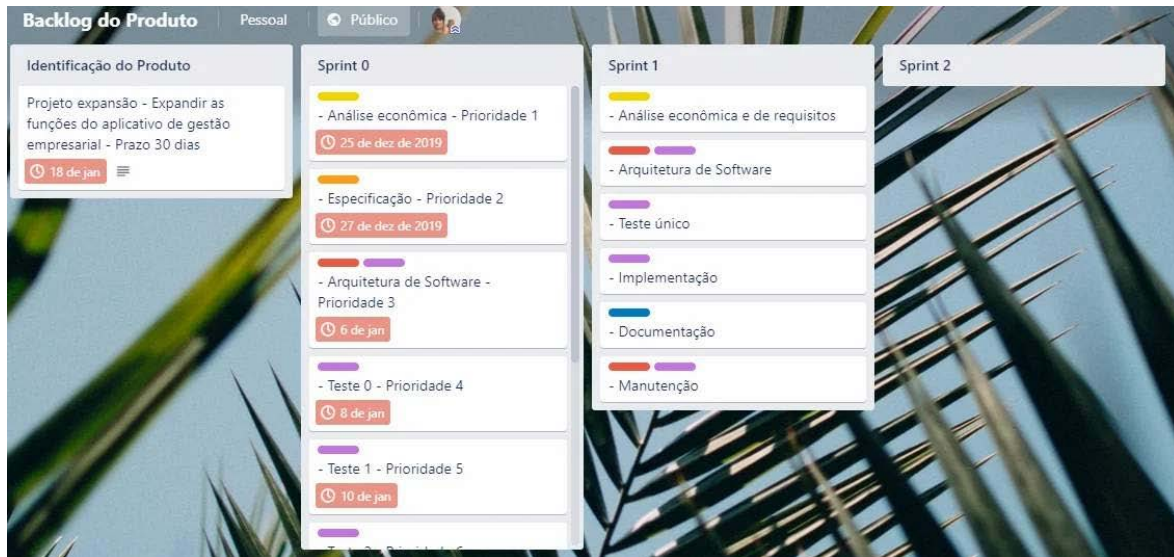


Figura 3 – Exemplo de *product backlog*

Fonte: Reprodução

Saiba Mais

Como você pode ver, é bom utilizar uma ferramenta para organizar o *backlog* do produto. Sugiro aqui que você use o *Trello* porque é simples, intuitivo, online, colaborativo e grátis.

Trello

ACESSE

Sprint Backlog

É o conjunto de itens que uma equipe multifuncional de produtos seleciona de seu *product backlog* para trabalhar durante o próximo *sprint*.

Normalmente, a equipe concorda com esses itens durante a sessão de planejamento do *sprint*. Na verdade, o *backlog* do *sprint* representa a principal saída do planejamento desse.

Se a equipe não for capaz de completar, ou mesmo começar certos itens do *sprint backlog* até o final do *sprint*, a equipe pode escolher adicionar esses trabalhos inacabados ao próximo *sprint backlog*, caso eles ainda forem considerados de alta prioridade, ou para o *backlog* do produto.

De acordo com a estrutura do *scrum*, toda a equipe ágil, incluindo o *scrum master*, o *product owner* e o time de desenvolvimento, compartilhará a propriedade do *sprint backlog*. Isso ocorre porque todos os membros da equipe trarão conhecimentos e percepções exclusivas para o projeto no início de cada *sprint*.

Sprint backlogs são geralmente planilhas embutidas, mas também podem ser desenvolvidas e mantidas em ferramentas de *software* projetadas para gerenciamento ágil de projetos. Como essas listas incluem apenas trabalhos que podem ser concluídos em um curto espaço de tempo, a que chamamos de *SPRINT*, que dura de 2 a 4 semanas, os *backlogs* de *sprint* costumam ser muito simples.

Diagrama de Atividades

Usamos Diagramas de Atividades para ilustrar o fluxo de controle em um sistema e fazer referência às etapas envolvidas na execução de um caso de uso. Modelamos atividades sequenciais e concorrentes usando diagramas de atividades. Portanto, basicamente representamos os fluxos de trabalho visualmente usando um diagrama de atividades. Um diagrama de atividades enfoca a condição do fluxo e a sequência em que ele acontece. Descrevemos ou representamos o que causa um determinado evento usando um diagrama de atividades.

Um diagrama de atividades é usado por desenvolvedores para entender o fluxo de programas em alto nível. Também permite que eles descubram restrições e condições que causam eventos específicos. Geralmente usamos o diagrama e a documentação textual para tornar a descrição do nosso sistema o mais clara possível.

TDD E Testes

Testes de Aceitação do Usuário: Eles devem incluir todos os cenários descritos nas histórias de usuário. Eles não devem ser muito detalhados (eles não precisam mencionar telas específicas ou uma lista completa de ações para executar as etapas). Devem ler:

DADA essa condição 1 e condição 2....
QUANDO eu faço a etapa 1 e a etapa 2...
ENTÃO, resultado desejado 1, resultado desejado 2....

Eles definem um conjunto de cenários reais que um testador pode percorrer para garantir que o recurso está completo.

Esses não são scripts de teste detalhados, pois o objetivo deles é transmitir um conjunto de testes que todos os envolvidos podem percorrer para entender como o recurso funcionará.

Vamos conhecer um pouco de TDD – Desenvolvimento Orientado a Testes. É uma abordagem evolutiva do desenvolvimento que requer disciplina e habilidades significativas e, claro, boas ferramentas.

A primeira etapa é adicionar rapidamente um teste, ou seja, um código básico o suficiente para falhar.

Em seguida, você executa seus testes, frequentemente, o conjunto de testes completo, ainda que, por uma questão de velocidade, você possa decidir executar apenas um subconjunto, para garantir que o novo teste de fato falhe.

Depois, você atualiza seu código funcional para que ele passe nos novos testes.

A quarta etapa é executar seus testes novamente. Se eles falharem, você precisará atualizar seu código funcional e testar novamente.

Depois que os testes forem aprovados, a próxima etapa é começar de novo, aproveite para refatorar qualquer duplicação de seu código conforme o necessário.

Wireframe, Mapa Conceitual e Mapa Navegacional

Wireframes: Uma imagem é necessária para cada tela envolvida. Eles podem ser desenhos simples em um quadro branco, que são digitalizados e postos no documento em *word*, ou um conjunto de caixas criadas em *softwares* para isso, e que depois podem gerar imagens para serem coladas no *word* ou algum outro *software* de documentação.

Alguns *softwares* para isso são:

Omni

A versão **trial** de 15 dias é mais que suficiente para você desenvolver todos os **wireframes** do desafio.

