

# Projeto Integrador Transdisciplinar em Sistemas de Informação I



**Conteudista:** Prof. Me. Artur Marques

**Revisão Textual:** Prof. Me. Luciano Vieira Francisco

## ☰ Material Teórico

### 📎 Material Complementar

## DESAFIO

---

### ☰ Situação-Problema 1

### ☰ Situação-Problema 2

### ☰ Situação-Problema 3

### ☰ Problema em Foco

## ATIVIDADE

---

### ☰ Atividade de Entrega

## REFERÊNCIAS

---

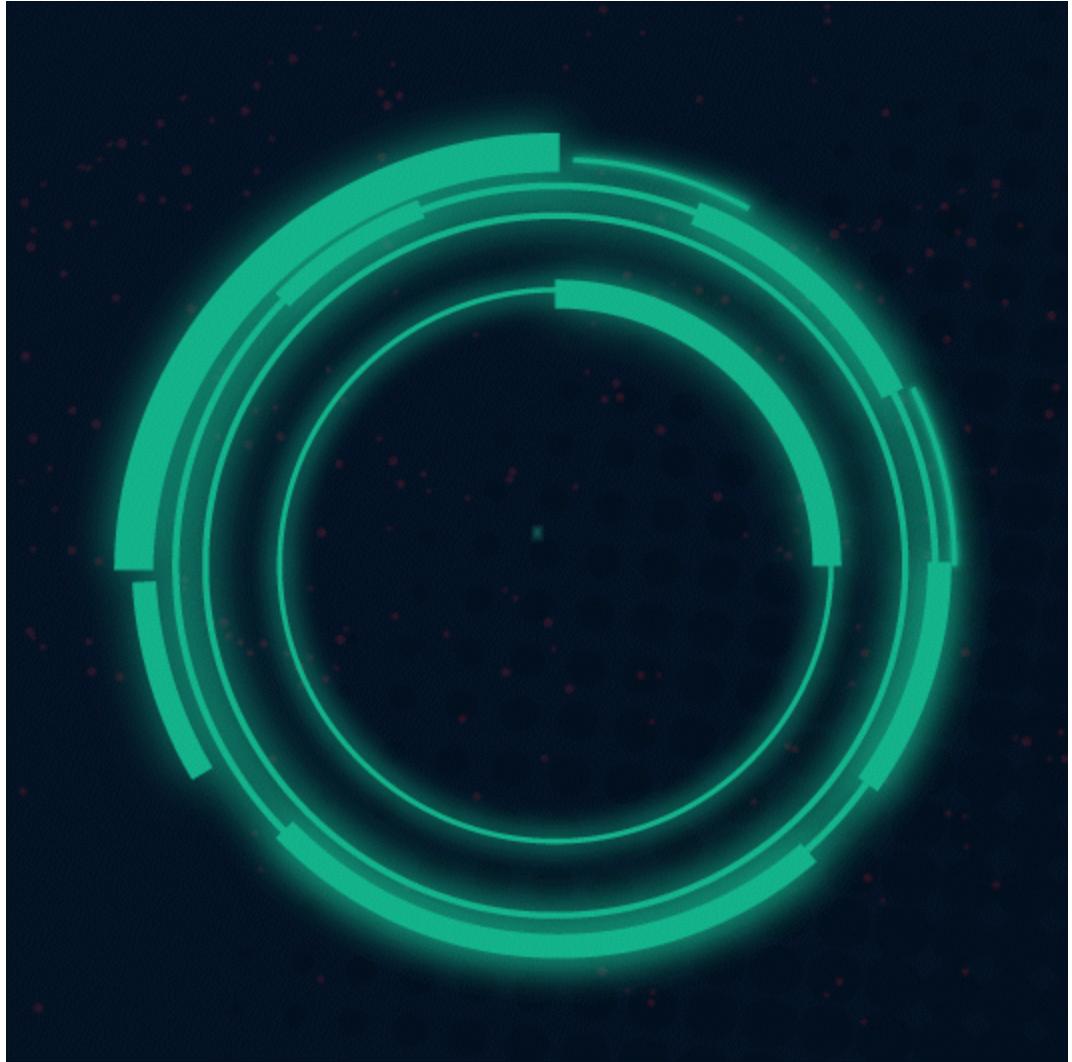
### Referências

## Material Teórico

---

Olá, estudante!

Vamos iniciar a disciplina abordando os conceitos necessários para que você possa realizar as atividades em cada estudo de caso mais à frente.



---

Unidade 1 - Projeto Integrador Transdisciplinar Em Sistemas de Informação I





0:00 / 23:17



1x



## Introdução

A ideia do desenvolvimento deste conteúdo didático em uma disciplina que preza pelo desenvolvimento prático de um projeto é fazer com que você possa se recordar do que já aprendeu de forma a lhe manter focado(a) no desafio e ter praticamente tudo do que precisa para desenvolvê-lo em um lugar só, incluindo referências e *sites* relevantes de ferramentas.

Você também encontrará leituras e material complementar que serão essenciais no aprofundamento e desenvolvimento desse desafio. Muitos discentes comentam que os Recursos Humanos (RH) das empresas pedem experiência aos candidatos.

Com certeza é bastante estranho pedir por algo dessa natureza a quem tenta entrar ou desenvolver uma carreira. Porém, com estes desafios você terá a oportunidade de mostrar que desenvolveu o seu aprendizado e demonstrar que está pronto(a) para o que se apresentar, concorrendo de igual para igual com qualquer outro candidato.

Vamos aproveitar para recordar?

## Modelos de Negócios

Você deve ter percebido que o modelo de negócio é fundamental porque a partir dele é que montamos o plano de negócios. 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; descrever o que Michael Lewis

definiu como modelo antes do “estouro da bolha da *internet*”, ou mais recentemente com Joan Magretta com sua teoria de que os modelos de negócios são histórias que tentam explicar como as empresas funcionam, focada mais nas premissas do que em apenas fazer dinheiro, assim como Drucker. Por sua vez, Lewis é mais direto, de modo que o negócio tem que fazer dinheiro. Nos tempos mais recentes temos Alex Osterwalder com o popular modelo **canva de negócios**, que tem se espalhado pelas *startups* ao redor do mundo.

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, sobrevivência ou ao fracasso.

Uma definição mais direta pode ser encontrada em Kriss (2020, p. 1):

“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 a seguir não existe um modelo único que possa ser aplicado a todos os negócios.

”

Depois de tantas definições, focaremos em um exemplo de modelo de negócio digital para que as coisas fiquem claras para você. Um exemplo popular de modelo de negócios é o de assinatura – no qual as empresas cobram uma taxa de assinatura (mensal, anual etc.) para que os clientes acessem um serviço. Usaremos o serviço **Netflix** como exemplo, dividindo-o com base em quatro perguntas simples:

Qual tipo de produto ou serviço uma empresa venderá? *Netflix* vende um serviço de *streaming online*.

Como pretende comercializar esse produto ou serviço? *Netflix* usa uma estratégia de *marketing multicanal* e comercializa o seu serviço por meio de mídia social, *marketing* por *e-mail*, publicidade e até mesmo *marketing boca a boca*.

Qual tipo de despesas enfrentará? Como uma empresa *Fortune 500*, as despesas da *Netflix* são extensas, mas talvez e notavelmente as 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.

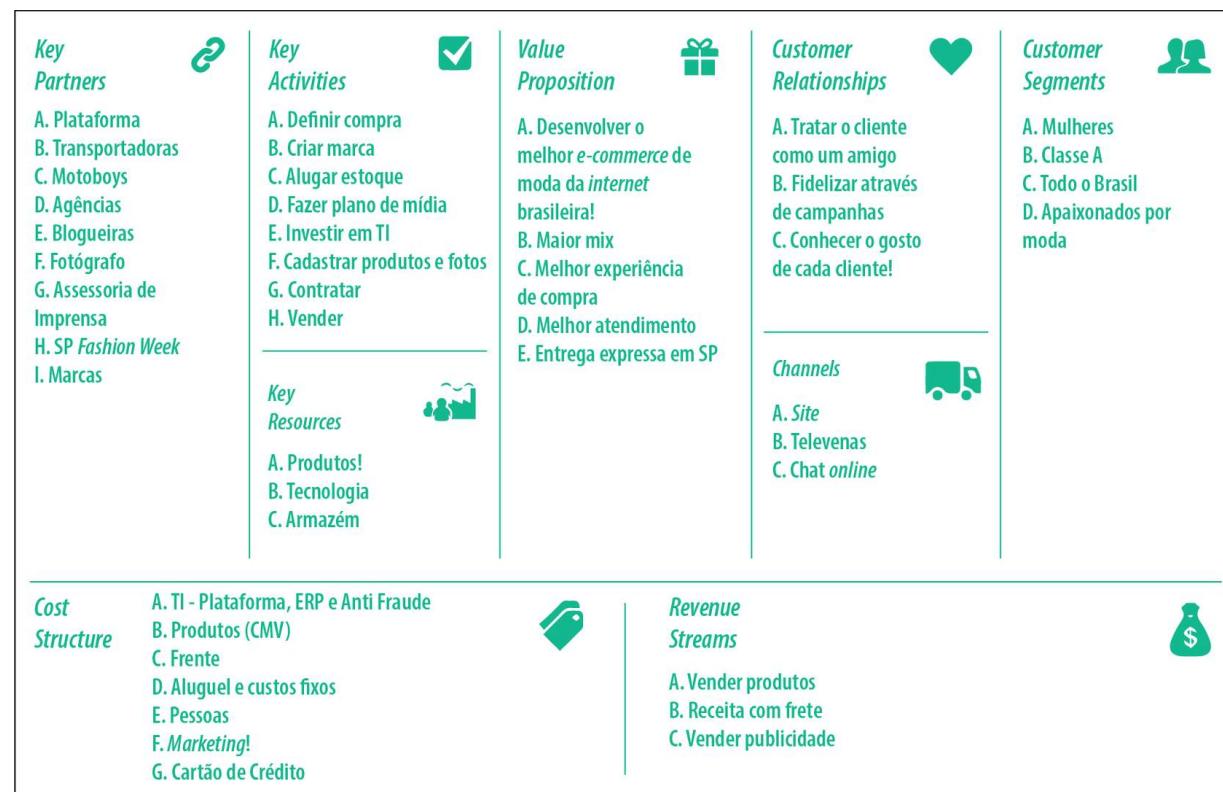
Como espera obter lucro? Embora a *Netflix* seja uma empresa tão grande – e tenha algumas maneiras diferentes de ganhar dinheiro –, ela espera obter lucro com as vendas de assinaturas.

