

Projeto Integrador Transdisciplinar em Ciência da Computação I.



Conteudista: Prof. Me. Artur Marques

Revisão Textual: Prof.^a Dra. Luciene Oliveira da Costa Granadeiro

≡ 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 das situações-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, numa disciplina que preza pelo desenvolvimento prático de um projeto, é fazer com que possamos recordar o que aprendemos de forma a nos manter focados no desafio e ter praticamente tudo do que precisamos para desenvolvê-lo num lugar só, incluindo referências e sites relevantes de ferramentas.

Aqui também encontraremos leituras e material complementar que, sinceramente, alertamos: serão essenciais no aprofundamento e desenvolvimento deste desafio.

Muitos discentes comentam que os Recursos Humanos das 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 estes desafios, você terá a oportunidade de mostrar que desenvolveu o aprendizado e demonstrar que está pronto para o que se apresentar, concorrendo de igual para igual, com qualquer candidato em qualquer processo seletivo dentro da sua área de conhecimento.

Vamos aproveitar para recordar agora?

Modelos de Negócios

Você deve ter percebido que o modelo de negócio é fundamental porque, a partir dele, montamos as engrenagens funcionais de uma organização. Então o que é o modelo de negócios?

Bem, aqui, as definições poderiam vir de gigantes como Peter Drucker, em sua Teoria dos Negócios, literatura canônica obrigatória a qualquer estudante ou empresário, ou descrever o que Michael Lewis definiu como modelo de negócios, antes do estouro da bolha da internet, ou ainda, mais recentemente, com Joan Magretta e sua teoria de que os modelos de negócios não passam de histórias que tentam explicar como as empresas funcionam, focada mais nas premissas do que em apenas fazer dinheiro. Assim como Drucker, Lewis, por sua vez, é mais direto: o negócio tem que fazer dinheiro.

Porém, mais recentemente, tem Alex Osterwalder, com o seu popular modelo *Canvas* de Negócios, que tem se espalhado pelas *startups* ao redor do mundo todo.

No fundo, um modelo de negócios é uma grande árvore de decisões onde escolhemos caminhos, navegando pelo mar de incertezas e riscos. Essa travessia cheia de tormentas pelo mar estatístico das incertezas pode nos levar ao outro lado "se o mercado ajudar", à riqueza, à sobrevivência ou ao fracasso retumbante.

Uma definição mais direta pode ser encontrada em Kriss (2020):

"Um modelo de negócio é um esboço de como uma empresa planeja ganhar dinheiro com seu produto e base de clientes em um mercado específico. Em sua essência, um modelo de negócios explica quatro coisas:

- Que produto ou serviço uma empresa venderá;
- Como pretende comercializar esse produto ou serviço;
- Que tipo de despesas ela enfrentará;
- Como espera obter lucro.

Como existem tantos tipos de negócios por aí, os modelos de negócios estão mudando constantemente - e embora discutiremos alguns dos tipos mais comuns abaixo - não existe um modelo único que possa ser aplicado a todos os negócios."

- KRISS, 2020, p. 1

Vamos focar em um exemplo de modelo de negócio digital para que as coisas fiquem claras para você depois de tantas definições.

"Um exemplo popular de modelo de negócios é o modelo de assinatura - no qual as empresas cobram uma taxa de assinatura (mensal, anual etc.) para que os clientes

acessem um serviço. Vamos usar o *Netflix* como exemplo, vamos dividi-lo com base em quatro perguntas simples:

- Que tipo de produto ou serviço uma empresa venderá: a *Netflix* vende um serviço de *streaming online*;
- Como pretende comercializar esse produto ou serviço: a *Netflix* usa uma estratégia de marketing multicanal e comercializa seu serviço por meio de mídia social, *marketing* por *e-mail*, publicidade e até mesmo *marketing* boca a boca;
- Que tipo de despesas ela enfrentará: Como uma empresa Fortune 500, as despesas da *Netflix* são extensas, mas talvez notavelmente, suas despesas incluirão os custos de produção ou aquisição do conteúdo em sua plataforma, bem como a tecnologia e a equipe necessária para manter o serviço;
- Como espera obter lucro: embora a *Netflix* seja uma empresa tão grande (e tenha algumas maneiras diferentes de ganhar dinheiro), ela espera obter lucro com as vendas de assinaturas.

Em última análise, então, é imperativo entender como sua empresa vai ganhar dinheiro - e dinheiro suficiente para permanecer lucrativa após a inicialização e os custos adicionais são considerados na equação - este é o seu modelo de negócios. "

- **KRISS, 2020, p. 3**

Então, vamos encarar o modelo de negócios Canvas!

Ele tem a forma de um gráfico visual com diversos elementos que descrevem a proposta de valor de uma organização, sua infraestrutura, mercado e finanças. O objetivo é auxiliar as empresas a alinharem suas atividades por meio da ilustração de possíveis *trade-offs* (compromissos/trocas/negócios), nesse caso, de negócios.

Podemos entender esse termo como compensação, perde-ganha, pois sempre numa troca comercial há esse balanceamento, uma parte oferece algo em troca de algo que precisa a outra parte, de tal forma que essa relação pretende ser a mais justa possível. O que está em jogo é, em outras palavras, dinheiro por um bem ou dinheiro por um serviço de forma tradicional. Claro que essas relações não se encerram nessas duas possibilidades.

O intuito do Canvas – vamos chamá-lo assim daqui por diante – é visualizar todos os blocos de construção quando você deseja iniciar um negócio, incluindo clientes, rota para o mercado, proposição de valor e finanças.

Lembre-se de que não estamos aqui fazendo apenas mais um trabalho acadêmico para você ganhar nota e ser aprovado. Na verdade, nossa preocupação é com seu futuro e nada melhor do que criá-lo aqui e agora. Pense e pesquise em algo disruptivo, que lhe toque profundamente e que você queira fazer de verdade. Esta é sua oportunidade.

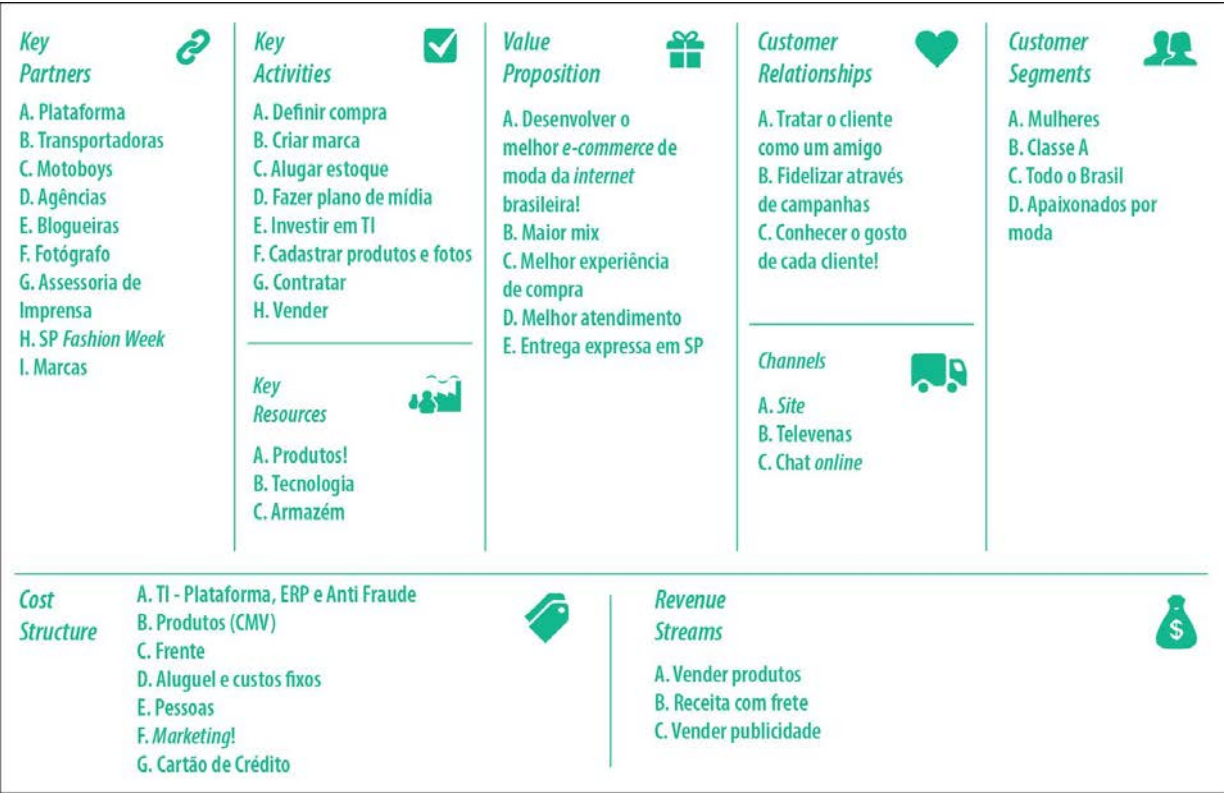


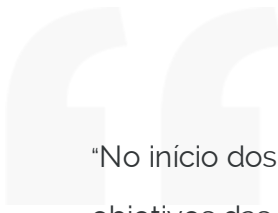
Figura 1 – Exemplo de Canvas

Os blocos do Canvas são:

- **Segmentos de clientes:** quem são os clientes? O que eles acham? Ver? Sentir? Fazer?
- **Proposições de valor:** o que é atraente sobre a proposição? Por que os clientes compram e usam?
- **Canais:** como essas propostas são promovidas, vendidas e entregues? Por quê? Está funcionando?
- **Relacionamento com o cliente:** como você interage com o cliente durante sua "jornada"?
- **Fluxos de receita:** como a empresa obtém receita com as propostas de valor?
- **Atividades principais:** quais são as coisas estrategicamente únicas que a empresa faz para entregar sua proposta?
- **Recursos-chave:** quais ativos estratégicos exclusivos a empresa deve ter para competir?
- **Parcerias-chave:** O que a empresa não pode fazer para se concentrar em suas atividades-chave?
- **Estrutura de custos:** quais são os principais direcionadores de custos do negócio? Como eles estão vinculados à receita?