ACESSE

Balsamiq

A versão **trial** de 15 dias é mais que suficiente para você desenvolver todos os **wireframes** do desafio.

ACESSE

Visio

Nesse caso, veja no seu pacote de estudante (cada universidade tem o seu) se está disponível, ou se no **site** da **Microsoft** há versões livres para estudantes.

ACESSE

Na pior hipótese, você pode usar algum *software* livre do pacote BR Office para *Ubuntu*, o *Microsoft Paint*, ou ainda o *Power Point*. Caso, por fim, tenha conhecimento, pode fazer direto em HTML5, o que vai te poupar tempo depois na hora de codificar o *front* do projeto, já pensando no *CSS3 inclusive*.

As seguintes leituras são muito importantes para o desenvolvimento dos *wireframes* e mapas:

Guia sobre *wireframing* para iniciantes

LIN, W. Guia Sobre Wireframing Para Iniciantes. 2020.

ACESSE

Como fazer um Mapa Conceitual

ASSISTA

Aula 15. Mapa do Site

ASSISTA

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta disciplina:

LEITURAS

Orientações básicas na elaboração de um diagrama de classes.

Este artigo orienta o estudante na elaboração de um diagrama de classe, procurando estabelecer, de forma sintética, os principais pontos para a abstração dos objetos e classes de um cenário específico.

<https://bit.ly/3Dhp4cF>

Backlog do Produto – Passo a passo como construir e priorizar.

Backlog do Produto, ou *Product Backlog*, é uma lista ordenada de tudo o que é necessário para chegar ao produto de um projeto de desenvolvimento de software.

<https://bit.ly/3o2lVrj>

Diagrama de Atividades

Ilustra a natureza dinâmica de um sistema pela modelagem do fluxo de controle de atividade à

atividade .

<https://bit.ly/3o1OsP5>

Wireframes: O Que São e Como Criar O Seu (+ 10 Exemplos)

Wireframes são fundamentais se você tem um projeto na web. E a razão para isso está em uma palavrinha mágica chamada planejamento. Para criar um site ou aplicativo, você precisa de objetivos traçados e ações organizadas. Pois é justamente para isso que usamos *wireframes*.

<https://bit.ly/3E6Qluo>

Situação-Problema 1

Caro(a), estudante.

Agora, vamos compreender o cenário que será abordado na primeira situação-problema da disciplina.

Atente-se à situação profissional que você precisará entender para poder realizar a atividade.

Entender o Negócio

Era uma segunda-feira daquelas, trânsito, chuva e frio. Você chegou cedo, todavia, tinha dormido pouco porque seu smartphone havia tocado às 2h15 com o pessoal de negócios avisando que pela manhã haveria uma reunião para discutir a implementação de um novo sistema para um cliente que possui uma loja de *cup cakes gourmet* e que quer um aplicativo *mobile* para incrementar suas vendas pela *internet*.

Tudo bem, você entra na sala de reunião e vários *stakeholders* estão presentes, inclusive o cliente, com cara de poucos amigos, também pudera, na última empresa que ele contratou, os analistas não souberam elicitar os requisitos, desprezaram a opinião dele e construíram um APP que não funcionava; as reclamações dos clientes foram tantas que ele simplesmente retirou do ar e, claro, teve a imagem de sua empresa arranhada. Assim, isso vai demandar uma abordagem extremamente cautelosa e uma análise minuciosa.

Por isso seu diretor, logo de início, na parte de apresentações, já dispara seu nome como sendo o analista responsável por levantar os cartões de história dos usuários, transformar em requisitos ágeis completos e apresentar os produtos para a aprovação do cliente e sua equipe para evoluirmos para a fase de planejamento do sistema. Isso foi uma exigência do cliente porque como já escrevemos, na última vez em que ele tentou fazer esse APP, foi uma catástrofe atrás da outra.

O que se espera de você nesse projeto:

- Utilize esse documento como sendo o seu *briefing* descrevendo o negócio e as oportunidades, leia-o atentamente e anote o que for importante;

Stands ou Loja de Cupcakes: Quanto Custa, Receitas e como Montar

ACESSE

- A partir do texto e vídeo com as ideias e os comentários sobre o negócio, você deverá criar os cartões de história do usuário para criar um sistema para uma loja virtual de *cupcakes*. Foco na vitrine virtual, pedido eletrônico, pagamento e entrega – (na seção de **atividade de entrega**, você encontrará um arquivo que servirá de base para a atividade);
- Crie um mapa de afinidade das histórias dos usuários;
- Agrupe as histórias e crie um *backlog* de produto priorizado;
- Extraia os requisitos das histórias e crie tarefas para realizá-las;
- Apresente um documento contendo a sua especificação ágil para esse projeto.

Situação-Problema 2

Vamos compreender o cenário que será abordado na segunda situação-problema da disciplina.

Atente-se à situação profissional que você precisará entender para poder realizar a atividade.

Fazer Artefatos UML

O seu diretor e o cliente adoraram seu material de requisitos ágeis e deram sinal verde para que você prosseguisse com a próxima fase: a de construção dos artefatos básicos da UML e os casos de uso expandidos.

Nessa fase, eles esperam de vocês as seguintes entregas:

- Diagrama de caso de uso geral;
- Casos de uso expandidos;
- Diagrama de classes;
- Diagrama de sequência;
- Apresente o documento contendo todos esses documentos conforme a ordem.

Situação-Problema 3

Por fim, vamos compreender o último cenário, abordado na terceira situação-problema da disciplina.

Atente-se à situação profissional que você precisará entender para poder realizar a atividade.

Fazer Protótipo das Telas

Parabéns, as coisas andaram bem para o seu lado.

Seus colegas acabaram contando sobre seu conhecimento em desenvolver wireframes, mapas conceituais e mapas de aplicativos web para a diretoria e eles acabaram "ventilando" isso ao cliente que até o momento vem gostando do seu trabalho. Ele deu sinal verde para que você assuma agora o compromisso com a entrega de design da interface humano computador. Todos estão confiantes e agora é o momento de você mostrar sua criatividade aliando usabilidade, acessibilidade e uma excelente arquitetura de informação.

O que eles esperam que você entregue:

- *Mockup/wireframe* das telas do APP dos *cupcakes*;
- Mapa conceitual;
- Mapa navegacional (mapa do app).

Problema em Foco

Caro(a) aluno(a), vamos lá!

Temos 3 desafios que são sinérgicos e complementares entre si, portanto, 3 etapas do mesmo projeto.

A primeira parte é fazer você escrever as histórias do usuário a partir do texto e vídeo indicados (estão ambos no mesmo *link*), explore sua criatividade e imaginação. Com certeza, cada aluno criará o seu e não pode haver projetos iguais.