Em última análise, então, é imperativo entender como a empresa ganhará dinheiro – o suficiente para permanecer lucrativa após a inicialização e os custos adicionais considerados na equação –, sendo este o seu modelo de negócios (KRISS, 2020, p. 3).

Então, encararemos o modelo de negócios *canva*, que tem a forma de um gráfico visual com diversos elementos que descrevem a proposta de valor de uma organização, a sua infraestrutura, o mercado e as finanças. O objetivo é auxiliar as empresas a alinharem as suas atividades por meio da ilustração de possíveis *trade-offs*, de modo que neste caso de negócios podemos entender esse termo como compensação, perde-ganha, pois sempre em uma troca comercial há esse balanceamento, de modo que uma parte oferece algo em troca daquilo que precisa a outra parte, de

tal forma que a relação seja a mais justa possível. O que está em jogo é, em outras palavras, dinheiro por um bem ou dinheiro por serviços, isto é, de forma tradicional. Claro que tais relações não se encerram nessas duas possibilidades.

O intuito do *canva* é visualizar todos os blocos de construção quando se deseja iniciar um negócio, incluindo clientes, rota para o mercado, proposição de valor e finanças.



**Figura 1 – Exemplo de Canva**

Ademais, os blocos do *canva* são os seguintes:

- Segmentos de clientes: quem são os clientes? O que acham? Ver? Sentir? Fazer?
- Proposições de valor: o que é atraente sobre a proposição? Por que os clientes compram e usam?

- Canais: como estas propostas são promovidas, vendidas e entregues? Por quê? Está funcionando?
- Relacionamento com o cliente: como você interage com o cliente durante a 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 a 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 estão vinculados à receita?

Você pode optar e fazer uma modelagem utilizando *Design Thinking* (DT) ao invés do *canva* de negócios.

É centrado no ser humano, significando que usa evidências de como os consumidores, ou seja, as pessoas realmente se envolvem com um produto ou serviço, em vez de como outra pessoa ou uma organização pensa que se envolverão com ele.

Entenderemos o contexto:

No início dos anos 1970, fabricar produtos mais baratos era um dos grandes objetivos das empresas. Se você projetou um produto com um preço de custo mais baixo é 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 os seus recursos, design e usabilidade. Nos últimos tempos, o foco das empresas se tornou fazer produtos melhores de pessoas

para pessoas, significando improvisar produtos com dados reais para fornecer aos usuários uma solução de que realmente precisam (KHAN, 2015, p. 1).

Por que isso é importante?

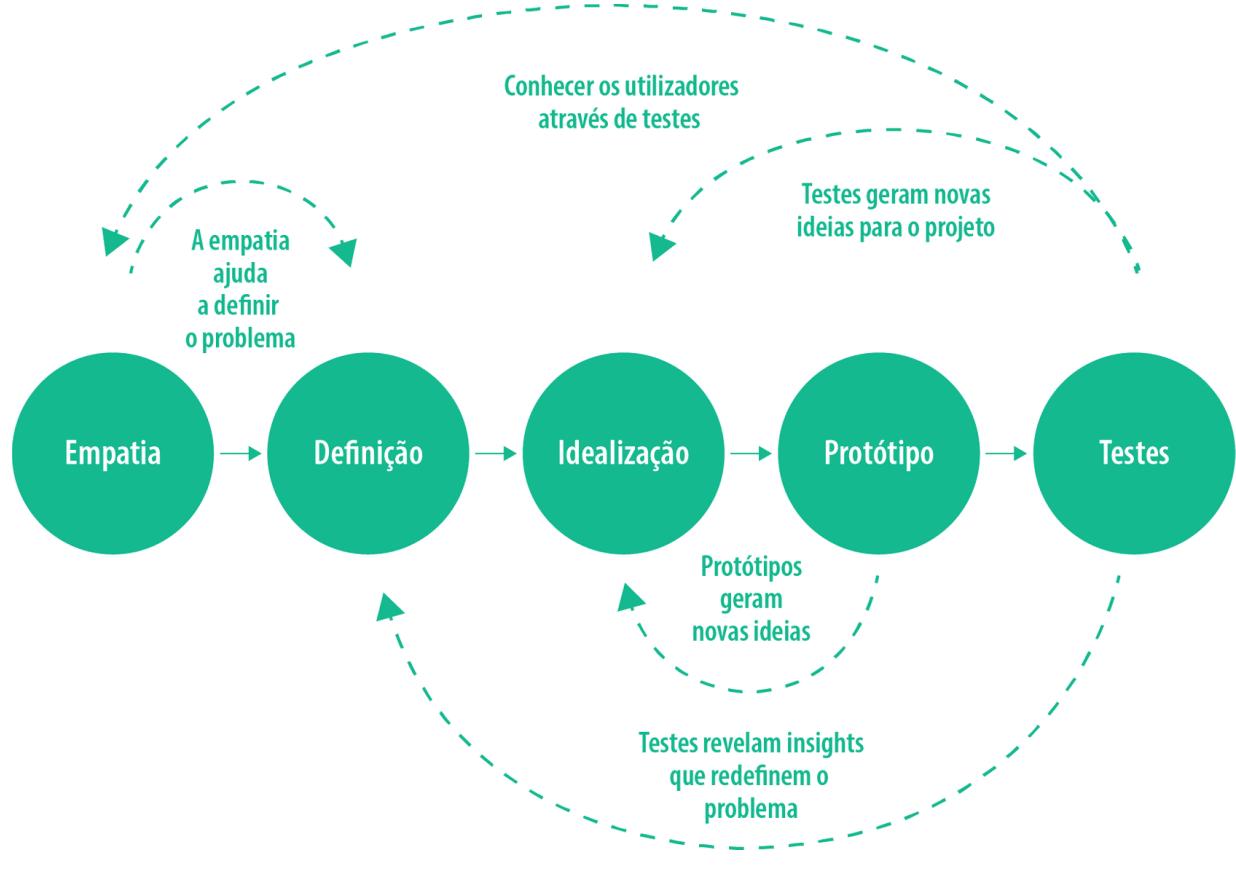
Bem, porque praticamente tudo o que é ofertado ou vendido na web passou, durante o seu processo de criação, por essa metodologia; fora isso, o seu empregador atual ou futuro apostava que você sabe fazer isso. Então, parece importante, afinal, 10 entre 10 universidades, cursos livres e gurus em geral tentam ensinar pessoas a adotarem DT em suas vidas profissionais e pessoais. É a chave que favorece à movimentação rápida para colocar protótipos para teste, em vez de pesquisas ou planejamentos intermináveis.

Em contraste à solução tradicional de problemas, que é um processo linear de identificação de um problema e, em seguida, o *brainstorming* de soluções, DT só funciona se for iterativo. É menos um meio de se chegar a uma solução única e mais uma maneira de desenvolver continuamente o seu pensamento e responder às necessidades do consumidor. O melhor é que DT permite que as organizações criem valor duradouro para os consumidores.

Pense o seguinte: os seus concorrentes, que você nem sabe quais são na web, atuam em seus negócios digitais e neste exato momento usam DT para fazer com que o seu cliente vá para eles, de modo que os motivos podem ser porque o DT:

- Visa resolver uma necessidade humana concreta;
- Resolve problemas ambíguos ou difíceis de se 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 estas fases, estágios ou modos nem sempre são sequenciais, não precisando seguir nenhuma ordem específica e muitas vezes podem ocorrer em paralelo e se repetir de forma iterativa.



**Figura 2 – Processo do DT**

## Requisitos Ágeis

Primeiramente, algumas constatações.

Muitas vezes, no início da carreira vemos projetos com dificuldades porque os seus requisitos são mal elaborados, comumente sem sentido. Olhamos para os “requisitos” e não são nada mais do que um quadro com um monte de notas e *Post-It*. Com muitas dúvidas ficam todas as pessoas envolvidas – clientes, usuários, desenvolvedores, analistas de negócio, arquitetos de solução e inclusive os testadores –, sem um bom entendimento do sistema como um todo e das várias *personas* que usam o sistema. Claro que falta a todas essas pessoas a apropriação do conhecimento e dos processos do negócio.

Então, respire fundo e se prepare para pensar e analisar em profundidade, dado que analistas de sistema criado no raso jamais enfrentaram grandes “tempestades e ondas avassaladoras”, acredite. Mas aqui vão dicas preciosas para você desenvolver o seu trabalho com menos incertezas.

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. Neste caso, você possuirá múltiplos “chapéus”, ora sendo 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*.

Sem nenhuma das seções mencionadas os requisitos começam a perder valor e sentido. Muitas vezes quando somos chamados para dar consultoria vemos times ágeis e especificações de sistema como um “punhado de gente bem-intencionada que não consegue escrever ou muitas vezes ler os livros que ganharam”. Daí fica realmente difícil construir algo se nem entendemos o português que está escrito, não é mesmo? Cada seção de análise traz para a mesa e muitas vezes são julgadas como “perda de tempo”, porém, quando codificamos fazem toda a diferença. Olha só:

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

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

Ou seja:

Como **gerente de contas a receber**, **quero** ter uma relação de devedores por safra (meses vencidos od, 6od, 9od, 18od, 36od), **para que possa** encaminhar às empresas de cobrança por seguimento e especialidade.