Você pode optar e fazer uma modelagem utilizando *Design Thinking* ao invés do Canvas de Negócios. É centrado no ser humano, o que significa que ele usa evidências de como os consumidores, ou seja, gente realmente se envolve com um produto ou serviço, em vez de como outra pessoa ou uma organização pensa que eles se envolverão.

Vamos entender o contexto.



"No início dos anos 70 do século XX, fabricar produtos mais baratos era um dos grandes objetivos das empresas. Se você projetou um produto com um preço de custo mais baixo, é muito provável que domine o mercado. Uma década depois, as empresas se concentraram em tornar esses produtos melhores. No entanto, no início dos anos 2000, o foco era fazer produtos melhores, aprimorando seus recursos, design e usabilidade. Nos últimos tempos, o foco das empresas é fazer produtos melhores com pessoas para pessoas. Isso significa improvisar produtos com dados reais para fornecer aos usuários uma solução de que eles realmente precisam."

- KHAN, 2015, p. 1

Por que isso é importante? Bem, porque praticamente tudo que é ofertado ou vendido na *web* passou durante seu processo de criação por essa metodologia. Fora isso, o seu empregador atual ou futuro aposta que você sabe fazer isso. Então isso parece importante, afinal, 10 entre 10 universidades, cursos livres e gurus em geral tentam ensinar pessoas a adotarem DT – como vamos chamar daqui por diante em sua vida profissional e pessoal.

Ele é a chave que favorece uma movimentação rápida para colocar protótipos para teste, em vez de pesquisas ou planejamentos intermináveis. Em contraste com a solução de problemas tradicionais, que é um processo linear de identificação de um problema e, em seguida, o *brainstorming* de soluções, o DT só funciona se for iterativo. É menos um meio de chegar a uma solução única e mais uma maneira de desenvolver continuamente seu pensamento e responder às necessidades do consumidor. O melhor é que DT permite que as organizações criem valor duradouro para os consumidores.

Pense: seus concorrentes que você nem sabe quais são na *web*, atuando em seus negócios digitais, estão exatamente neste momento, usando DT para fazer com que seu cliente vá para eles!

Os motivos podem ser porque o DT:

- Visa resolver uma necessidade humana concreta;
- Resolve problemas ambíguos ou difíceis de definir;
- Leva a soluções mais inovadoras;
- Faz com que as organizações funcionem com mais rapidez e eficiência.

DT possui 5 grandes fases, mas é importante observar que essas fases, estágios ou modos nem sempre são sequenciais, não precisam seguir nenhuma ordem específica e, muitas vezes, podem ocorrer em paralelo e se repetir de forma iterativa.

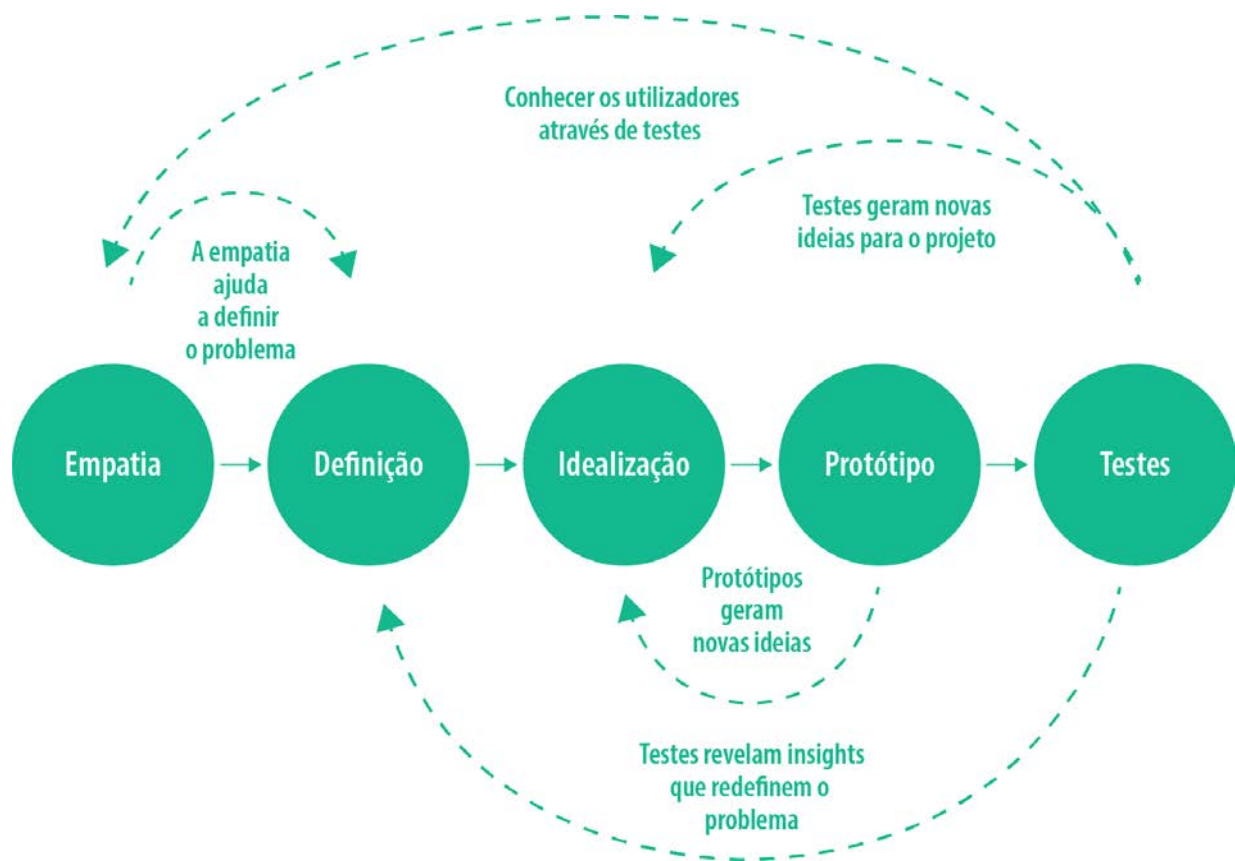


Figura 2 – Processo do DT – *Design Thinking*

Requisitos Ágeis

Primeiramente, algumas constatações. Muitas vezes, no início de carreira é comum ver projetos com dificuldades, com requisitos mal elaborados, muitas vezes, sem sentido. Isso nos faz sentir inúteis. Dá-nos a impressão que os requisitos não são nada mais do que um quadro com um monte de notas e *post-its*. Por exemplo, quando se é desenvolvedor e há muitas dúvidas, em geral, todas as pessoas envolvidas, clientes, usuários, desenvolvedores, analistas de negócios, arquitetos de solução, inclusive, os testadores não têm um bom entendimento do sistema como um todo, nem sobre quais são as várias personas que usam o sistema. Numa situação assim, é evidente que falta também que todas essas pessoas se apropriem de verdade do conhecimento e dos processos do negócio.

Então respire fundo e se prepare para pensar e analisar muito em profundidade, pois analistas de sistemas "criados no raso" jamais enfrentarão grandes tempestades e ondas avassaladoras, acredite. Mas aqui vão dicas preciosas para você desenvolver seu trabalho com menos incerteza.

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 de 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 etc.

Bons requisitos precisam de:

- Histórias de usuários;
- Testes de aceitação do usuário;
- Fluxo de trabalho;
- Detalhes dos requisitos;
- *Wireframes* ou protótipos rápidos.

Sem nenhuma das seções mencionadas, os requisitos começam a perder valor e sentido. Muitas vezes, quando somos chamados para consultoria, podemos ver 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 entendermos a língua que está escrita, não é mesmo? Cada seção de análise traz muito para a mesa e, muitas vezes, são situações julgadas como "perda de tempo", porém, quando vamos codificar, fazem toda a diferença.