Os requisitos iniciais nos projetos Ágeis são obtidos nas escritas das *User Story* (histórias do usuário) e, conseqüentemente, as tarefas que serão executadas ou, nesse primeiro momento, conhecidas.

Cada história de usuário carrega consigo uma coleção de tarefas, onde a história descreve a necessidade do usuário e a tarefa descreve como a funcionalidade será implementada. Como a tarefa representa o trabalho real, teremos um nível de granularidade muito maior.

A definição das tarefas para cada história ocorre quando alocamos a história na iteração atual (*SPRINT* se estivermos falando de *SCRUM*) e isso é muito bom, pois teremos maior *feedback* e detalhes para assim melhor elaborar as tarefas a serem executadas para aquela história.

As tarefas são estimas em horas, é recomendado estimar o tamanho das tarefas entre 2-12 horas, para tarefas que requerem mais que 12 horas quebre estas em várias tarefas menores que 12 horas.

Veja um exemplo abaixo das entregas esperadas:

Copie e cole essa Tabela abaixo num documento do *Word*, por exemplo, e escreva as histórias neles, no mínimo umas 15, o máximo dependerá de quão preciso você quer ser nesse APP dos *cupcakes*.

Tabela 1 – Template de Cartão de História do Usuário

ID:
Título:
Requerente:
Ação:
Comentários:
Critérios de aceitação: CA#1...
Regras de negócio: RN #1...
Requisito não funcional: RN F#1...
Prioridade: [A] [B] [C] [D] [E]
Pontos de história:

Agrupe as histórias que você escrever por afinidade como você aprendeu.

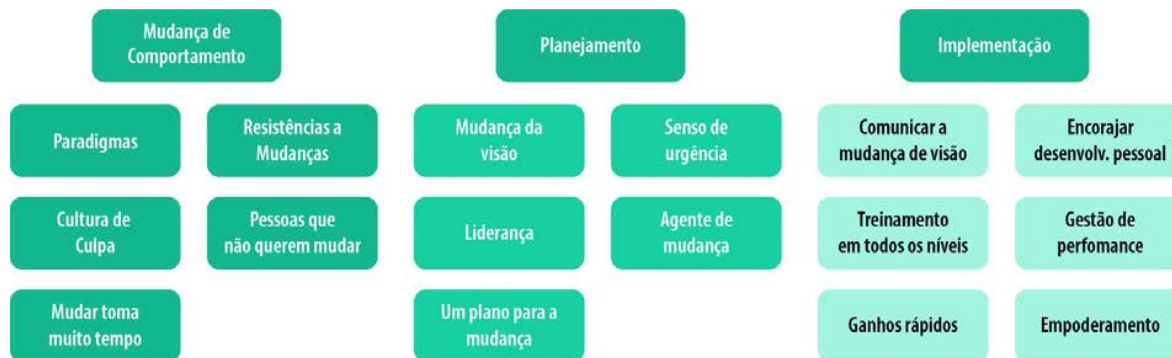


Figura 4

Veja que se trata de um agrupamento por eixo temático para concentrar cartões de história do usuário e facilitar a visualização do trabalho a ser feito.

Monte um *Backlog* de Produto já priorizando as histórias e colocando os cartões de história do usuário organizados por tema. Lembre-se, isso quer dizer que todos os cartões de história do bibliotecário comporão um tema, assim como, por exemplo, os cartões que se refiram a função do usuário (quem pega o livro emprestado) e demais funções.

O seu artefato de *backlog* terá a seguinte aparência e poderá ser feito até mesmo no Excel, se for o caso.

Tabela 2 –Template de backlog de histórias dos usuários

ID	História do usuário	Estimativa em pontos	Prioridade

Isso vai ficar mais ou menos assim:

ID	História do Usuário	Estimativa	Prioridade
7	Como um usuário não cadastrado eu quero criar uma conta (login e senha) para acesso.	3	1
1	Como usuário cadastrado eu quero efetuar login	1	2
1	Como usuário cadastrado eu quero efetuar logout	1	3
0	Criar script para apagar o banco de dados	1	4
9	Como usuário autorizado eu quero ver uma lista de itens para que eu possa selecionar algum	2	5
2	Como usuário cadastrado eu quero adicionar, apagar ou editar um novo item à lista e ele dever[a aparecer ou desaparecer na mesma	4	6
8	Como administrado eu quero ver uma lista de contas logadas	8	7

Organize as coisas, veja nesse exemplo o que esperamos que você faça para reconhecer as atividades envolvidas em cada cartão de história além dos artefatos acima.

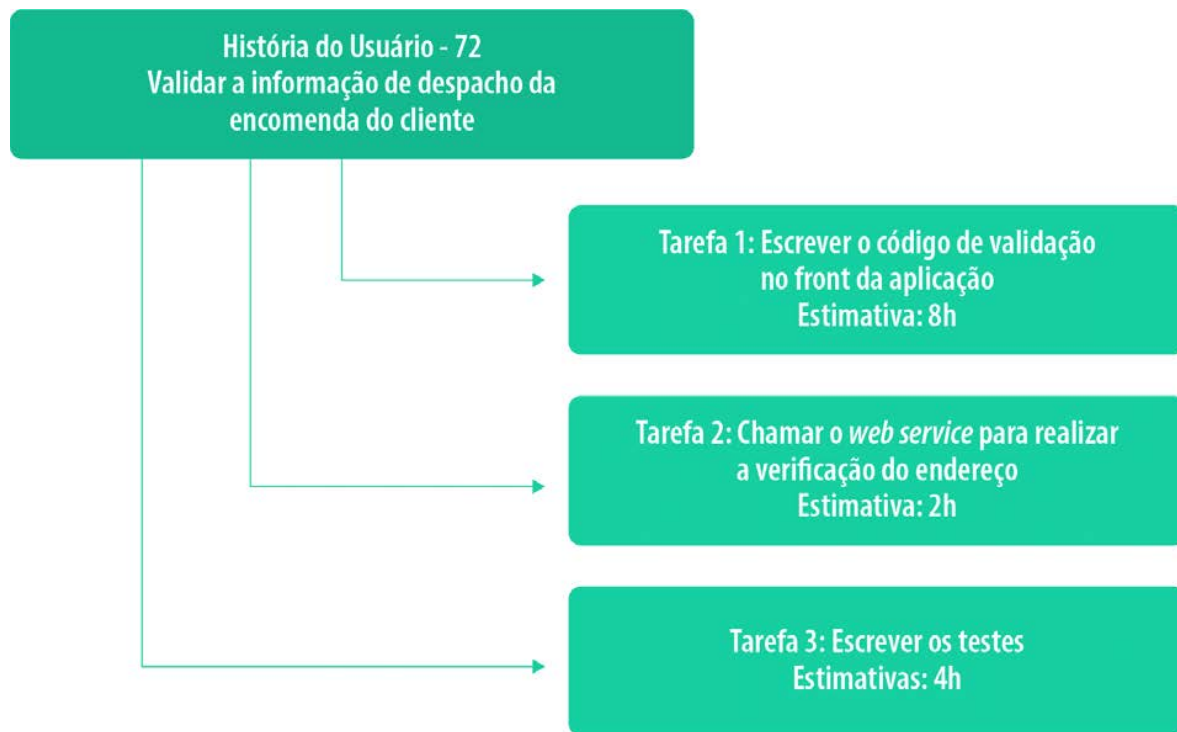


Figura 5

É apenas um exemplo, há dezenas de histórias de usuários num APP e centenas de tarefas. É delas que tiraremos as sprints depois.