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, reflita como um profissional, mas você pensará: "mas sou aluno(a)", sim, todos somos alunos, incluindo o confeudista deste material, mas atitude é tudo e é isto 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: deve incluir uma imagem das telas envolvidas (você fará isso na IHC e no protótipo das telas, pois 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 complexos e é difícil explicar os detalhes na próxima seção – aqui visão é tudo.

---

## Explore:

**Segue uma lista de alguns sites para você usar na construção de sua aplicação, os quais ajudarão a explicar o sistema através de fluxos e apoiarão no desenvolvimento. Suas versões demo permitirão que você os**

**utilize pelo tempo desse desafio tranquilamente, portanto, não “durma no ponto”. São todos amigáveis e não precisam de curso para usar:**

[Gliffy](#)

[Creately](#)

[Lucidchart](#)

[Canva](#)

[Diagrams](#)

Detalhamentos dos requisitos: são os detalhes do recurso. Documente todas as telas e todos os campos, rótulos, validações, mensagens e ações. 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 precisará de vários dias para eliciar e – por que não? –, prever os requisitos de alto nível e entender o escopo e, neste caso, o que acha que o sistema deverá fazer.

O seu objetivo será 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 o seu modelo de requisitos inicial você precisará de um modelo de:

- **Uso:** permite que você explore como os usuários trabalharão com o seu sistema. Pode ser uma coleção de casos de uso essenciais ou uma coleção de histórias de usuário;

- **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, mas cobrir informações suficientes para que você se sinta confortável com os conceitos de domínio primário – leia mais no Material Complementar;
- **Interface do usuário:** para projetos intensivos de *interface* de usuário, de modo que você deverá considerar o desenvolvimento de alguns esboços de tela ou até mesmo um protótipo de interface de usuário – conforme veremos no tópico *Wireframe*.

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

A ideia é que você busque artefatos de requisitos que sejam bons o suficiente para lhe dar esse entendimento e nada mais, portanto, pense simples, porém, de forma direta. Veja este exemplo de um caso de uso geral:

Nome do caso de uso geral (macro): inscrever-se em uma disciplina.

Curso básico de ação (trata-se de um caso de uso normal):

- 1 O aluno digita o 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 se 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 da disciplina quando o contas a receber der baixa no crédito.

Esta descrição 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 de o que se pede.

Porém, podemos detalhar isso em um caso de uso mais aprofundado e que chamaremos de caso de uso expandido:

**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.

**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 é fundamental quando estamos elicitando requisitos e/ou desenvolvendo casos de uso.

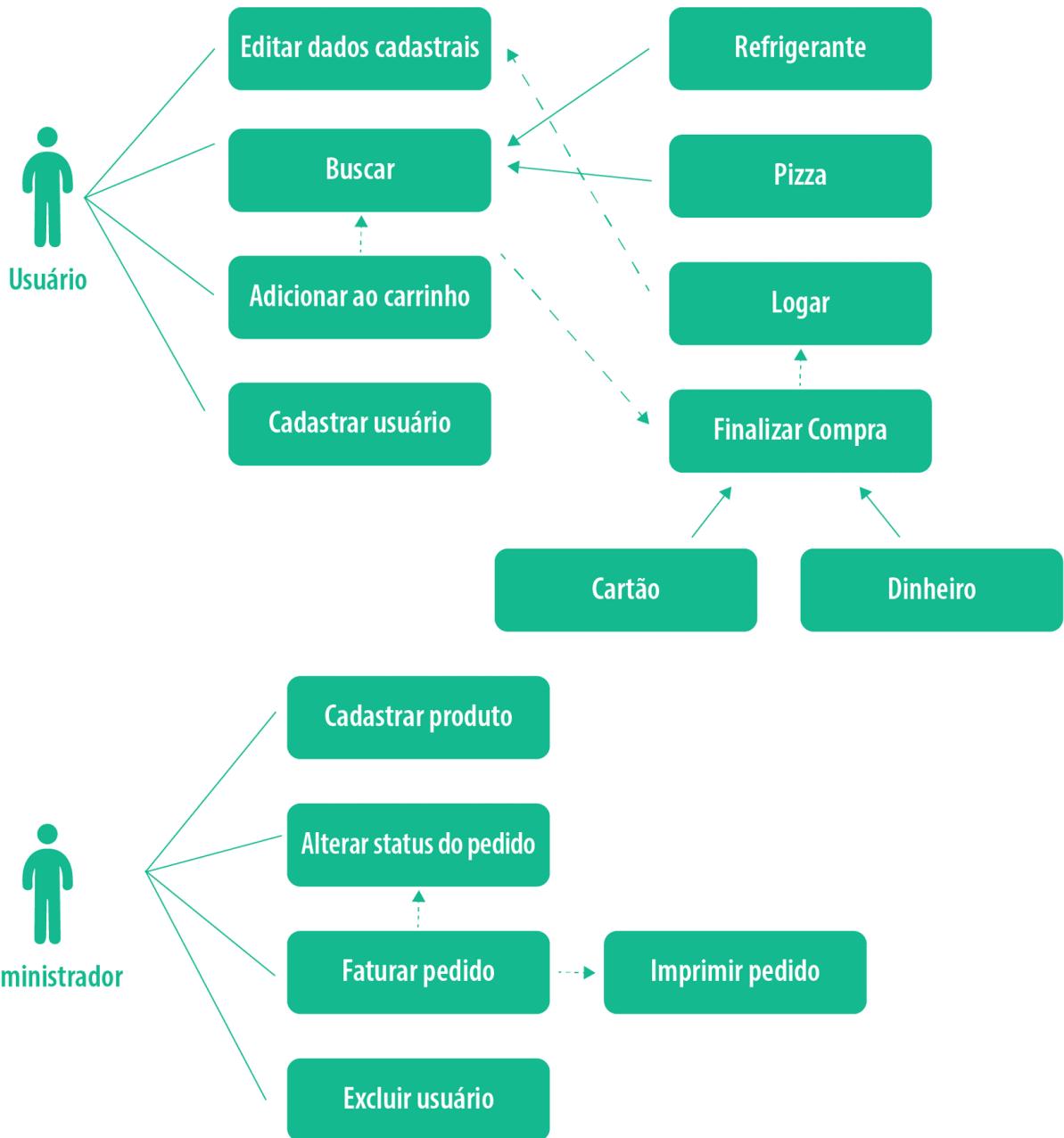
Temos aqui a descrição do mesmo caso de uso totalmente documentado. Este é um exemplo maravilhoso de um caso de uso bem construído, mas apresenta mais detalhes do que você possivelmente precisa no primeiro momento. Se você realmente precisa desse nível de detalhe e, na prática, raramente o faz, pode 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 etapas posteriores, declare analiticamente como no segundo exemplo, afinal, fica difícil um desenvolvedor saber o que ele tem que fazer sem as regras de negócios ou as telas da interface do usuário (camada de apresentação). Por isso vemos tanta coisa sem sentido por aí, porque deixaram a critério do desenvolvedor algo que não é a sua missão. *Interface* do usuário, *front end* e *back end* são grandes atividades que devem trabalhar seguindo padrões e metodologias porque uma depende da outra.

Caso você esteja no nível mais inicial, poderá recorrer ao bom, velho e, sem dúvida alguma, útil diagrama de caso de uso:



**Figura 3 – Diagrama de caso de uso macro de uma pizzaria**

Inicialmente, crie todos os diagramas de caso de uso, como o da Figura 3, 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 converter em artefatos – a seguir há alguns listados –, mas dê muita 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. Neste caso use reuniões remotas síncronas ou documentos como questionários;
- As partes interessadas do projeto não sabem o que querem – mais comum do que parece; neste caso seja um(a) guia ou “terapeuta”;
- As partes interessadas do projeto mudam de ideia – negocie, pois fazem isto por causa do item anterior, de modo que vão descobrindo conforme pensam no próprio negócio, coisa que a rotina não viabiliza;
- 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 aterrem-se à área de domínio dos quais, ou seja, negócios;
- As partes interessadas do projeto são incapazes de ver além da situação atual – em tecnologia chamamos isso de paralisia de paradigma, crença limitante;
- As partes interessadas do projeto têm medo de ser fixadas – é uma questão cultural que 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, pois pessoas se encantam com histórias, de modo que a técnica de *storytelling* veio para ficar;
- Os desenvolvedores não entendem o domínio do problema – é importante os desenvolvedores estudarem o negócio em profundidade, pois não existe mais desenvolvedor “trancado em uma sala”, de modo que precisam ter contato com a

- operação para amadurecerem e melhorarem os seus entendimentos sobre o negócio e 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 as travas;
  - As partes interessadas do projeto exigem formalidade significativa em relação aos requisitos – evite esta armadilha, dado que as partes interessadas muitas vezes não conhecem toda a solução e querem garantias impossíveis, de modo que pedir por detalhamento é um tipo de transferência de responsabilidade. Em um projeto ágil os requisitos e a própria arquitetura vão se revelando aos poucos;
  - Os desenvolvedores não entendem os requisitos – convide-os a passar um tempo na operação, de preferência fazendo o papel do usuário, 1 semana é o suficiente para entenderem tudo.

## Diagrama de Classes

Recordaremos os diagramas de classes, os quais mostram as classes do sistema, os seus inter-relacionamentos, incluindo herança, agregação, associação e as operações e os atributos das classes. São usados para 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 se descreve por diagramas de classe ou pacote, comumente para mostrar camadas globais. 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 o seu objetivo.