Veja:

Histórias de usuários: indicam todos os cenários dos usuários envolvidos. O padrão mundial disso é:

Como algum papel, eu quero fazer alguma coisa, para que posso obter alguns benefícios.

Ou seja,

Como **gerente de contas a receber**, **eu quero** ter uma relação de devedores por safra (meses vencidos 0d, 60d, 90d, 180d, 360d), **para que possa** encaminhar às empresas de cobrança por seguimento e especialidade delas.

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. Você pode pensar: "mas sou aluno!". Sim, todos nós somos ou fomos alunos, 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 - Interface Humano Computador 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, por meio do recurso, podem ser bastante complexos, e é difícil explicar os detalhes na próxima seção.

Veja uma lista de ferramentas de software para você usar na construção de sua aplicação. Elas o ajudarão a explicar o sistema por meio de fluxos e vão apoiar no desenvolvimento. Suas versões demo (demonstração) vão permitir que você as utilize durante o tempo deste desafio tranquilamente, portanto, não "durma no ponto". São todas amigáveis e não precisam de curso para usar. Use e abuse:

- [Giffy](#);
- [Creately](#);
- [Lucidchart](#);
- [Canva](#);
- [Diagrams](#).

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 das telas envolvidas. Por estar no contexto do *wireframe*, 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, se é 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 o que você acha que o sistema deve fazer.

A documentação pode vir mais tarde, se você realmente precisar dela. Para seu modelo de requisitos inicial, por experiência, entendemos 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:** um modelo de domínio 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. Esse 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. (consta na 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 consta no tópico *Wireframe* deste material).

Mas, afinal, que nível de detalhe você realmente precisa?

Sabemos 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 este 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 para 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 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, porque 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 isto, num caso de uso mais aprofundado, 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 Universidade.

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

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 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 a 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;
- **Deltag:** O caso de uso continua e, é desviado para a linha 4 do curso básico de ação deste mesmo caso de uso;

Bem, aqui você percebeu como é importante ser analítico, crítico e lógico nesta nossa profissão. A clareza e declaração sem ambiguidade é fundamental 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 modelar coisas que não refletem o que as partes interessadas realmente precisam.

Portanto, 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 um desenvolvedor saber o que ele tem que fazer se nem as regras de negócios e nem as telas da interface do usuário (camada de apresentação) 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: [interface do usuário, *front end* e *back end*] são grandes atividades que devem trabalhar seguindo padrões e metodologias porque uma depende da outra.

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

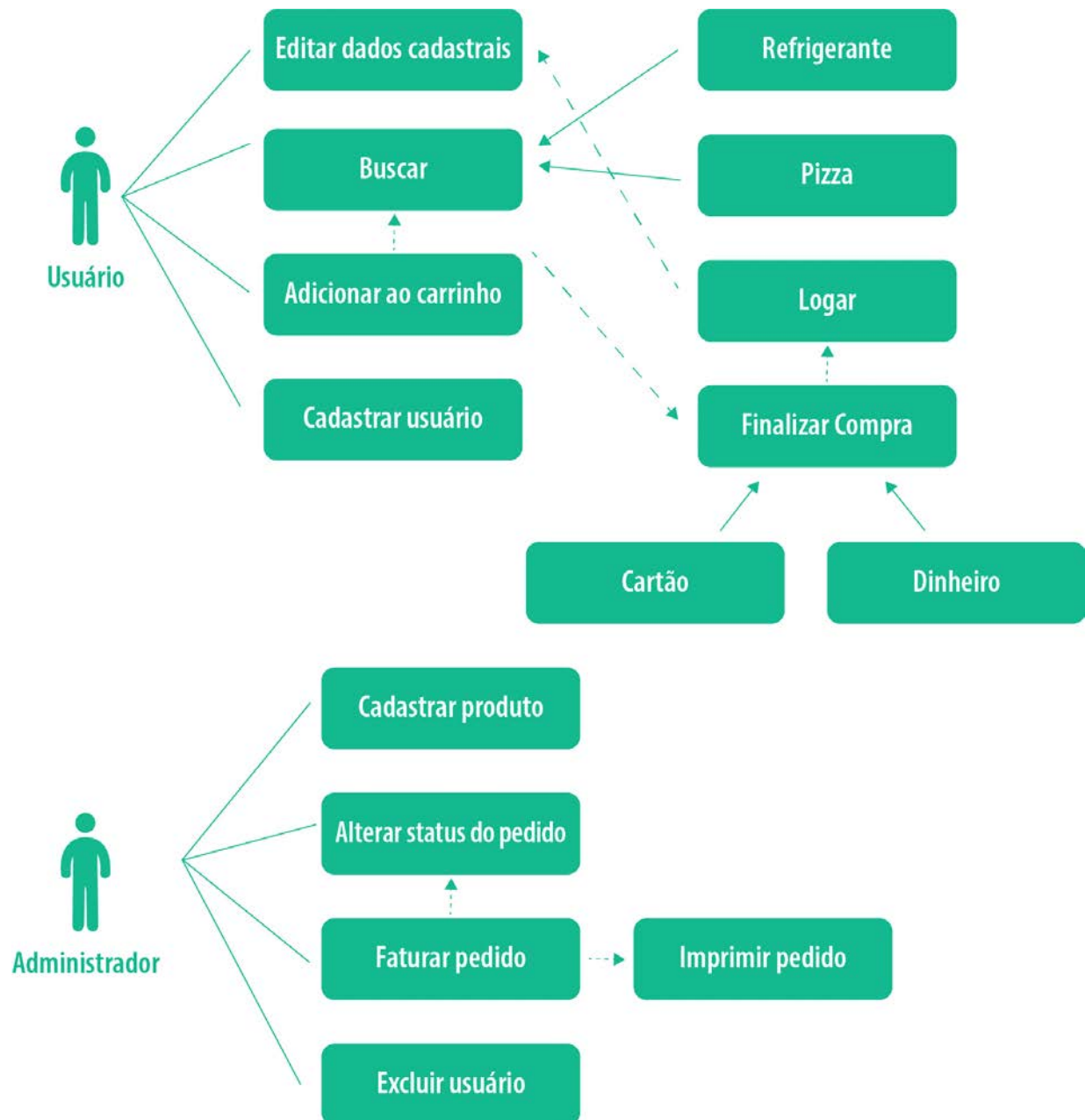


Figura 3 – Diagrama de Caso de Uso Macro de uma Pizzaria

Recomendamos 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 "bicho tem cabeça tronco e membros".

Tome cuidado porque há desafios no processo de levantar requisitos e converter em artefatos. Abaixo, listamos alguns, dê bastante atenção a todos eles para não cometer estas 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 (chamamos isso em tecnologia 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, precisa de tempo e de 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* e 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

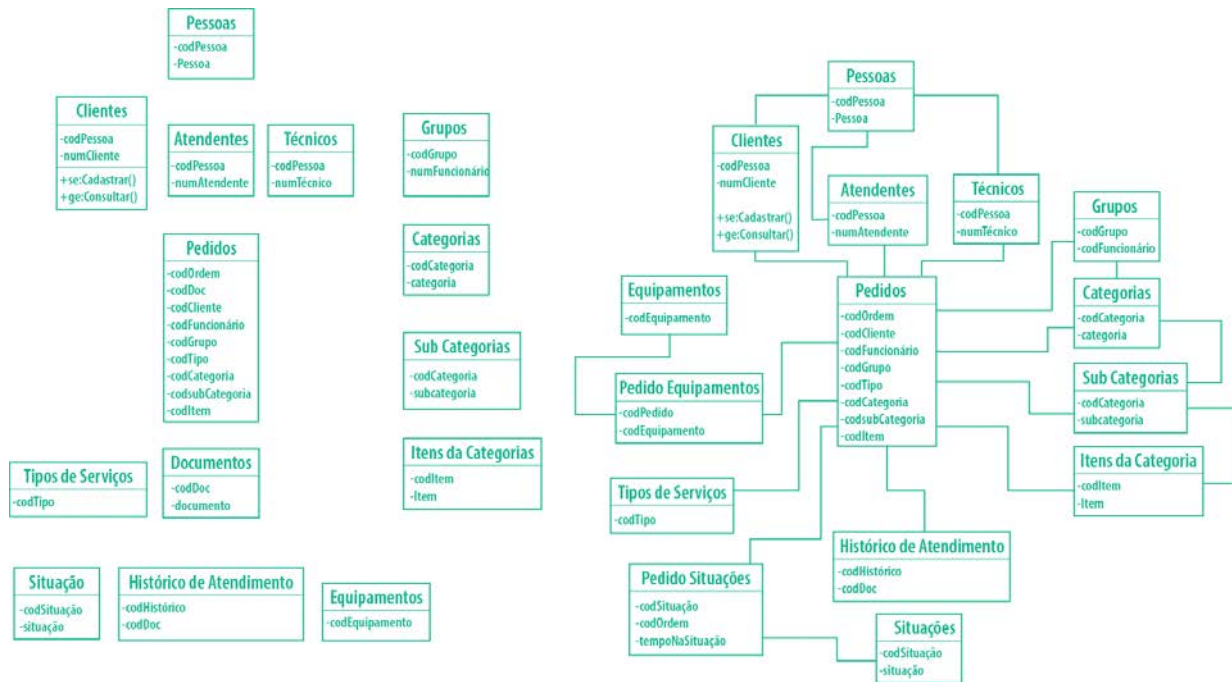


Figura 4 – Exemplo de Classes levantadas e ao lado Diagrama de Classes Completo

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 nessa "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.