Já para o segundo desafio, vamos trabalhar a base da UML com os diagramas essenciais.

Para ajudar você a se lembrar, aqui vai um exemplo para sua leitura, reciclagem e exemplo de elaboração de um diagrama de sequência bem completo, levando em consideração cenários normais e os cenários alternativos de forma simples.

Primeiramente, você deverá pensar no caso de uso geral do APP de *cupcakes*, abaixo coloco um exemplo desse com outro aplicativo para servir de inspiração.

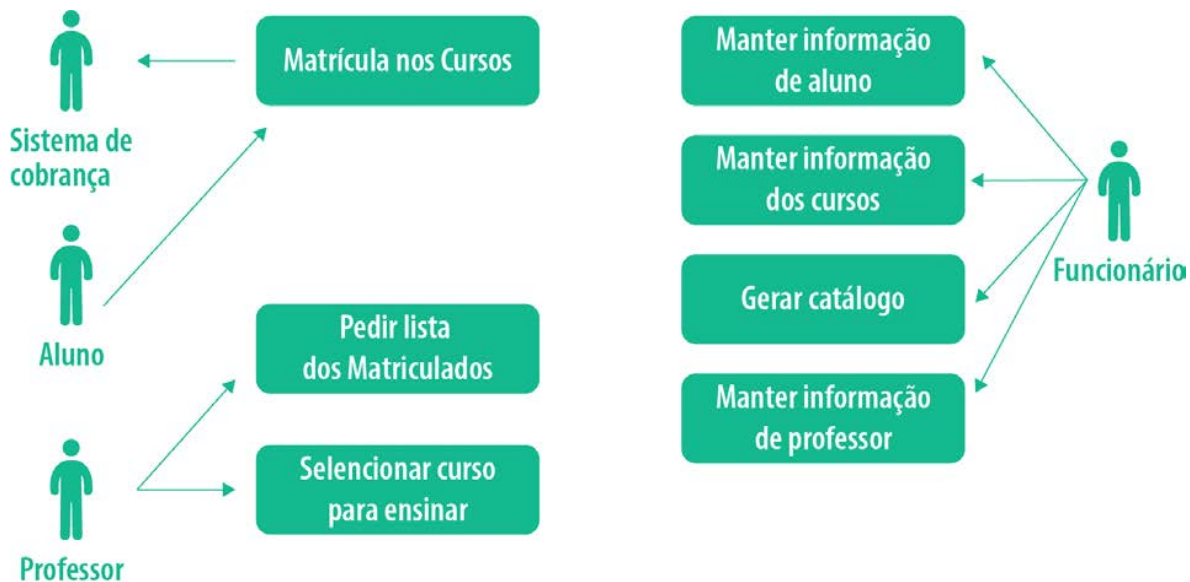


Figura 6 – Exemplo de caso de uso geral

Depois do caso de uso geral, você deverá expandir os casos de uso e descrevê-los como coloquei na parte teórica desse documento. Lembra do exemplo? Vejamos novamente.

Nome: inscrever-se em uma disciplina.

Identificador: #C 49.

Descrição: inscrever um aluno existente em uma disciplina existente desde que ele seja elegível.

Pré-condições: o aluno precisa estar regularmente matriculado e pagando pontualmente à Universidade.

Pós-condições: o aluno é matriculado na disciplina escolhida porque é elegível, pagante e havia vagas.

Caso básico de ação:

1

O caso de uso começa quando um aluno deseja se inscrever em uma disciplina;

2

O aluno insere o seu nome e Registro Geral de Matrícula (RGM) no sistema por meio da TELA101 – TELA DE LOGIN;

3

O sistema verifica se o aluno está qualificado para se inscrever na disciplina, de acordo com a Regra de Negócios (RN) 32 – determinar elegibilidade para matrícula na disciplina; <Curso Alternativo Alfa>

4

O sistema exibe a TELA102 – MENU DE DISCIPLINAS, que indica a lista de disciplinas disponíveis;

5

O aluno indica a disciplina em que deseja se inscrever; <Curso Alternativo Beta: o aluno decide não se matricular>

6

O sistema valida se o aluno está qualificado para se inscrever na disciplina de acordo com a RN 73 – determinar a elegibilidade do aluno para se inscrever na disciplina; <Curso alternativo Delta>

7

O sistema valida se a disciplina se encaixa na programação/trilha de aprendizagem existente do aluno de acordo com a RN 97 – validar programação/trilha da disciplina do aluno;

8

O sistema calcula as taxas da disciplina com base nas taxas publicadas no catálogo das disciplinas, os descontos de estudante aplicáveis e os impostos. Aplica as RN 143 – calcular taxas de alunos –, 107 – calcular descontos para alunos – e 59 – calcular impostos;

9

O sistema exibe as taxas via TELA189 – Exibir a tela de taxas da disciplina;

10

O sistema pergunta ao aluno se ainda deseja se inscrever na disciplina;

11

O aluno indica que deseja se inscrever na disciplina;

12

O sistema inscreve o aluno na disciplina;

13

O sistema informa ao aluno que a inscrição foi bem-sucedida por meio da TELA68 – Resumo da Matrícula da disciplina;

14

O sistema emite a fatura/boleto para o aluno pagar pela disciplina, de acordo com a RN 71 – faturar o aluno pela disciplina;

15

O sistema pergunta ao aluno se ele deseja um extrato impresso da matrícula que será disponibilizado após confirmação do pagamento;

16

O aluno indica que deseja uma declaração impressa;

17

O sistema imprime um *Portable Document Format* (PDF) da declaração de inscrição TELA39 – Relatório de resumo de inscrição (acessível somente após confirmação do pagamento);

18

O caso de uso termina quando o aluno pega a declaração impressa.