Outros padrões de arquitetura como MVC – *Model-View-Controller* – podem ser escolhidos como 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 os significados dos componentes da arquitetura para que os membros da equipe possam escrever códigos que se encaixem no lugar certo na

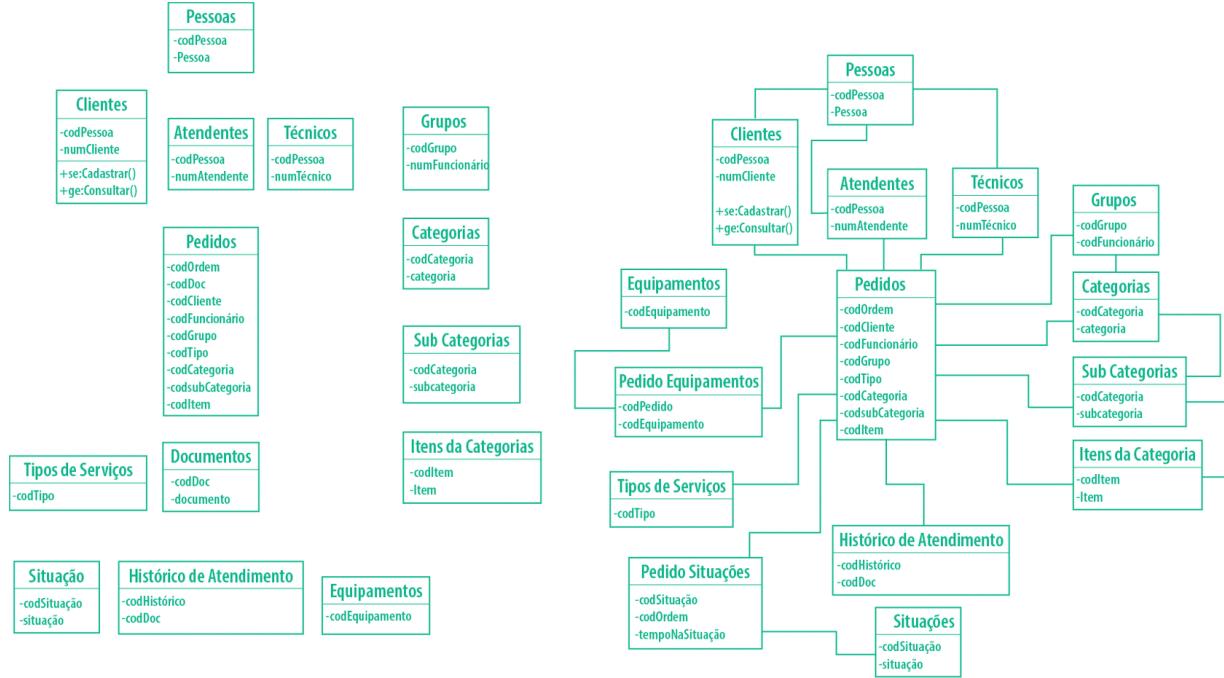
arquitetura e de forma consistente. "Dependências" são frequentemente expressas neste diagrama entre pacotes para evitar acoplamentos indesejados ou dependências circulares.

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

**Modelos de domínio como diagramas de classe ou Entidade-Relacionamento (ER)/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, tais como classes, dados, métodos e outras convenções.

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 a 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 importante (JOBA, 2016, p. 5-6).

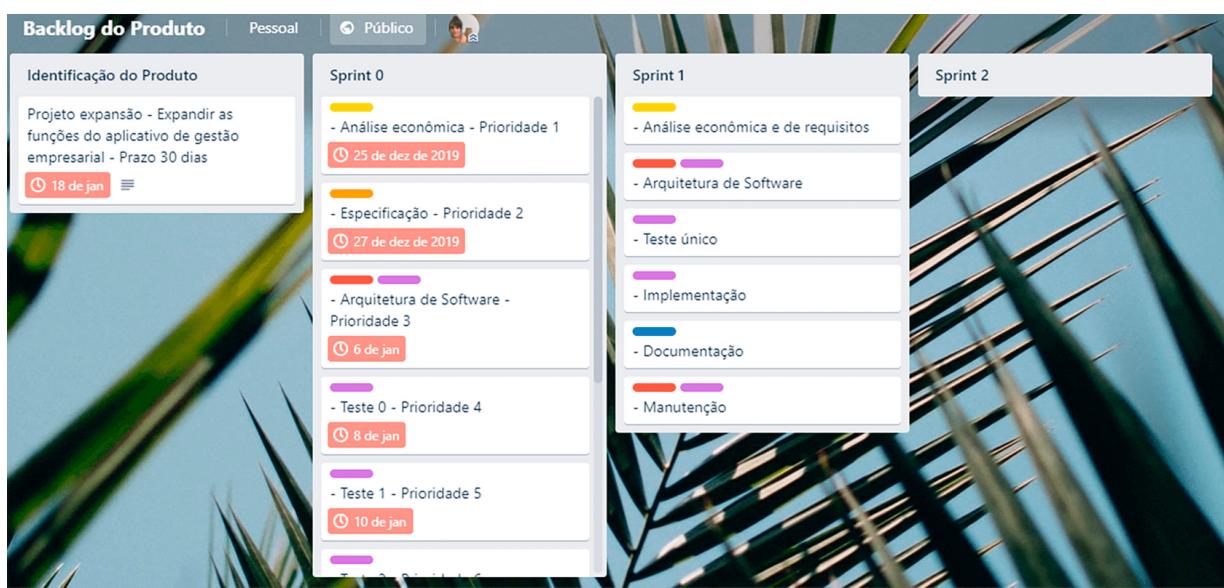


**Figura 4 – Exemplo de classes levantadas e ao lado diagrama de classes completo**

## Backlog de Produto

Para recordarmos, *backlog* de produto é uma lista priorizada de entregas e isto inclui novos recursos que devem ser implementados como partes de um projeto ou no 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. Garante que a equipe trabalhe nos recursos mais importantes e valiosos, corrigindo os bugs mais importantes ou fazendo outro trabalho crítico para o desenvolvimento do produto.

*Backlog*, portanto, é extremamente útil em situações em que você não consegue fazer tudo o que é solicitado, ou em contextos em que mesmo uma pequena quantidade de planejamento ajudará significativamente. Alguns o consideram como uma lista de tarefas pendentes e o definem exatamente dessa forma, como uma lista de coisas que você deve fazer para entregar o seu produto de *software* ao mercado. Na verdade, não é necessariamente uma lista de tarefas pendentes, de modo que pense nisto 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, por exemplo o *Trello*, porque é simples, intuitivo, online, colaborativo e grátis.

Clique no botão para acessar o *Trello*.

[ACESSE](#)

## 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 os 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 determinados itens do *sprint backlog* até o final do *sprint*, poderá escolher adicionar esses trabalhos inacabados ao próximo *sprint backlog*, caso ainda sejam 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*, *product owner* e o *time* de desenvolvimento – compartilhará a propriedade do *sprint backlog* – isto ocorre porque todos os membros da equipe trarão percepções e conhecimentos exclusivos para o projeto no início de cada *sprint*.

*Sprint backlogs* são geralmente planilhas embutidas, mas também podem ser desenvolvidos e mantidos em ferramentas de *software* projetadas para o gerenciamento ágil de projetos. Como essas listas incluem apenas trabalhos que podem ser concluídos em um curto espaço de tempo – que chamamos de *sprint* –, durando de 2 a 4 semanas, os *backlogs* de *sprint* costumam ser 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, o qual 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.

Ademais, o diagrama de atividades é usado por desenvolvedores para entender o fluxo de programas em alto nível, permitindo que descubram restrições e condições que causam eventos

específicos. Geralmente usamos o diagrama e a documentação textual para tornar a descrição do nosso sistema a mais clara possível.

## TDD e Testes

Testes de aceitação do usuário: devem incluir todos os cenários descritos nas histórias de usuário, não devendo ser detalhados (não precisam mencionar telas específicas ou uma lista completa de ações para executar as etapas). Devem ter:

**Dada** essa condição 1 e condição 2...

**Quando** eu faço a etapa 1 e a etapa 2...

**Então**, resultado desejado 1, resultado desejado 2...

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

Não são *scripts* de testes detalhados, de modo que o seu objetivo é transmitir um conjunto de testes que todos os envolvidos podem percorrer para entender como o recurso funcionará.

Conheceremos um pouco de desenvolvimento orientado a testes (TDD), sendo uma abordagem evolutiva do desenvolvimento e 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 os seus testes completos, de modo que por uma questão de velocidade você pode decidir executar apenas um subconjunto para garantir que o novo teste de fato falhe. Depois, você atualiza o seu código funcional para que ele passe nos novos testes.

A quarta etapa é executar os seus testes novamente, de modo que se falharem você precisará atualizar o seu código funcional e testar novamente.

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

## Wireframe, Mapa Conceitual e Mapa Navegacional

**Wireframes:** uma imagem é necessária para cada tela envolvida, podendo ser desenhos simples em um quadro branco que são digitalizados e colocados no documento em *Office Word* ou em um conjunto de caixas criadas em *softwares* para isso e que depois podem gerar imagens para serem coladas no *Office Word* ou em algum outro software de documentação. Alguns *softwares* para isso são os seguintes:

### 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](#)

### Flowchart Software

Neste 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 poderá usar algum software livre do pacote BR Office para *Ubuntu* ou o *Microsoft Paint*, ou ainda o *Office PowerPoint*. Caso tenha conhecimento, poderá fazer direto em *HyperText Markup Language (HTML) 5*, o que lhe poupará tempo na codificação do *front* do projeto – já pensando no CSS3, inclusive.

---

## Explore:

**As seguintes leituras são 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.**

Publicado pelo canal Lucidchart Português

[ACESSE](#)

### **Aula 15. Mapa do site.**

Publicado pelo canal Rafael Arlindo

[ACESSE](#)

## 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* desse usuário. Os protótipos de *software* modernos são interativos, pois imitam o mais próximo possível o comportamento real do *software*.