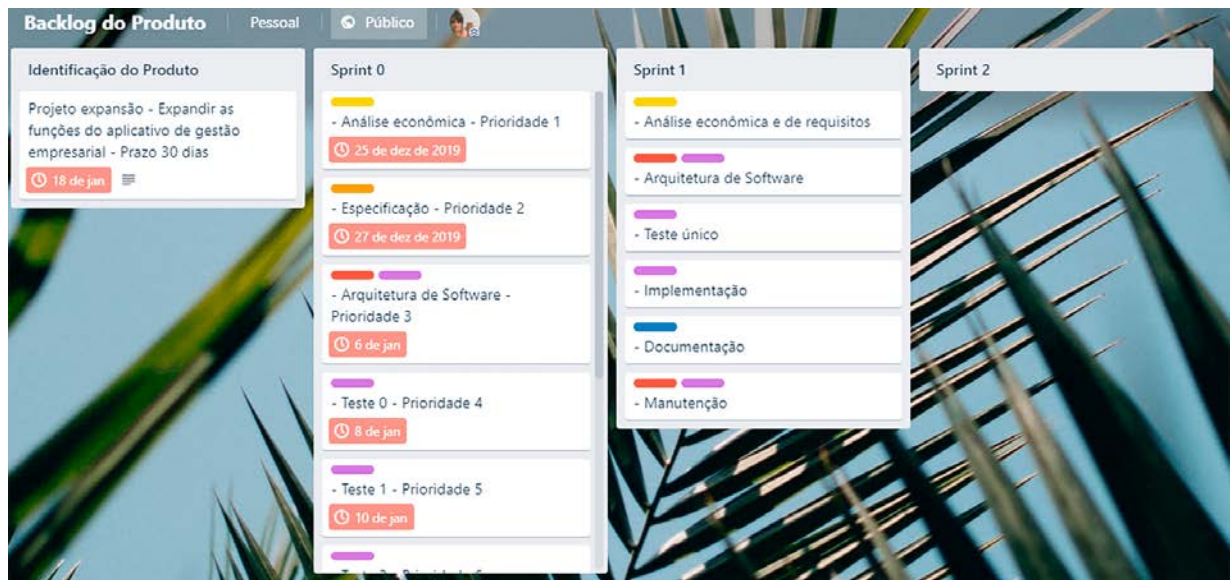


Figura 5 – Exemplo de *Product Backlog*

Fonte: Reprodução

Explore

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

Trello

Clique no botão para conferir o conteúdo.

Sprint Backlog

É o conjunto de itens que uma equipe multifuncional de produto 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 do *sprint*.

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, isso inclui o *scrum master*, *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 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 – Test Driven Development 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, 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 habilidade 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, embora, 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.

Explore *Web*

Alguns *softwares* para isso são:

Omnigroup

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

Microsoft Visio

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

ACESSE

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

Leitura

A seguinte leitura é muito importante para o desenvolvimento dos *wireframes*:

Guia sobre *Wireframing* para Iniciantes

Clique no botão para conferir o conteúdo.

ACESSE

Vídeos

Os vídeos a seguir são importantes para o desenvolvimento de mapas:

Como Fazer um Mapa Conceitual

Um mapa conceitual representa visualmente as relações entre ideias. Na maior parte das vezes, os conceitos são retratados como círculos ou caixas unidos por linhas ou setas que contêm palavras para demonstrar como as ideias se conectam.

Clique no botão para conferir o conteúdo.

ASSISTA

Clique no botão para conferir o conteúdo.

ASSISTA

Prototipagem

Por fim, precisamos escrever um pouco sobre prototipagem, já que o desafio requer que você apresente um protótipo de nível intermediário para entendermos a IHC e os elementos de arquitetura de informação presentes para que avaliemos a usabilidade entre outras coisas.

Quando descrevemos a prototipagem no desenvolvimento de *software* estamos nos referindo ao processo de construção de uma interface de usuário simulada para fins de idealização, avaliação e *feedback* do usuário. Os protótipos de *software* modernos são interativos. Eles imitam o mais próximo possível o comportamento real do *software*.

Um protótipo permite que nossos desenvolvedores e *designers* mostrem nossa compreensão dos requisitos do cliente. Se a nossa interpretação dos requisitos estiver um pouco errada, o cliente perceberá a discrepância e nos informará imediatamente. Como já dissemos, é importante compartilhar com os colegas e submeter a críticas!

Passe para os outros verem e espere dos outros que eles também passem os deles para que sejam criticados. Isso tem tudo a ver com *Design Thinking*.

Fazer isso no início do processo de desenvolvimento economiza tempo, dinheiro e aborrecimento.

Existem quatro tipos de prototipagem:

- Prototipagem rápida (descartável);
- Prototipagem evolutiva;
- Prototipagem incremental;
- Prototipagem extrema.

Para este projeto, estamos nos referindo apenas à primeira que explanamos rapidamente para lhe dar um norte.

A prototipagem rápida é a mais comumente usada. Seu nome se refere à facilidade e rapidez com que um protótipo pode ser modificado para experimentar diferentes ideias com o público do usuário e incorporar seus comentários. Ela também é conhecida como, prototipagem descartável, porque espera-se que o protótipo seja relevante apenas no curto prazo, como um *sprint* na estrutura ágil.

Pode passar por vários ciclos de *feedback*, modificação e avaliação durante esse tempo. Quando todas as partes interessadas estão satisfeitas, torna-se uma referência para *designers* e desenvolvedores usarem.

A prototipagem descartável também pode ser aplicada a protótipos de papel, nos quais os *designs* são simulados em pedaços de papel ou papelão. São, por definição, descartáveis.

Então podem ser usados para essa finalidade.

- Sequência de navegação feita em telas estáticas postas em uma apresentação do *PowerPoint* ou *Prezi*;
- Papéis contendo as telas, incluindo as de mensagem de erros, apresentada pelos analistas de sistemas;
- Ferramentas automatizadas de fluxos de IHC;
- Modelos “grossos” feitos em HTML com CSS – (mais usado, porque poupa tempo do desenvolvimento do front da aplicação).

Leitura

Aqui, para ajudar na prototipagem rápida de soluções de IoT ou Robótica, sugerimos as seguintes leituras para sua orientação:

Internet das Coisas: da Teoria à Prática

Clique no botão para conferir o conteúdo.

ACESSE

Prototipação Rápida de Aplicações Orientadas à Contexto em Cidades Inteligentes: o caso *BuzApp*

Clique no botão para conferir o conteúdo.

ACESSE

Plataforma IoT – Como prototipar nossas ideias (Parte 1)

Clique no botão para conferir o conteúdo.

ACESSE

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta disciplina:

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/3DzptYS>

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/3HD36nx>

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/3Fo5exA>

Diagrama de Atividades

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

Clique no botão para conferir o conteúdo.

ACESSE

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 sexta-feira daquelas, trânsito, chuva e frio.

Você chegou cedo, está cansado e estressado da semana puxada. Ainda por cima, havia dormido pouco porque seu smartphone havia tocado às 3h 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 café e era hora de crescer criando uma versão digital de sua cafeteria *gourmet*.

Portanto, será necessário o desenvolvimento de uma contraparte digital, ou seja, um *e-commerce* para incrementar suas vendas pela *internet*.

Tudo bem, você entra na sala de reunião e vários *stakeholders* estão presentes, inclusive o cliente, para variar, bastante mal-humorado, claro, pois, na última empresa que ele contratou, os analistas não souberam elicitar os requisitos, desprezaram a opinião dele e dos usuários-chave, e construíram

um produto/*software* que não funcionava e as reclamações deles foram muitas: que ele simplesmente retiraram do ar tiveram a imagem de sua empresa arranhada. 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 o cientista da computação responsável por levantar os requisitos dos usuários, e transformá-los em agilidade, depois, apresentar para a aprovação do cliente e sua equipe para evoluirmos para a fase de planejamento do produto/sistema.

Isso foi uma exigência do cliente porque, como já escrevemos, na última vez em que se tentou fazer o produto/*software*, foi uma bomba após outra.

O que se espera de você neste projeto?