Curso alternativo Alfa: o aluno não é elegível para se inscrever nas disciplinas.

- **Alfa3:** o algoritmo determina que o aluno não está qualificado para se inscrever nas disciplinas.
- **Alfa4:** o algoritmo informa ao aluno que ele não pode se inscrever.
- **Alfa5:** o caso de uso termina.

Curso alternativo Beta: o aluno decide não se inscrever em uma disciplina disponível.

- **Beta5:** o aluno visualiza a lista de disciplinas e não encontrou aquela à qual ele deseja se inscrever.
- **Beta6:** o caso de uso termina.

Curso alternativo Delta: o aluno não possui os pré-requisitos para alguma disciplina.

- **Delta6:** o algoritmo determina que o aluno não é elegível para se inscrever na disciplina que ele escolheu.
- **Delta7:** o algoritmo informa ao aluno que ele não possui os pré-requisitos.
- **Delta8:** o algoritmo informa ao aluno sobre os pré-requisitos de que ele precisa para se tornar elegível para aquela disciplina.

- **Delta**: o caso de uso continua e é desviado para a linha 4 do curso básico de ação deste mesmo caso de uso.

Você deverá também fazer um diagrama de classes para persistência do APP dos *cupcakes*. Abaixo dou um exemplo que foi utilizado para modelar uma pizzaria express.

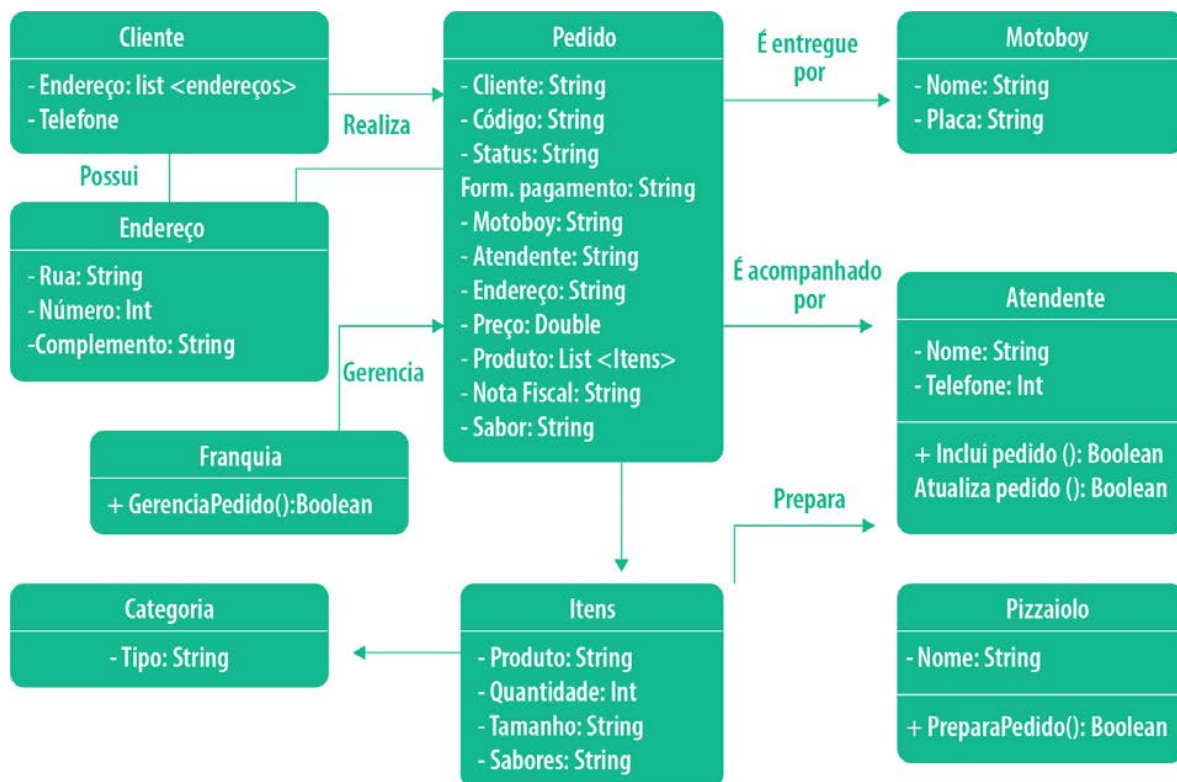


Figura 7 – Modelagem de classes de uma pizzaria express – Exemplo

O mais importante aqui é que você pense nas classes, organize os atributos e o tipo de dependência, se agregação, composição etc.

O diagrama de sequência é um elemento essencial no desenvolvimento do APP. Para ajudar você a se lembrar como faz, coloquei um outro exemplo. A partir dele desenvolva os diagramas para o APP do *cupcake*.

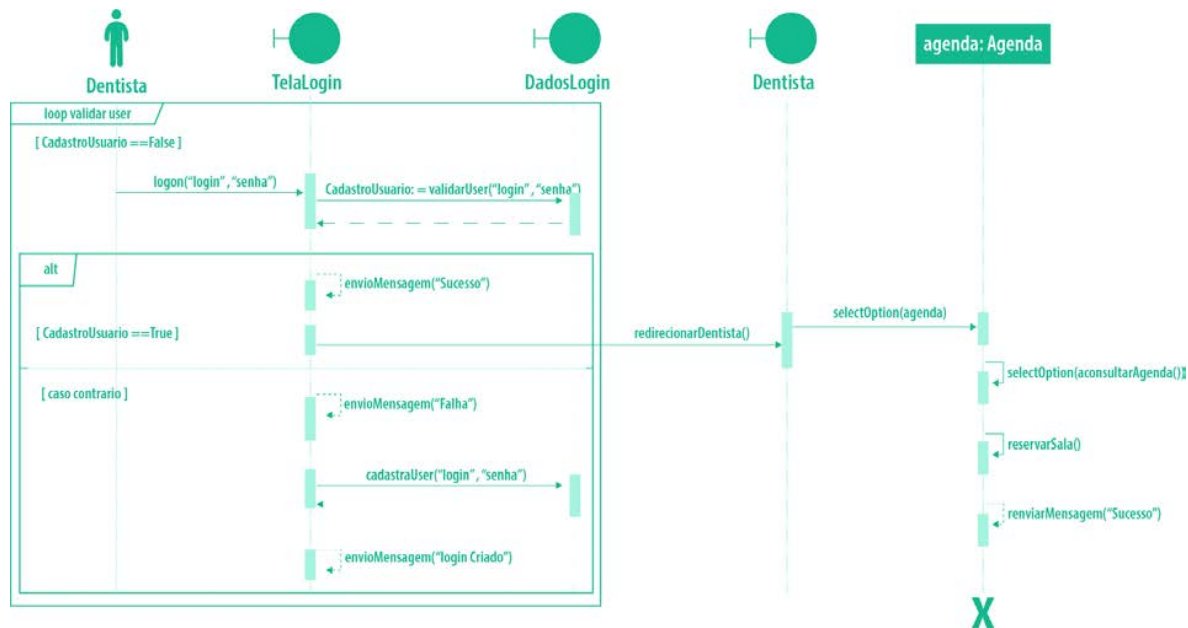


Figura 8

Para o terceiro desafio, o importante é criar a interface gráfica do APP dos *cupcakes* e apresentar protótipos de média resolução, isso incluirá as telas de mensagens de erro entre outras coisas.