Um protótipo permite que os desenvolvedores e *designers* mostrem a 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. Lembra-se de ser importante compartilhar com os colegas e submeter a críticas? Pois é disto que se trata. Passe para os outros verem e espere dos outros que também passem os deles para que sejam criticados – tudo isto tem relação com *design thinking*.

Fazer isso no início do processo de desenvolvimento economiza tempo, dinheiro e aborrecimento, de modo que existem quatro tipos de prototipagem:



Rápida (descartável);



Evolutiva;



Incremental;



Extrema.

Para este projeto estamos nos referindo apenas à primeira, dado que 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 os seus comentários. É também 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, de modo que quando todas as partes interessadas estiverem satisfeitas, torna-se uma referência para *designers* e desenvolvedores a usarem.

A prototipagem descartável também pode ser aplicada a protótipos de papel, nos quais os *designs* são simulados em pedaços de papel ou papelão, sendo, por definição, descartáveis. Então, podem ser usados para essa finalidade:

- A sequência de navegação feita em telas estáticas colocadas em uma apresentação de *Office PowerPoint* ou *Prezi*;
- Os papéis contendo as telas, incluindo as de mensagem de erro, apresentadas pelos analistas de sistemas;
- As ferramentas automatizadas de fluxos de IHC;
- Os modelos “grossos” feitos em HTML com CSS – mais empregados porque pouparam tempo do desenvolvimento do *front* da aplicação.

# Material Complementar

---

**Indicações para saber mais sobre os assuntos abordados nesta disciplina:**

---

## SITES

### Orientações básicas na elaboração de um diagrama de classes

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

<https://bitly/3nMYVoA>

## SITES

### Orientações básicas na elaboração de um diagrama de classes

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

<https://bitly/3nMYVoA>

### Backlog do produto – passo a passo para 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://bitly/3DUDHDG>

## **Diagrama de atividades**

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

<https://bitly/3Flvram>

## **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://bitly/3nQmeqy>

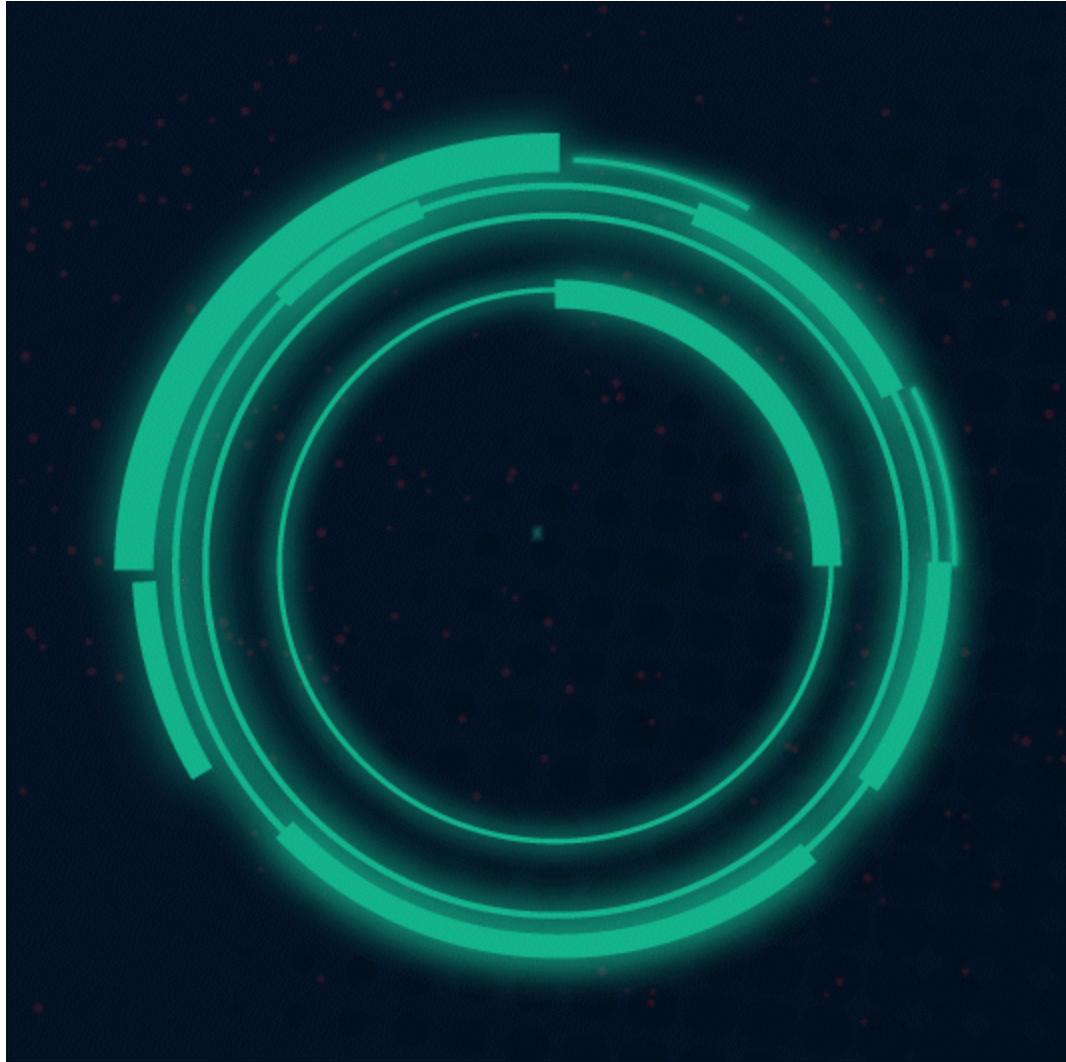
## Situação-Problema 1

---

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 o seu *smartphone* havia tocado 3h15 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 as suas vendas pela *internet*.

Tudo bem, você entra na sala de reunião e vários *stakeholders* estão presentes, inclusive o cliente, para variar, mal-humorado, também pudera, na última empresa que ele contratou os analistas não souberam eliciar os requisitos, desprezaram a opinião dele e construíram um aplicativo que não funcionava e as reclamações dos clientes foram tantas que ele simplesmente retirou do ar e, claro, teve a imagem de sua empresa arranhada, o que demandará uma abordagem extremamente cautelosa e uma análise minuciosa.

Por isso o seu diretor, logo de início, na parte de apresentações, já dispara o 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 de sua equipe para evoluirmos para a fase de planejamento do sistema – isto foi uma exigência do cliente porque como já sabemos, na última vez em que ele tentou fazer esse aplicativo foi uma “bomba” após outra.

O que se espera de você neste projeto:

- Utilize estes documentos como o seu *briefing*, descrevendo o negócio e as oportunidades:

CAFETERIA entrega mais de 15 mil kits de café da manhã durante a pandemia. **Jornal do Comércio, 10/2020**

[ACESSE](#)

COMO montar uma cafeteria. **Sebrae**

[ACESSE](#)

MIRAGAIA, M. Redes de café exploram novos hábitos de consumo e criam *delivery* de produtos. **Folha de São Paulo**

**ACESSE**

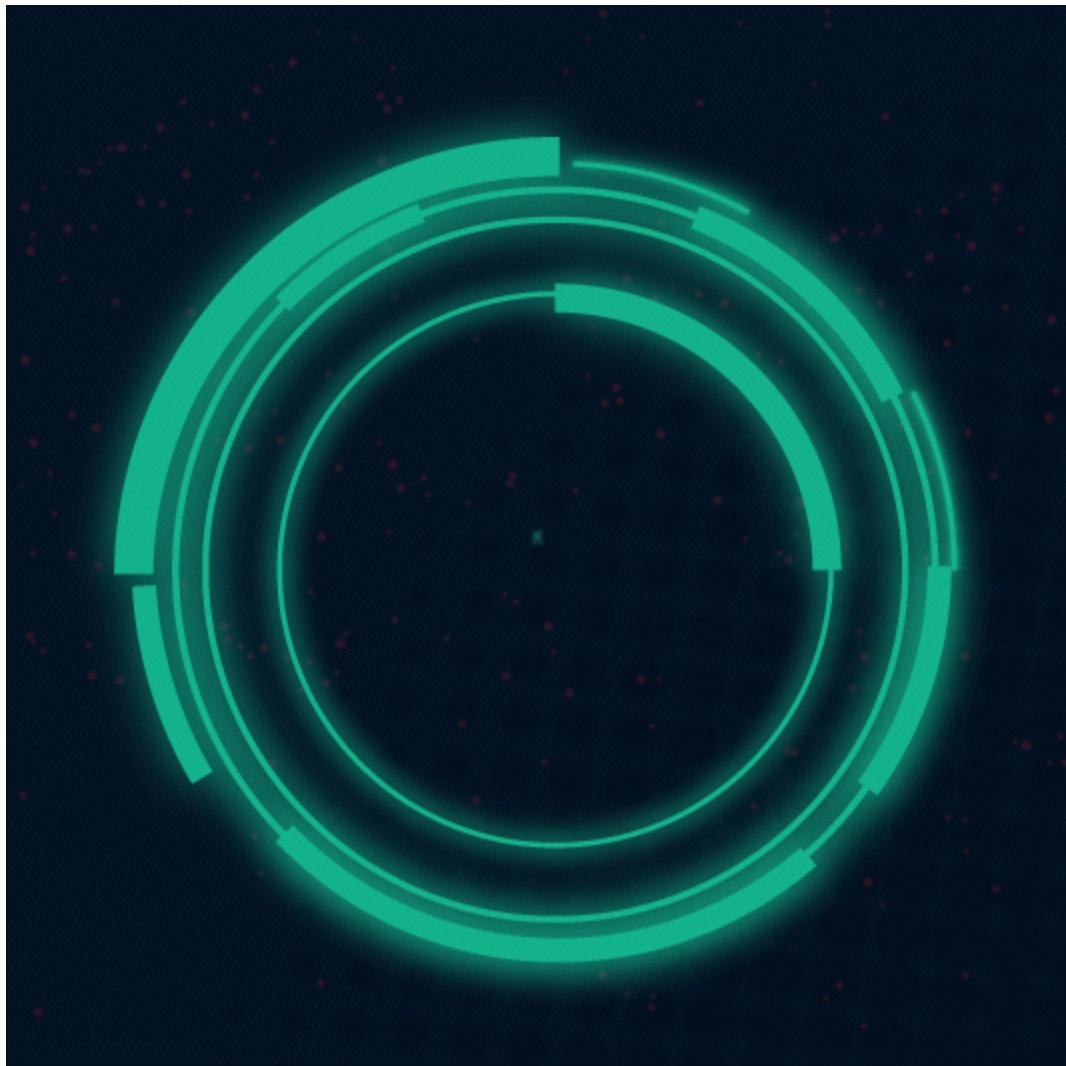
- Leia-os atentamente e anote o que for importante;
- A partir do material de apoio modele o negócio utilizando o *canva* de negócios, *design thinking* ou BPMN – (na seção de **atividade de entrega**, você encontrará um arquivo que servirá de base para a atividade);
- Agora você deverá elaborar os cartões de história do usuário para criar um sistema para uma loja virtual de café *gourmet delivery*, com foco na vitrine virtual, pedido eletrônico, pagamento e entrega;
- 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 as realizar;
- 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 cliente adoraram a sua modelagem e material de requisitos ágeis e deram "sinal verde" para que você prosseguisse com a próxima etapa, a da construção dos artefatos básicos da UML e os casos de uso expandidos.