- Pesquise na internet por algo relacionado a (*apps, software, internet*, inteligência artificial, IoT, *software* embarcado ou robótica), fica a seu critério. Se já tiver algo em mente, ótimo, utilize-o para o projeto;
- Utilize esses documentos de sua pesquisa como sendo o seu *briefing* do projeto, descrevendo o negócio e as oportunidades percebidas para justificar seu planejamento;
- Leia-os atentamente e anote o que for importante.
- A partir do material de apoio, desenhe o modelo do negócio utilizando o Canvas de Negócios, *Design Thinking* ou BPMN (crie esse documento);
- Agora você deverá criar os cartões de história do usuário para ter os requisitos iniciais do produto/sistema que você escolheu desenvolver;
- Crie um mapa de afinidade das histórias dos usuários, conforme apresentamos no material didático para entender seus relacionamentos;
- Agrupe as histórias e crie um *backlog* de produto já priorizado;

- Extraia dos requisitos das histórias as tarefas e atividades para realizá-las, conforme apresentado no material didático;
- 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 os Artefatos UML

Parabéns, mudamos de fase!

O seu diretor e o cliente adoraram sua modelagem e material de requisitos ágeis e deram sinal verde para que você prosseguisse com a próxima fase do seu projeto de ciência da computação. A da construção dos artefatos básicos da UML e os casos de uso expandidos para provar a factibilidade e funcionalidades.

Nessa fase, eles esperam de vocês as seguintes entregas:

- Diagrama de caso de uso geral;
- Casos de uso expandidos;
- Diagrama de classes;
- Diagramas de máquinas de estado;

- Diagrama de sequência;
- Apresente o documento contendo todos esses artefatos de maneira a que eles contendam a "história" da solução.

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!

Agora você foi autorizado a desenvolver os *wireframes*, mapas conceituais, mapas de aplicativos web e protótipos rápidos.

A diretoria lhe deu sinal verde para que você assuma agora o compromisso com a entrega de *design* via prototipagem rápida.

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 da solução caso haja uma IHC;
- Mapa conceitual da solução;

- Mapa navegacional (mapa do app) caso haja uma IHC;
- Protótipo Navegacional (IHC) ou Protótipo Físico (média fidelidade) para produtos de IoT ou Robótica.

Problema em Foco

Caro(a) aluno(a), vamos lá!

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

A primeira parte é fazer você modelar o negócio e escrever as histórias do usuário a partir dos textos indicados. Explore sua criatividade e imaginação. Com certeza, cada aluno criará o seu, portanto, não poderá haver projetos iguais.

A modelagem do negócio é obtida mediante estudos, *insights* que você terá para a solução entre outros e é representada pelo Canvas de negócio, ou um dos outros métodos citados.

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 estimadas em horas, é recomendado estimar o tamanho das tarefas entre 2-12 horas, para tarefas que requerem mais que 12 horas, quebre essas em várias tarefas menores que 12 horas.

Veja um exemplo abaixo de algumas entregas esperadas:

Utilize o arquivo disponível na seção de **atividade de entrega** como template.

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



Figura 6 – Exemplo de Seleção de Histórias do Usuário por Afinidade

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: isso quer dizer que todos os cartões de história do bibliotecário comporão um tema, assim como 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 1: padrão de planilha para escrever as histórias do usuário.

ID	História do Usuário	Estimativa em Pontos	Prioridade

Isso vai ficar mais ou menos assim:

Tabela 2: História do usuário classificada por estimativa e prioridade

ID	História do Usuário	Estimativa	Prioridade
7	Como um usuário não cadastrado eu quero criar uma conta (<i>login</i> e senha) para acesso.	3	1
1	Como usuário cadastrado eu quero efetuar <i>login</i> .	1	2
1	Como usuário cadastrado eu quero efetuar <i>logout</i> .	1	3

ID	História do Usuário	Estimativa	Prioridade
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[ia] aparecer ou desaparecer na mesma	4	6
8	Como administrado eu quero ver uma lista de contas logadas	8	7

Organize as coisas, veja esse exemplo do que esperamos que você faça para reconhecer as atividades envolvidas em cada cartões de história além dos artefatos acima.

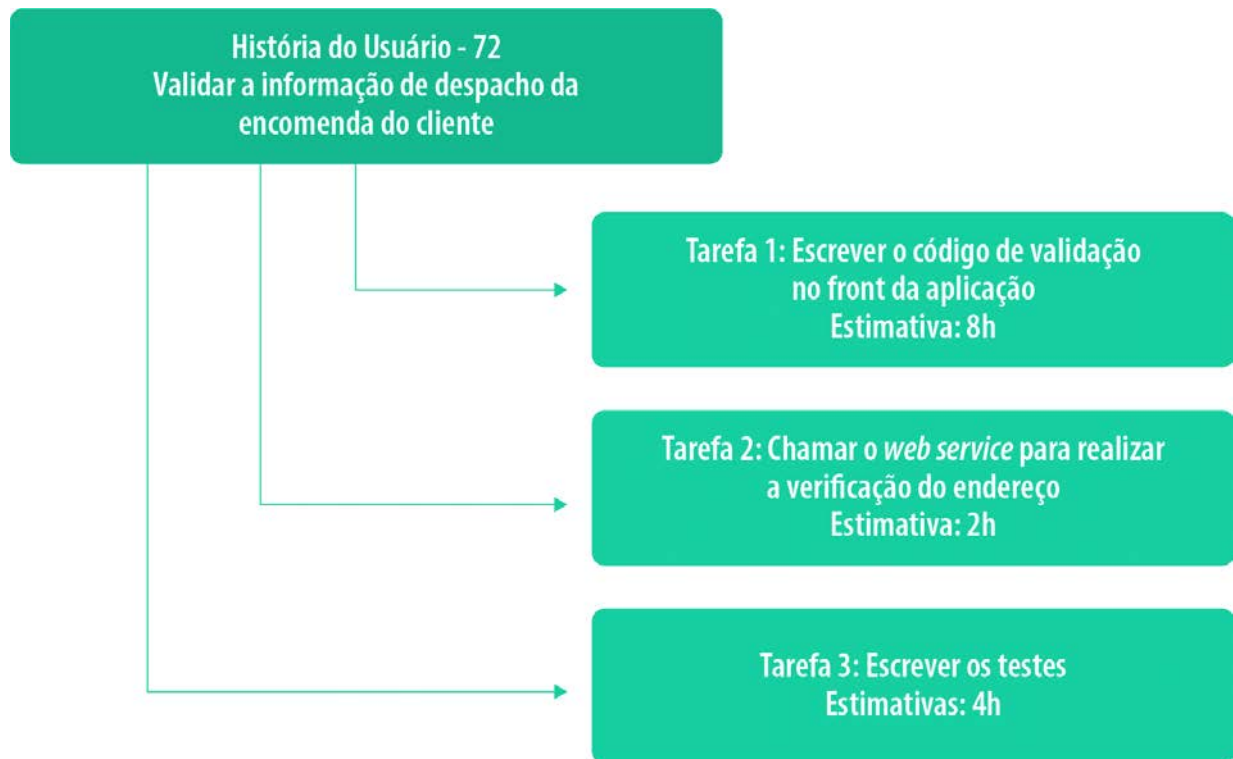


Figura 7 – Exemplo de História do Usuário Quebrada em Tarefas

É apenas um exemplo, há dezenas de histórias de usuários num *APP – Application Program* 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 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, há um exemplo desse com outro aplicativo para servir de inspiração.

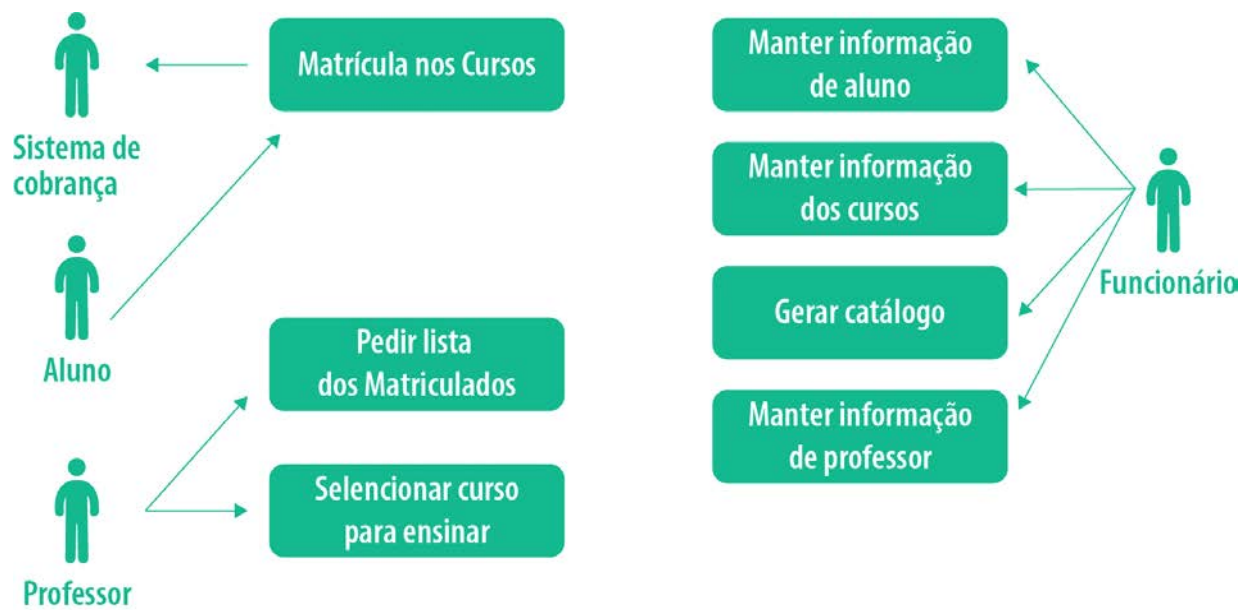


Figura 8 – Exemplo de caso de uso geral

Você, depois do caso de uso geral, deverá expandir os casos de uso e descrevê-los como colocados na parte teórica deste documento. Lembra do exemplo? Veja novamente.