Primeiramente, crie todos os *WIREFRAMES* de sua aplicação, ou seja, a "cara" do produto de seu projeto.

A ideia é que você complete todo o arcabouço arquitetônico visual de sua aplicação.

O ideal é que você gere um pdf ou uma apresentação ppt com os *wireframes* do APP.

Você pode utilizar o Draw.io gratuitamente on-line, e criar todos os *wireframes*. Faça isso para toda a aplicação, portanto, teremos muitos. Para tanto, utilize a opção *wireframe* do *website*.

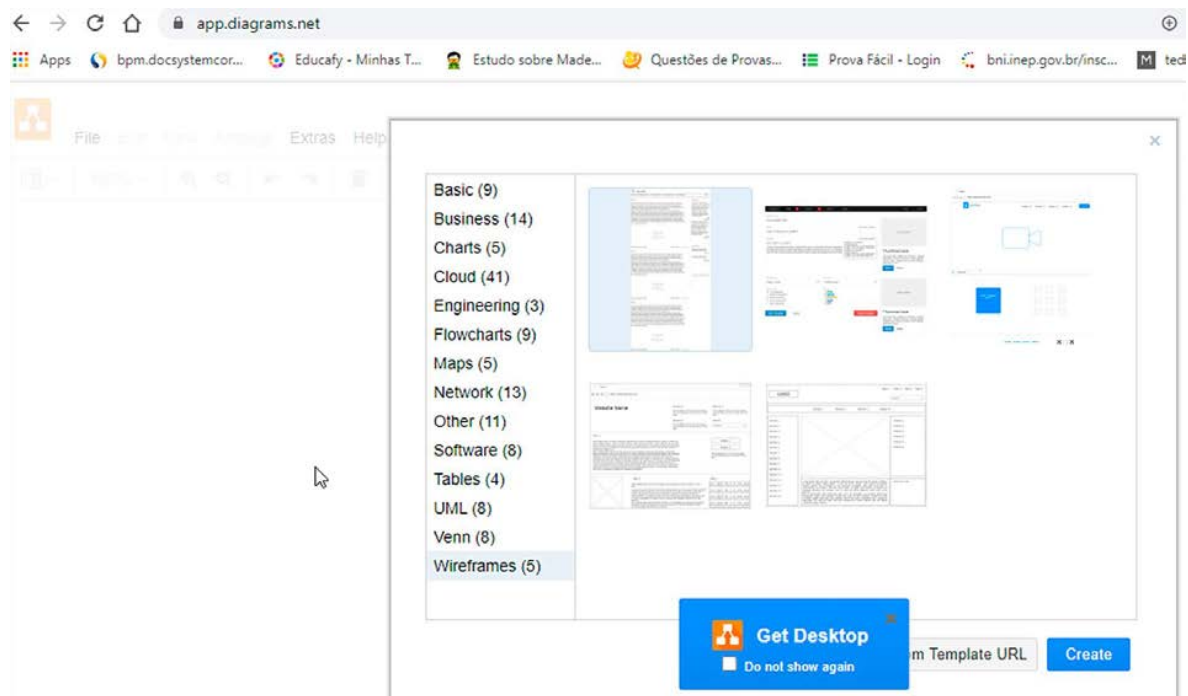


Figura 9 – Tela do Draw.io

Fonte: Reprodução

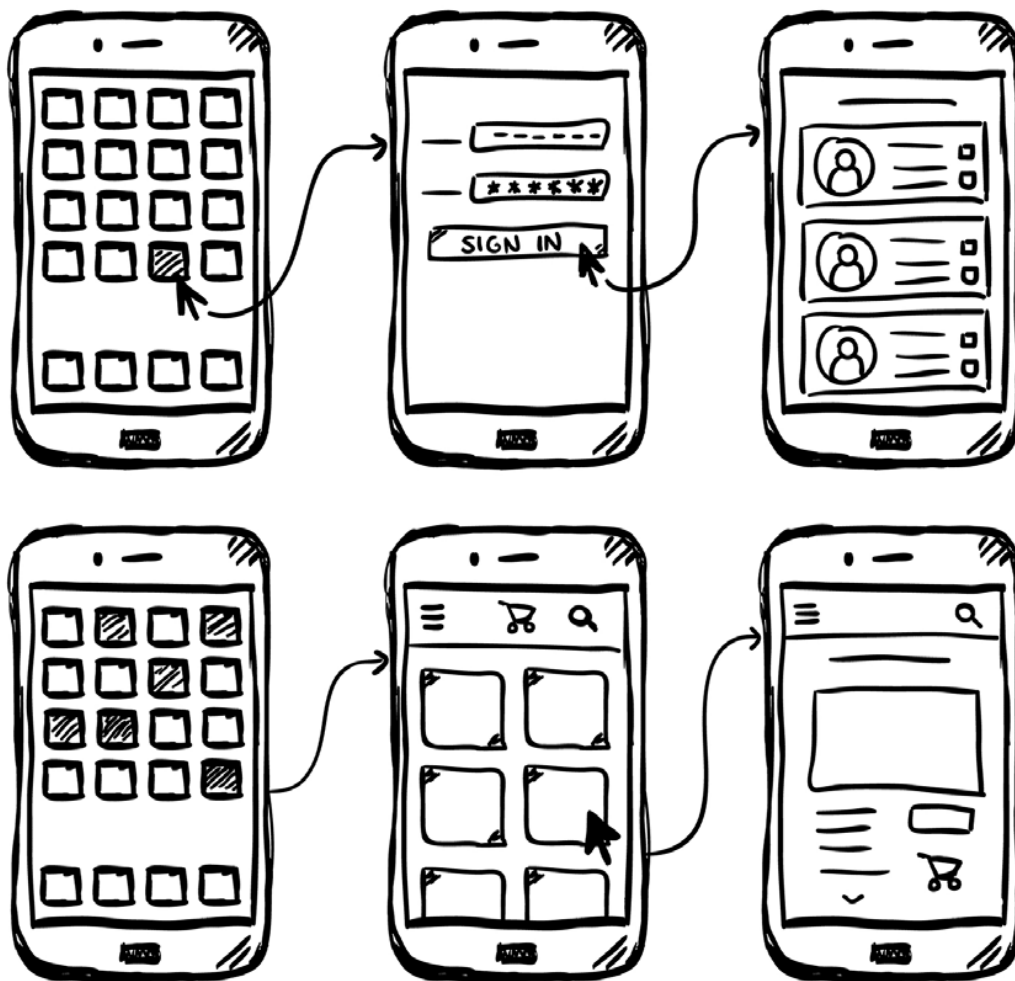


Figura 10 – Exemplo de *wireframe*

Fonte: Reprodução

Veja o exemplo abaixo que eu fiz para um aplicativo. A partir da página central, todas as outras vão para alguma outra página ou camada do APP. Isso é um tipo de mapa navegacional.