Nesta fase esperam de você as seguintes entregas:

- Diagrama de caso de uso geral;

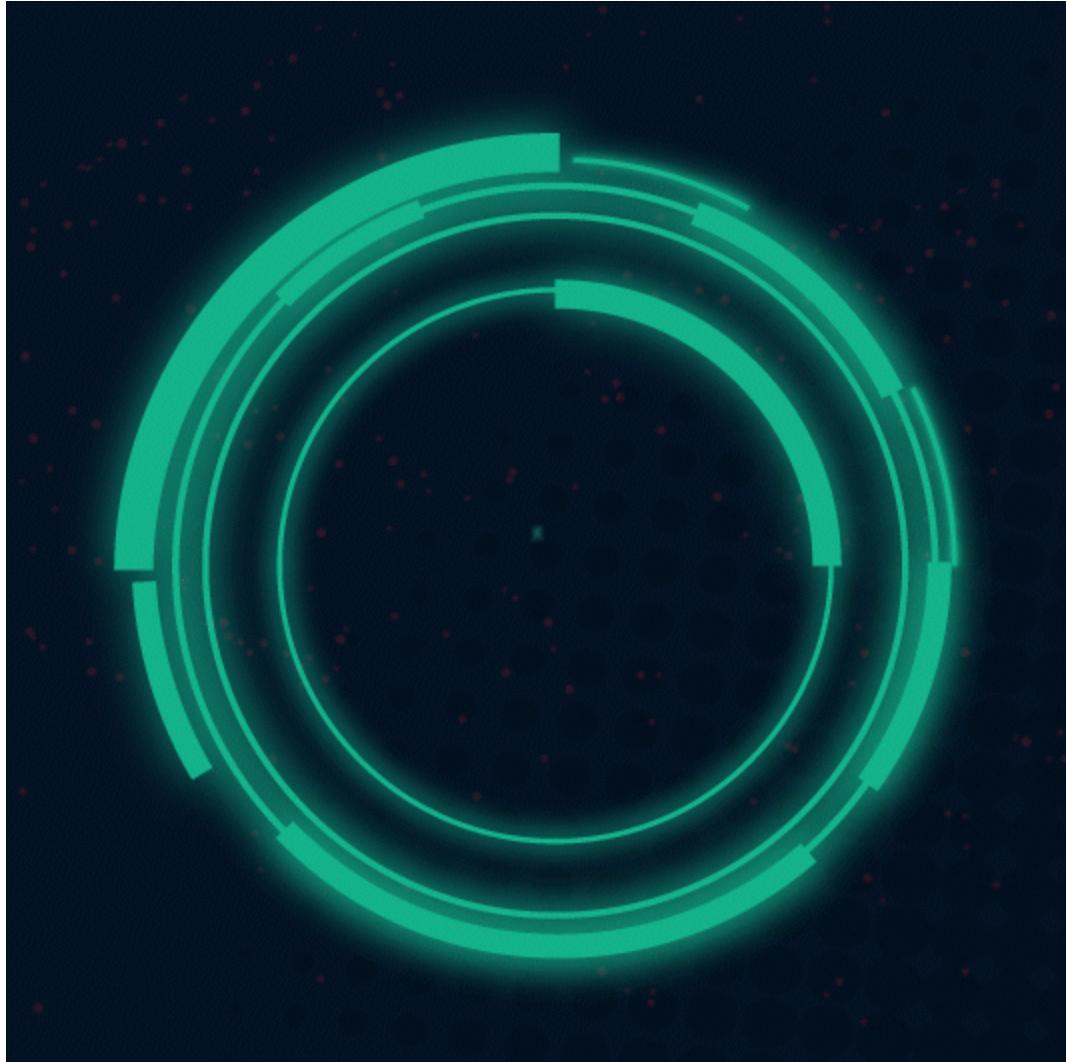
- Casos de uso expandidos;
- Diagrama de classes;
- Diagrama de sequência;
- Que apresente todos esses documentos conforme a ordem.

## 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, pois as coisas andaram bem para o seu lado.

Os seus colegas acabaram contando sobre o seu conhecimento em desenvolver *wireframes*, mapas conceituais, mapas de aplicativos *web* e protótipos rápidos para a Diretoria, que acabou "ventilando" isto com o cliente, que até o momento vem gostando do seu trabalho.

Deu "sinal verde", então, para você assumir o compromisso com a entrega de *design* da *interface* humano e computador via prototipagem rápida.

Todos estão confiantes e agora é o momento de você mostrar a sua criatividade aliando usabilidade, acessibilidade e uma excelente arquitetura de informação.

Eis o que esperam que você entregue:

- *Mockup/wireframe* das telas do aplicativo de *cupcakes*;
- Mapa conceitual;
- Mapa navegacional (mapa do aplicativo);
- Protótipo navegacional da *interface* humano e computador.

# Problema em Foco

---

Caro(a) aluno(a), vamos lá!

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

A primeira parte é fazer você modelar o negócio e escrever as histórias do usuário a partir dos textos indicados, de modo que explore a 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 *canva* 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, consequentemente, 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 maior.

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

As tarefas são estipuladas em horas, sendo recomendado estimar o tamanho das tarefas entre 2 e 12 horas; para tarefas que requerem mais que 12 horas quebre estas em várias tarefas menores que 12

horas.

Veja um exemplo a seguir das entregas esperadas:

Utilize o arquivo disponível na seção de **atividade de entrega**, e escreva, no mínimo, 15 histórias – o máximo dependerá de quão preciso você quer ser nesse aplicativo de *cupcakes*.

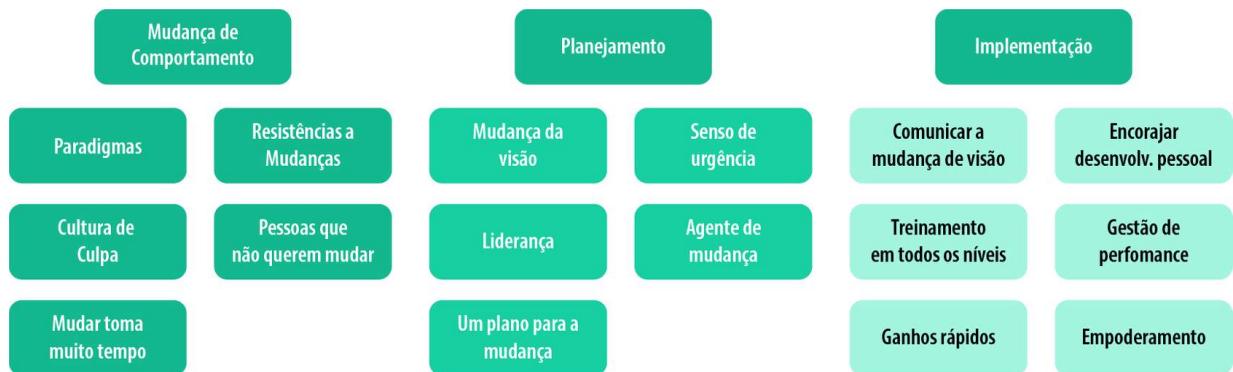
**Tabela 1 – Template de Cartão de História do Usuário**

ID:
Título:
Requerente:
Ação:
<b>Comentários:</b>
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, tal como você aprendeu.



**Figura 6 – Exemplo de Agrupamento por Afinidade de Histórias**

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. Significa que todos os cartões de história do bibliotecário compõem um tema, assim como, por exemplo, os cartões que se refiram à 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 *Office Excel*, se for o caso.