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 Universidade.

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

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 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 a 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;
- **Delta9:** 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* e abaixo dou um exemplo que foi utilizado para modelar uma pizzeria *express*.

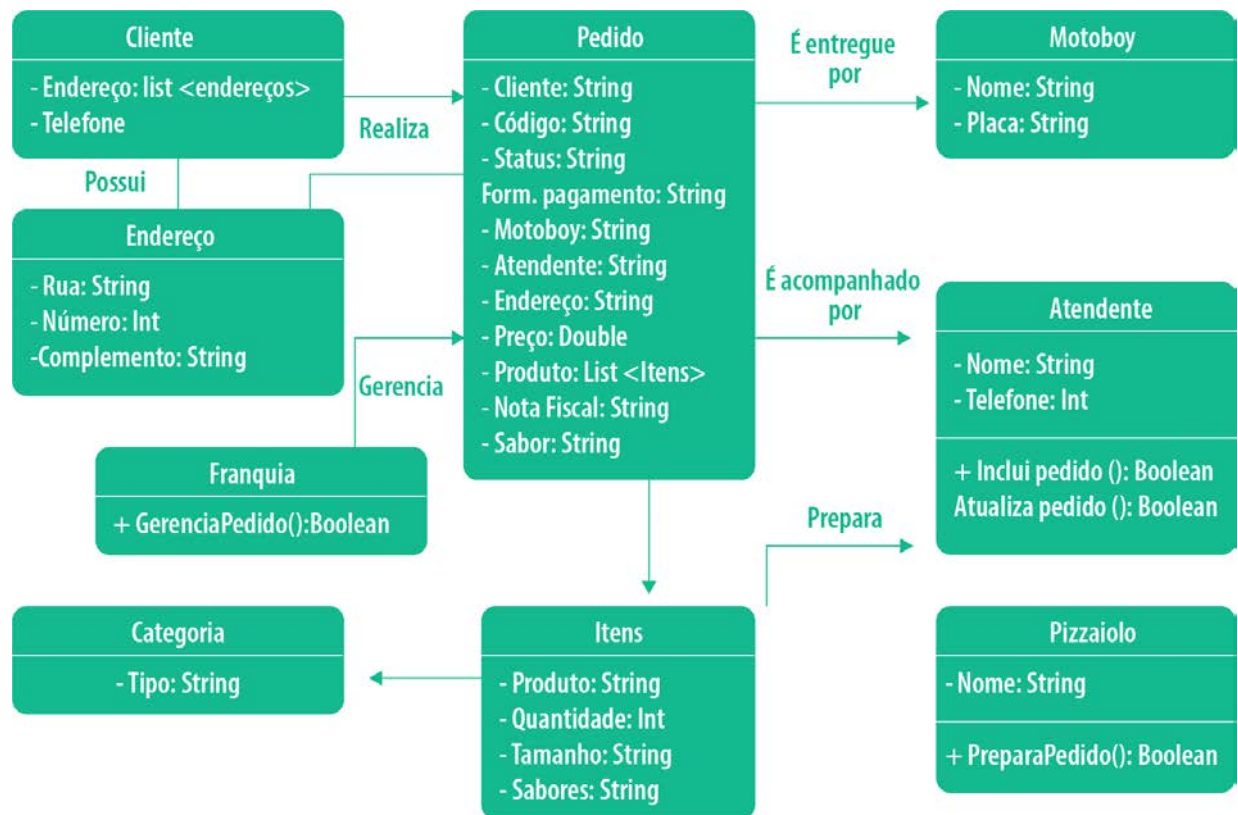


Figura 9 – Modelagem de classes de uma pizzaria express exemplo

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 lembrar como se faz, veja outro exemplo. A partir dele, desenvolva os diagramas para o *app* do *cupcake*.

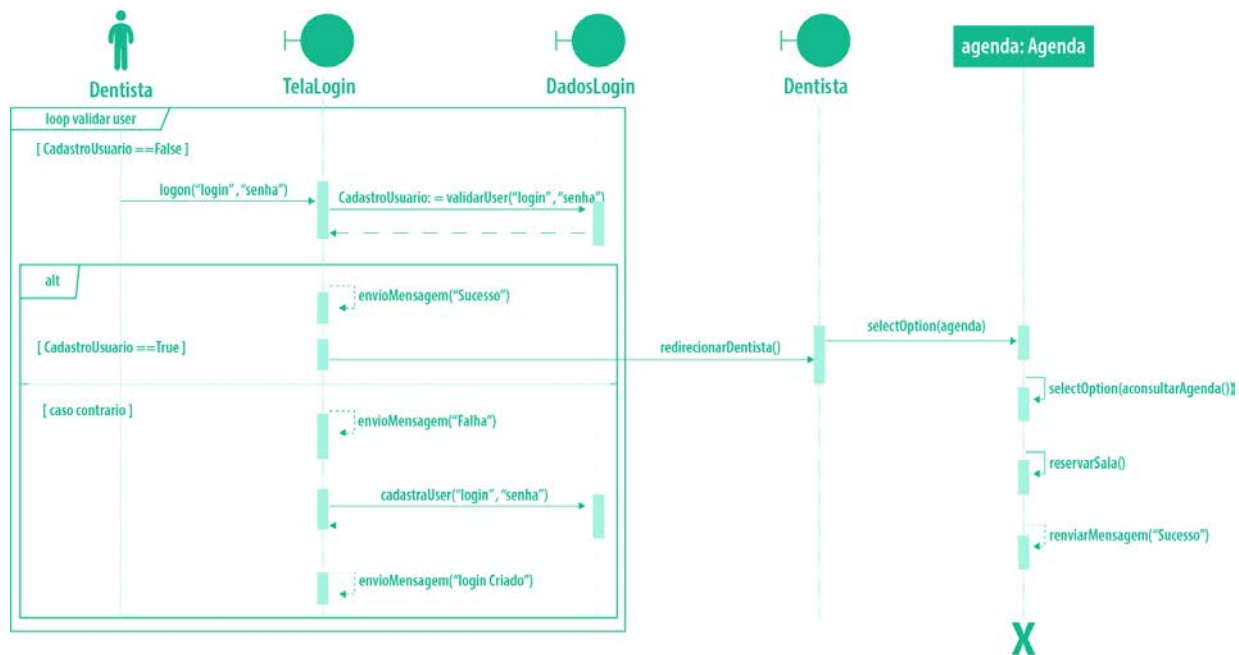


Figura 10

Para o terceiro desafios, 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, que será a "cara" do produto de seu projeto.

A ideia aqui 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*:

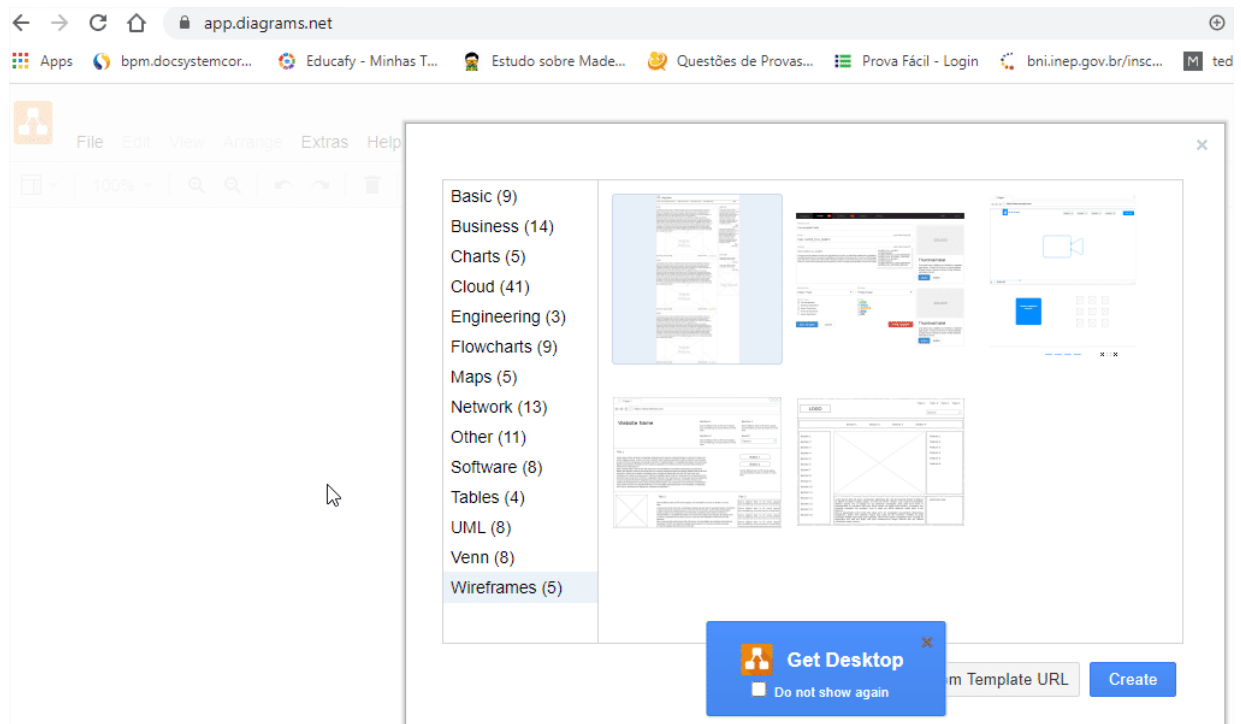


Figura 11 – Tela do *Draw.io*

Fonte: Reprodução

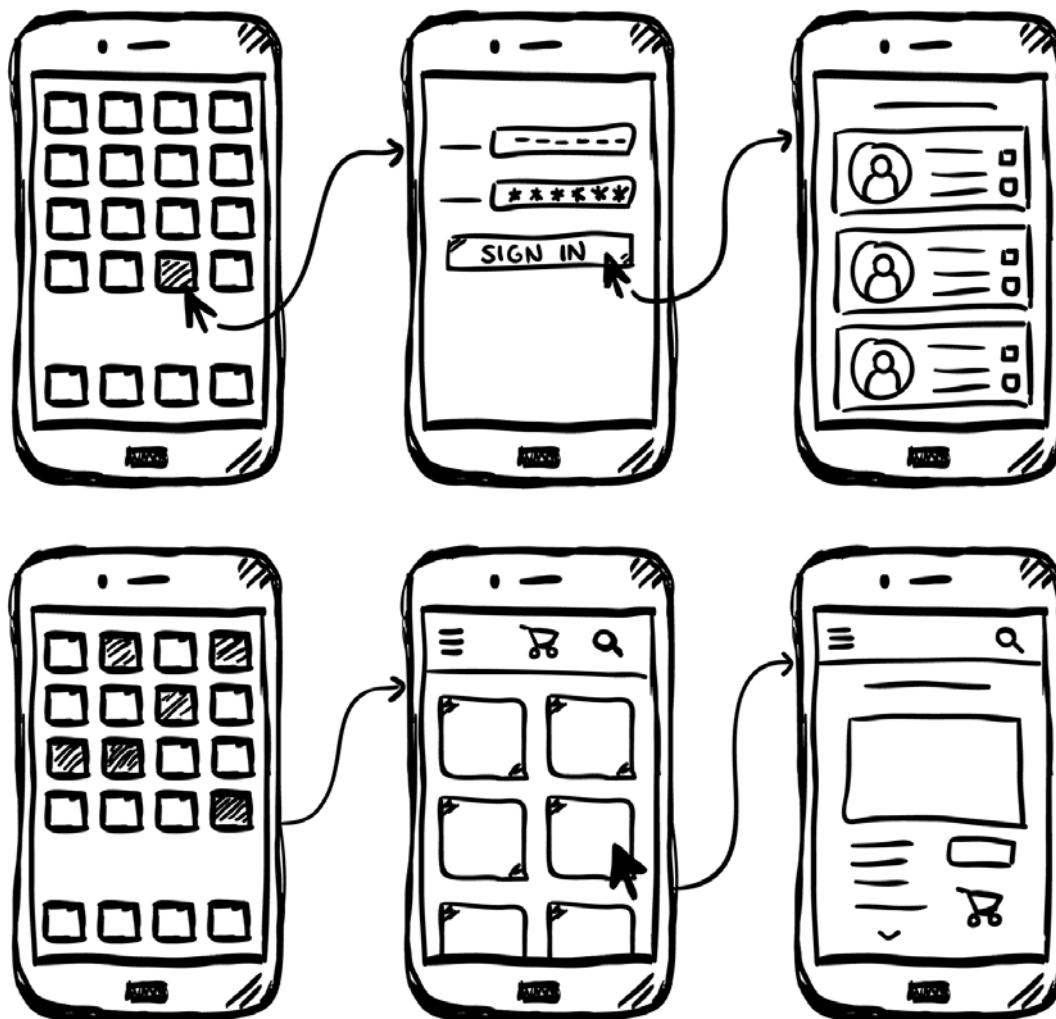


Figura 12 – Exemplo de um *Wireframe*

Fonte: Getty Images

Veja o exemplo abaixo, feito 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

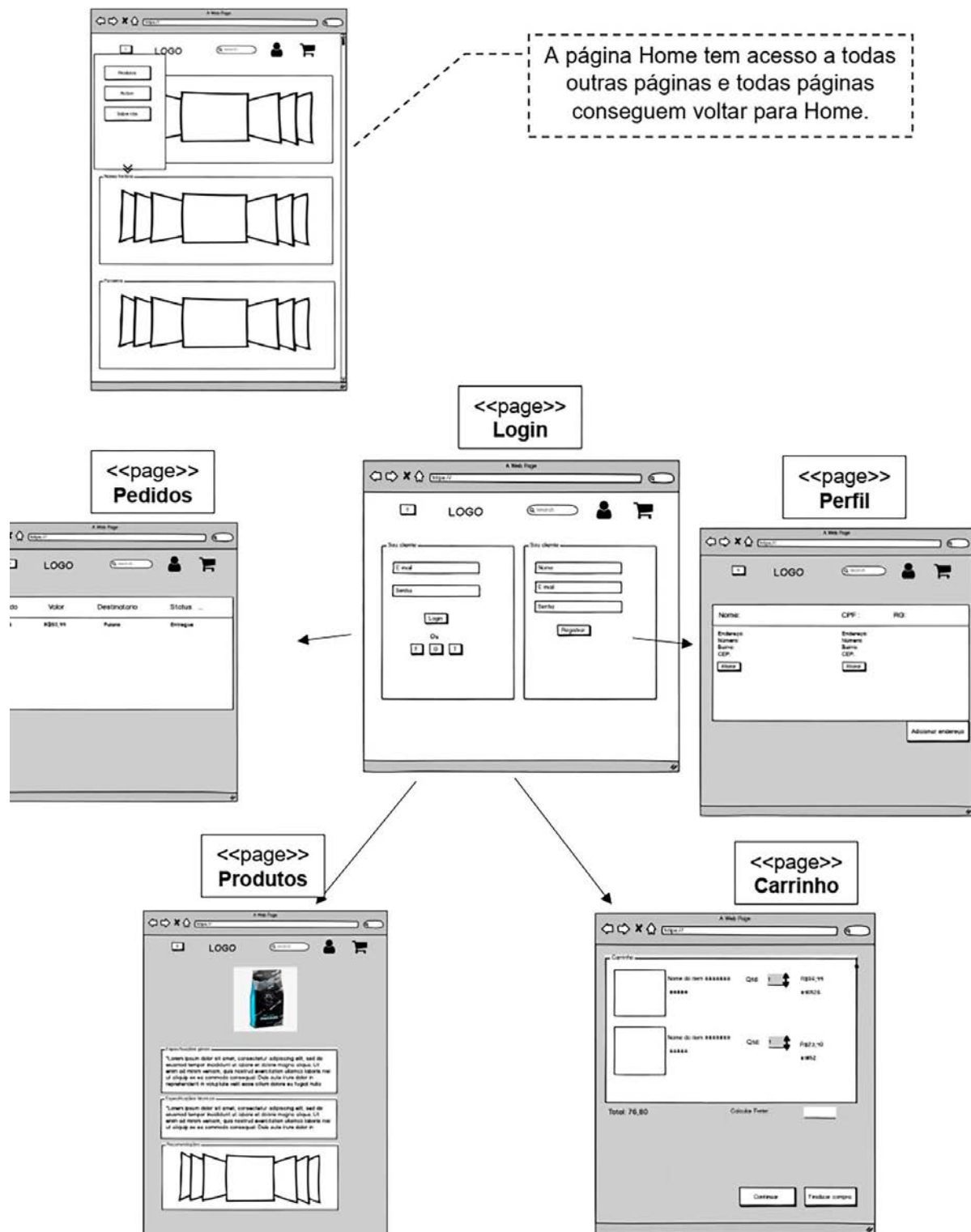


Figura 13 – Exemplo Simples de Mapa Navegacional

Fonte: Acervo do Conteudista

Caso você queira pode fazer num nível um pouco mais profundo e sofisticado, veja o exemplo abaixo. Neste caso, foi feito para cada setor (tela de *login* → *home page*)