Mapa Navegacional

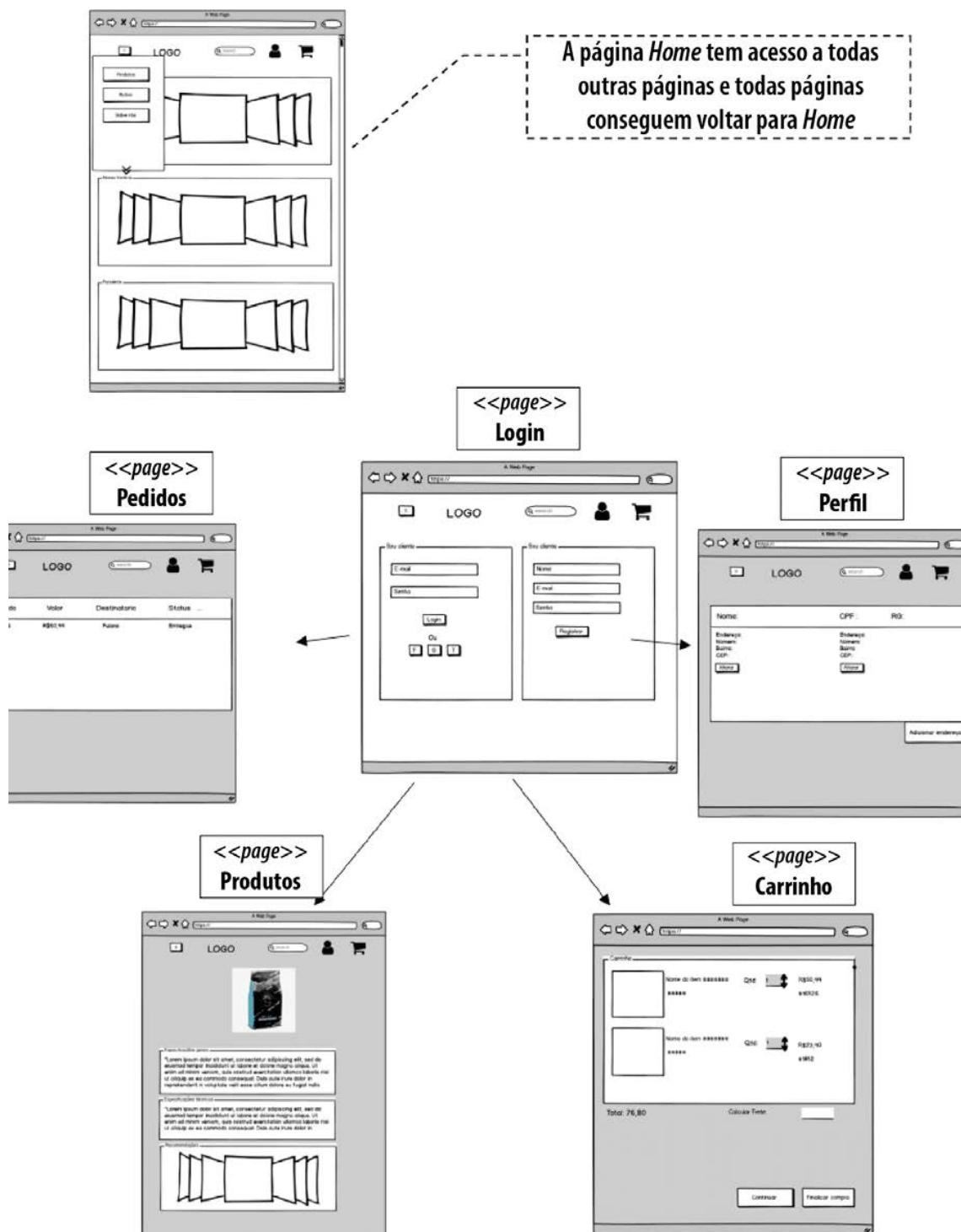


Figura 11 – Exemplo simples de mapa navegacional

Fonte: Reprodução

Caso você queira pode fazer em um nível um pouco mais profundo e sofisticado, como no exemplo abaixo, eu fiz para cada setor (tela de login à home page)