**Tabela 2 –Template de backlog de histórias dos usuários**

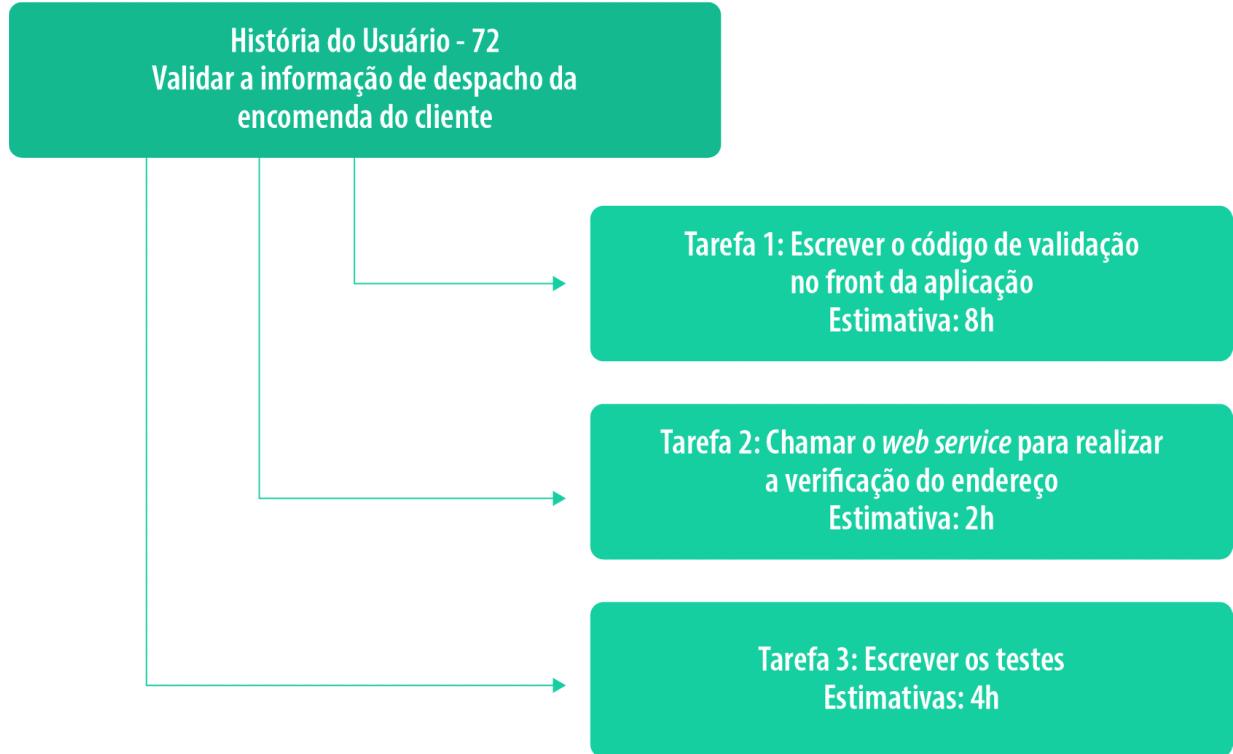
ID	História do usuário	Estimativa em pontos	Prioridade

Ficará 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

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

Organize as coisas tal como neste exemplo, sobre o que se espera que você faça para reconhecer as atividades envolvidas em cada cartão de história, além dos artefatos mencionados:



**Figura 7 – Desmembramento de Atividades e/ou tarefas**

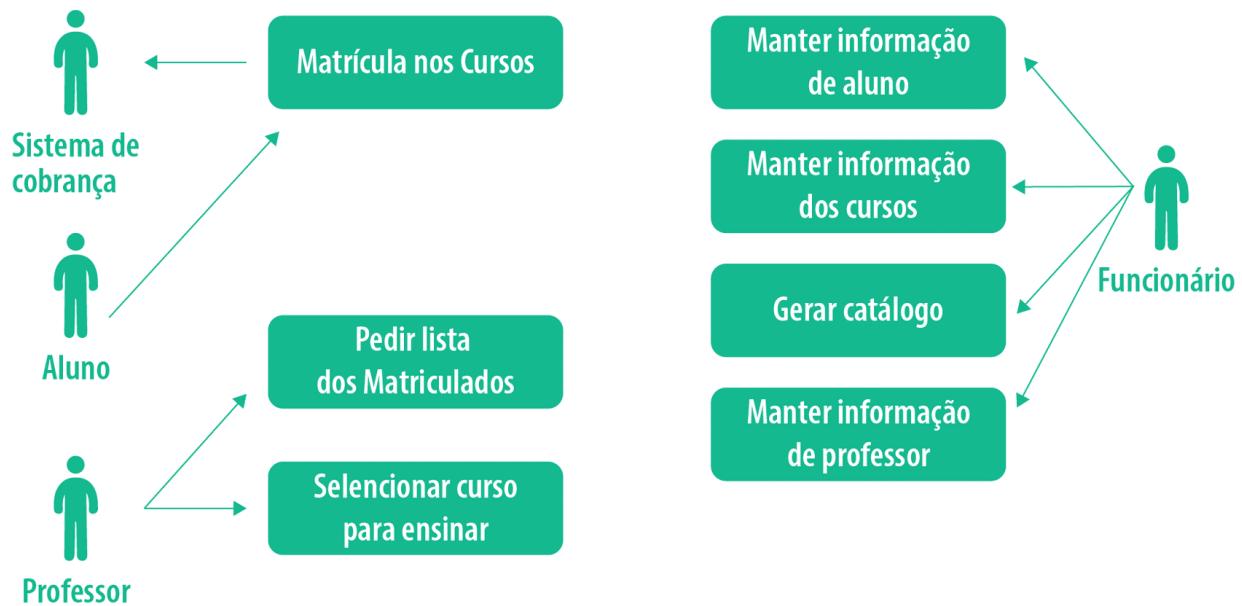
---

É apenas um exemplo, de modo que há dezenas de histórias de usuários em um aplicativo e centenas de tarefas – é delas que tiraremos as *sprints*.

Já para o segundo desafio trabalharemos a base da UML com os diagramas essenciais.

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

Primeiramente, você deverá pensar no caso de uso geral do aplicativo de *cupcakes* – vejamos um exemplo com outro aplicativo para servir de inspiração:



**Figura 8 – Exemplo de caso de uso geral**

Depois do caso de uso geral você deverá expandir os casos de uso e descrevê-los conforme apresentado no Material Teórico – lembra-se 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.

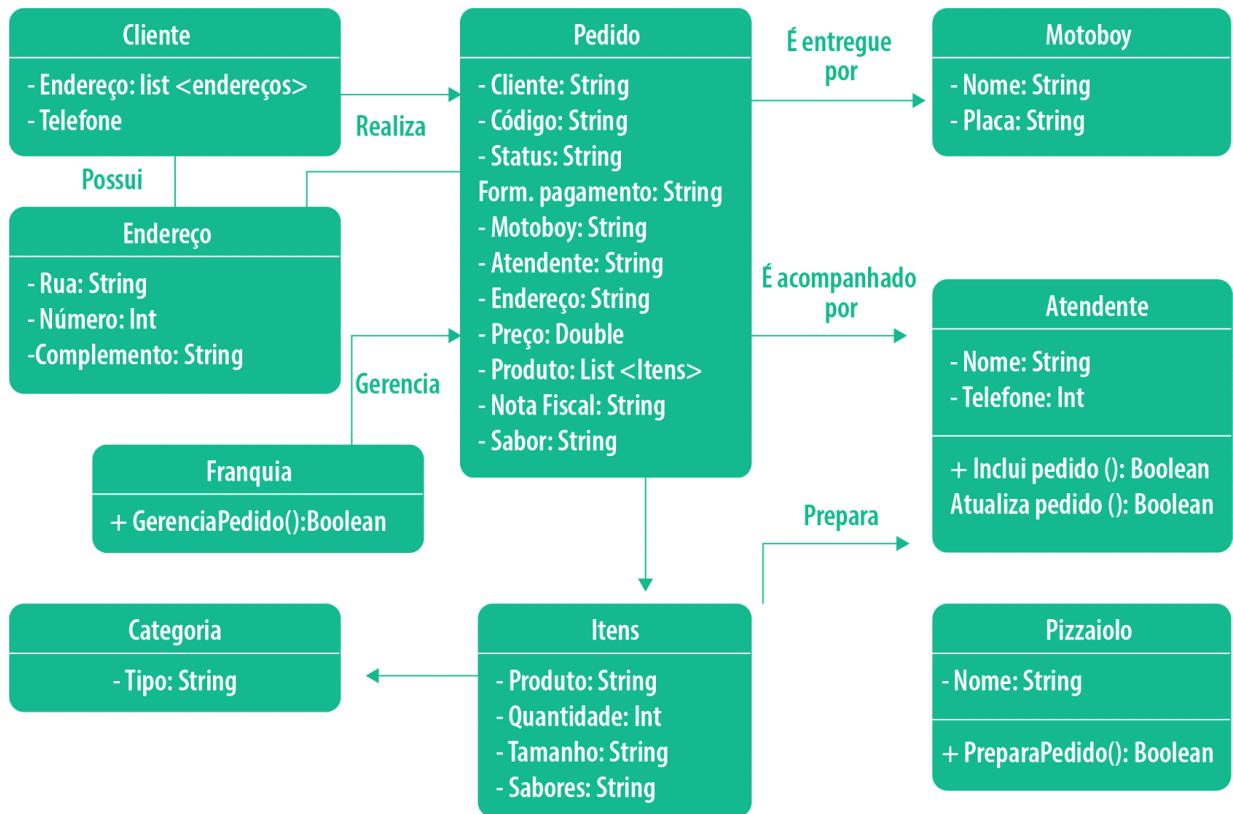
Curso alternativo Beta: o aluno decide não se inscrever em uma disciplina disponível.