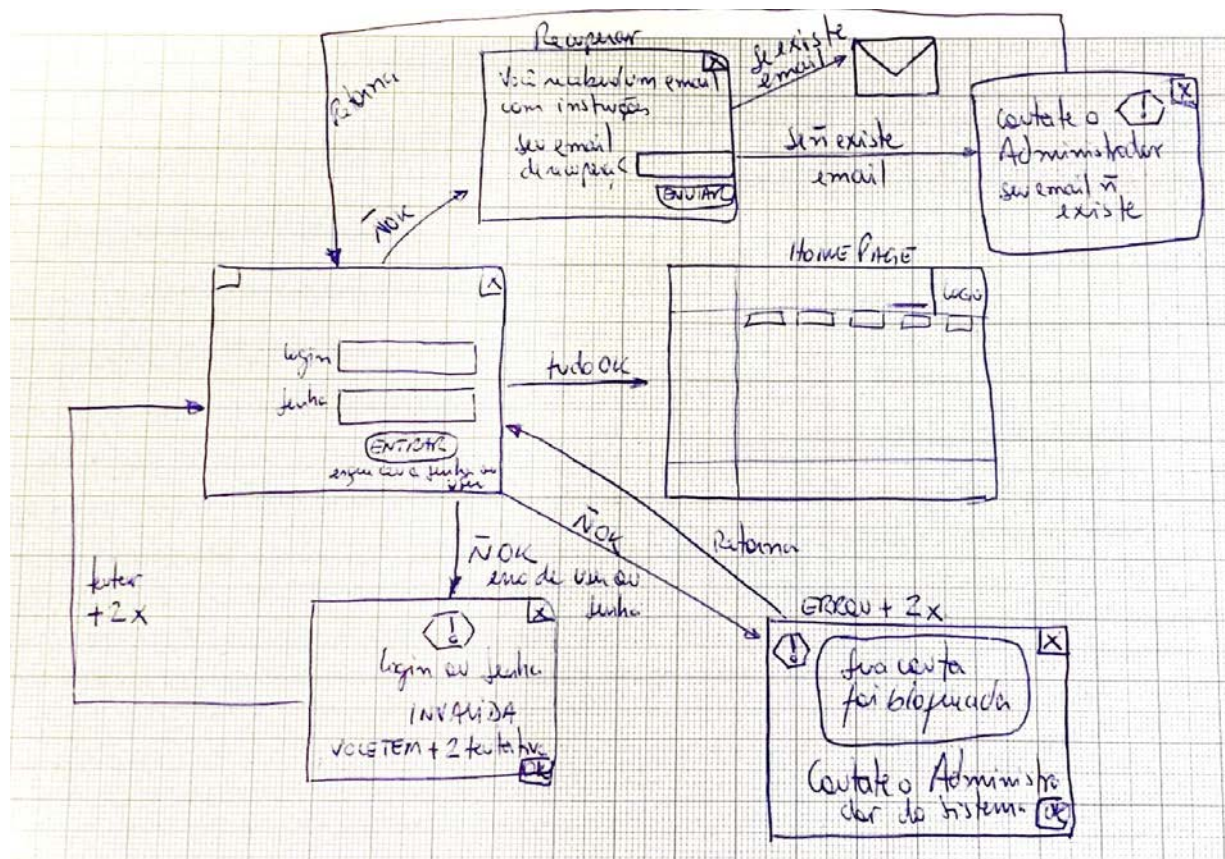


Figura 14 – exemplo de mapa navegacional feito à mão

Fonte: Acervo do Conteudista

Bem, 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, deixamos um exemplo do que você deve fazer baseado no projeto:

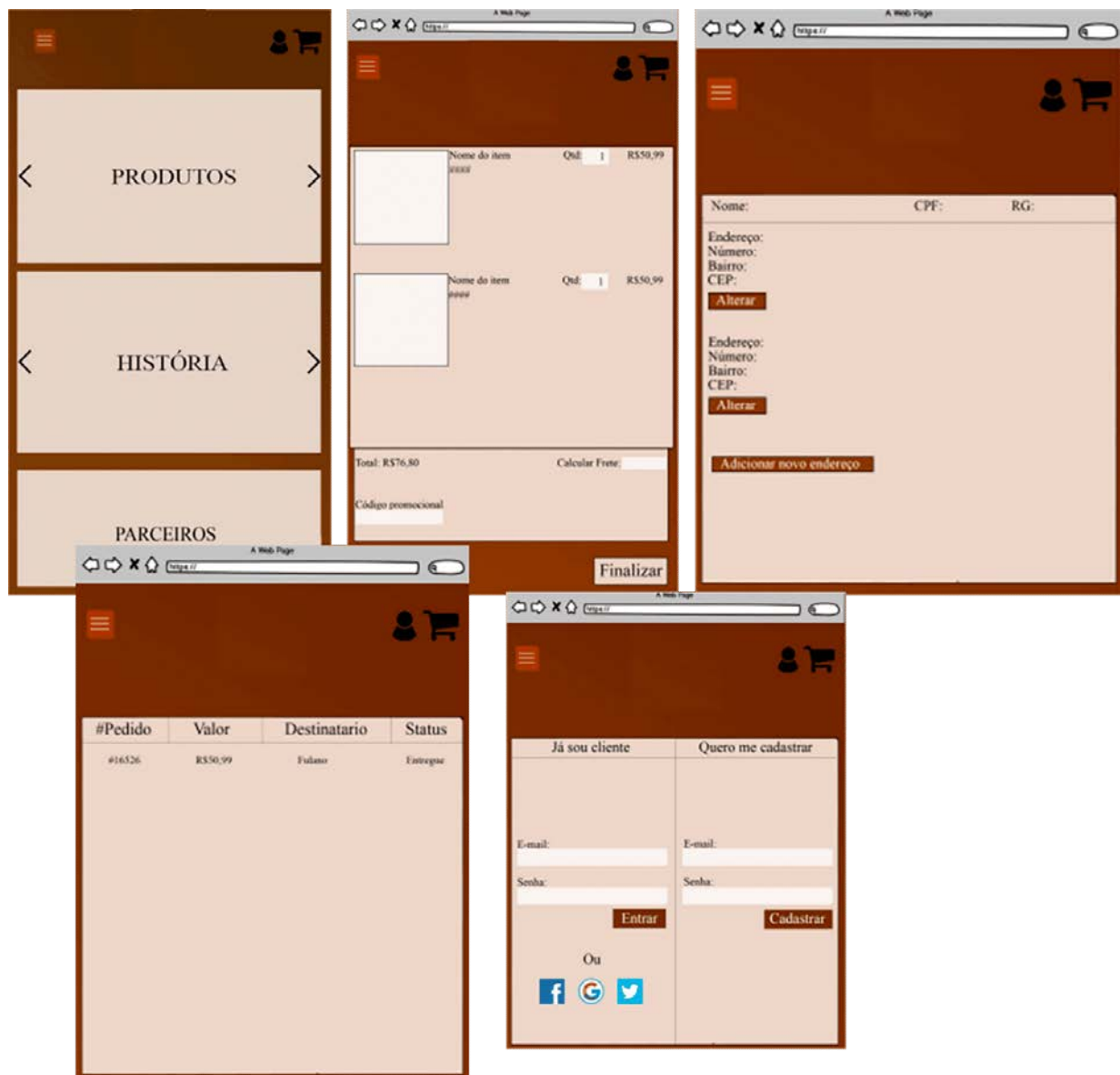


Figura 15 – Exemplo de *Mockup* Feito para Aplicação

Fonte: Acervo do Conteudista

Agora, é só juntar tudo e criar o protótipo para apresentar a navegação das telas em um dos formatos de prototipação rápida que apresentamos.

Desejamos todo o sucesso para você em seu desafio de projeto.

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 arquivo também se encontra no Ambiente Virtual de Aprendizagem.

Feedback

Muito bem, estudante.

O objetivo desta etapa é auxiliar na resolução geral das situações-problema.


Aqui, você pode extrair os melhores caminhos a serem escolhidos em cada cenário descrito anteriormente.

 Atenção, estudante! Aqui, reforçamos o acesso ao conteúdo *online* para que você tenha o *feedback* do professor. Será muito importante para o entendimento dos estudos de caso.

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: 1/10/ 2021.

KRISS, R. **O que é um modelo de negócios?** 2020. Disponível em: <<https://www.nerdwallet.com/article/small-business/what-is-a-business-model>> Acesso em: 1/10/ 2021.

 Muito bem, estudante! Você concluiu o material de estudos! Agora, volte ao Ambiente Virtual de Aprendizagem para realizar a Atividade.