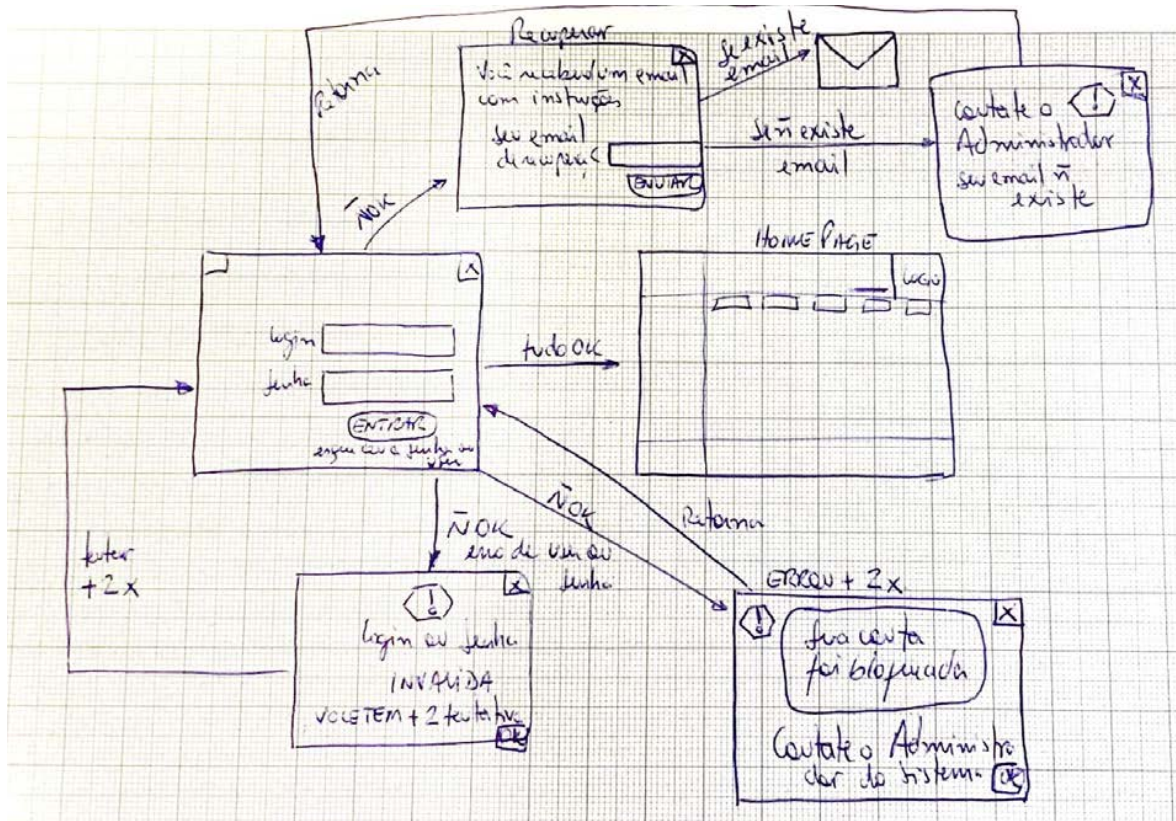


Figura 12 – Exemplo de mapa navegacional feito a mão

Fonte: Acervo do Conteudista

Depois disso, está na hora de preparar os *mockups* de média-alta qualidade já usando as cores para a aplicação. Gere um pdf ou uma apresentação ppt com eles.

Aqui logo abaixo deixo um exemplo do que você deve fazer, baseado no projeto.

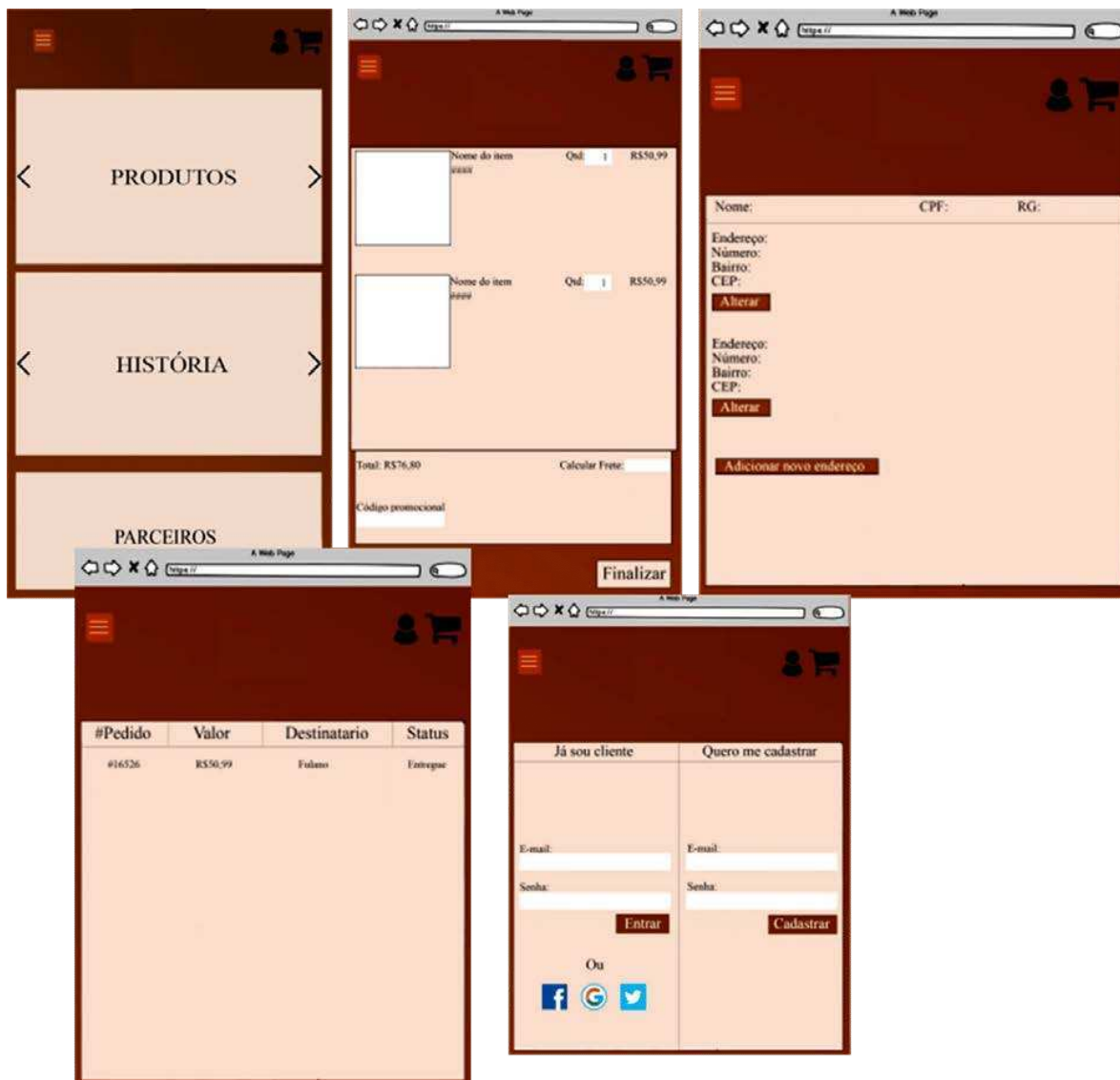


Figura 13 – Exemplo de *mockup* feito para aplicação.

Fonte: Acervo do Conteudista

Atividade de Entrega

Muito bem, estudante.

Agora que você já leu todas as situações-problema, você pode fazer o *download* [deste arquivo](#) para realizar a atividade de entrega.

Caso prefira, o mesmo arquivo também se encontra no Ambiente Virtual de Aprendizagem.

Como a atividade pretende a reunião de todas as etapas descritas nas 3 situações-problema, você, além de utilizar as tabelas disponíveis no arquivo, também deverá adicionar nele os demais arquivos solicitados para que, quando completo, você possa submeter no Ambiente Virtual.

Referências

JOBA, S. **Modelagem na era do Agile** - O que manter próximo ao código para escalar as equipes do Agile. 2016. Disponível em: <https://astahblog.com/2016/10/26/modeling-in-the-agile-age/> Acesso em: 10/09/2021.