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

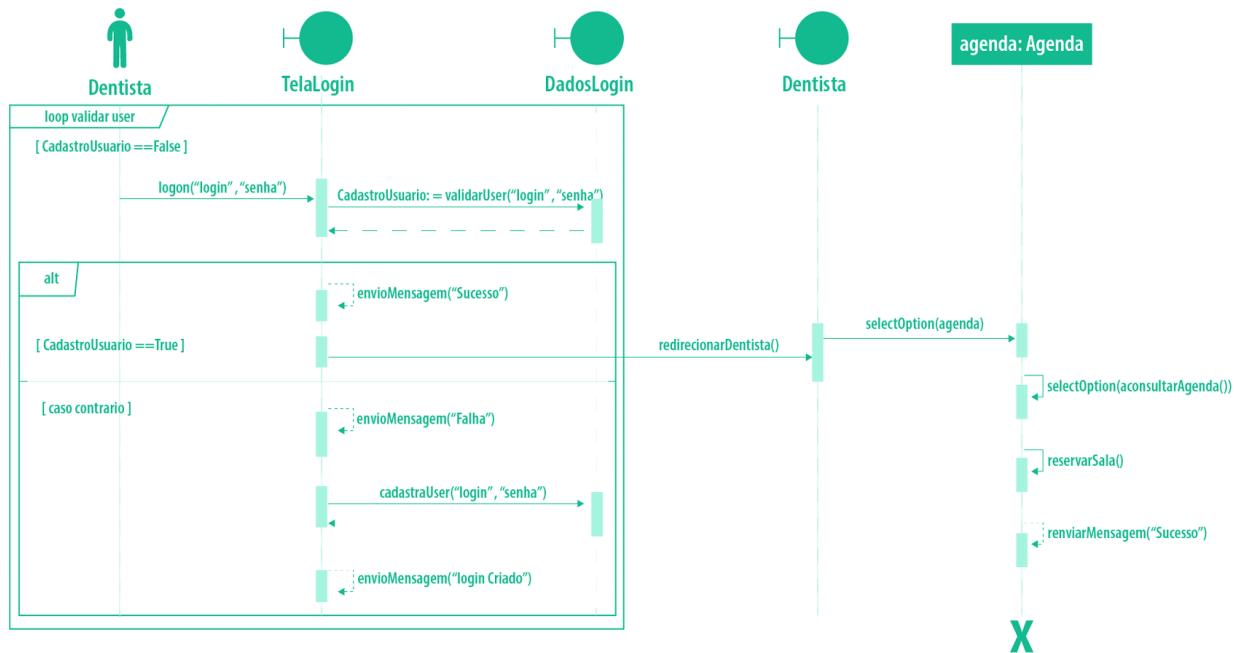
Ademais, você deverá fazer um diagrama de classes para persistência do aplicativo de *cupcakes* – a seguir temos um exemplo que foi utilizado para modelar uma pizzaria express:



**Figura 9 – Exemplo de modelagem de classes de uma pizzaria express**

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

O diagrama de sequência é um elemento essencial no desenvolvimento do aplicativo – veja outro exemplo e a partir dele desenvolva os diagramas para o aplicativo de *cupcake*:



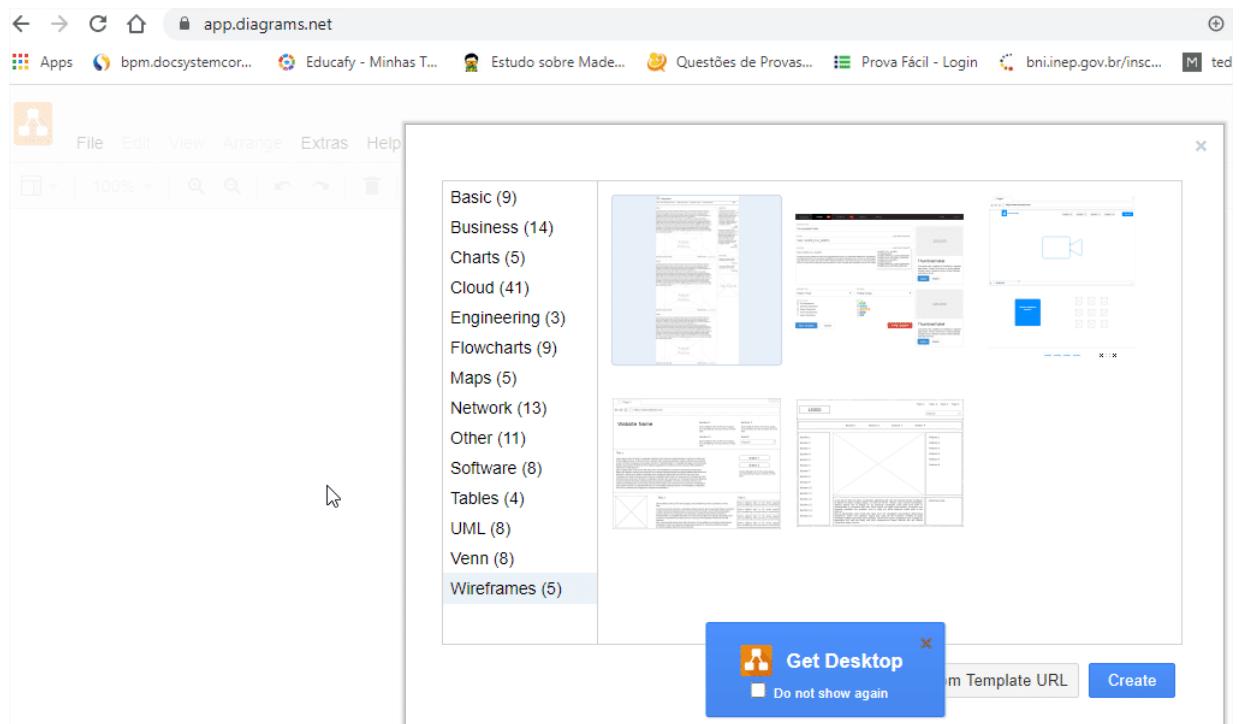
**Figura 10 – Exemplo de diagrama de sequência**

Para o terceiro desafio o importante é criar a interface gráfica do aplicativo de cupcakes e apresentar protótipos de média resolução, incluindo as telas de mensagem de erro, entre outros elementos.

Primeiramente, crie todos os *wireframes* de sua aplicação, ou seja, o que será a "cara" do produto de seu projeto. A ideia é que você complete todo o arcabouço arquitetônico visual de sua aplicação.

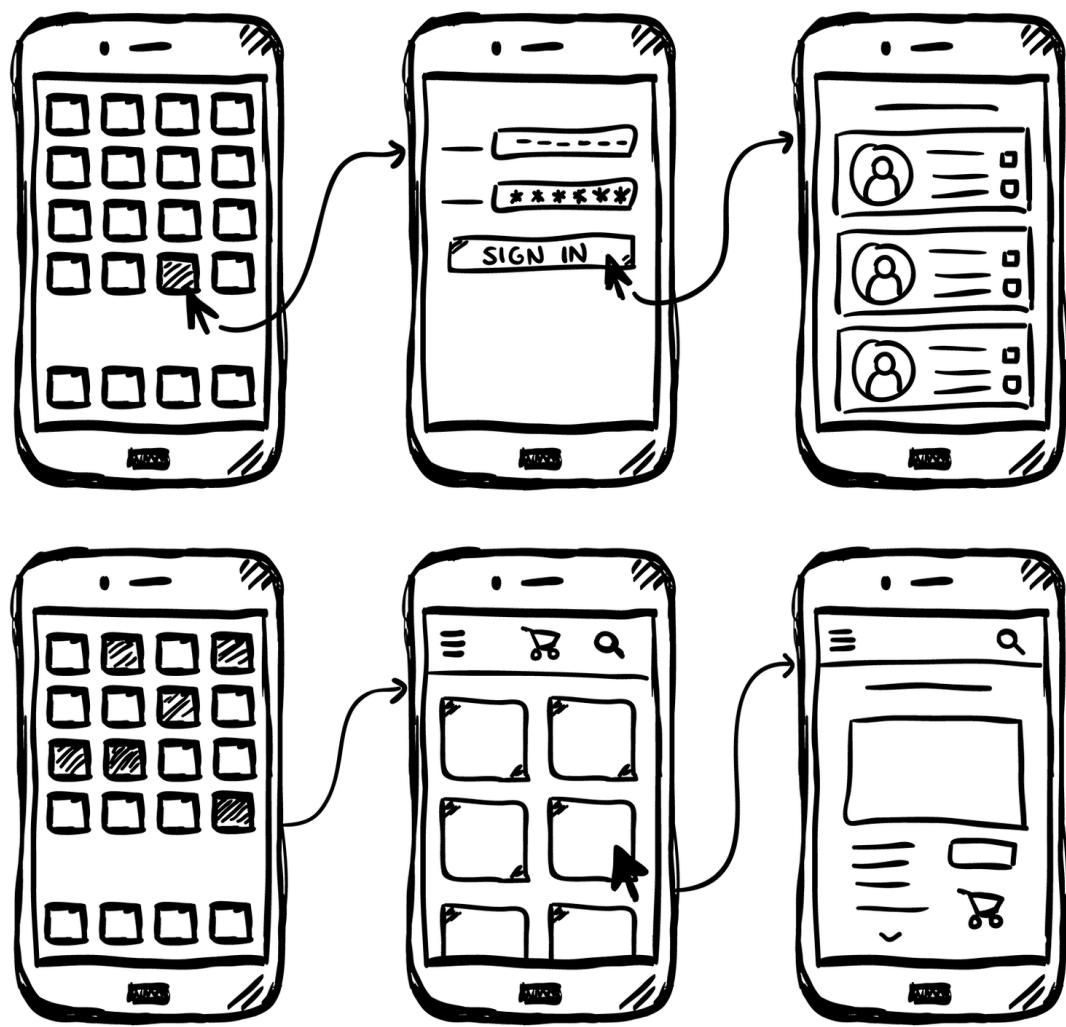
Gere um *Portable Document Format* (PDF) ou uma apresentação em *Office PowerPoint* (PPT) com os *wireframes* do aplicativo.

Você poderá utilizar o [Draw.io](#) gratuitamente e *on-line* a fim de criar todos os *wireframes*; faça isto para toda a aplicação – para tanto, utilize a opção *wireframe* do *website*.



**Figura 11 – Tela do Draw.io**

Fonte: Reprodução



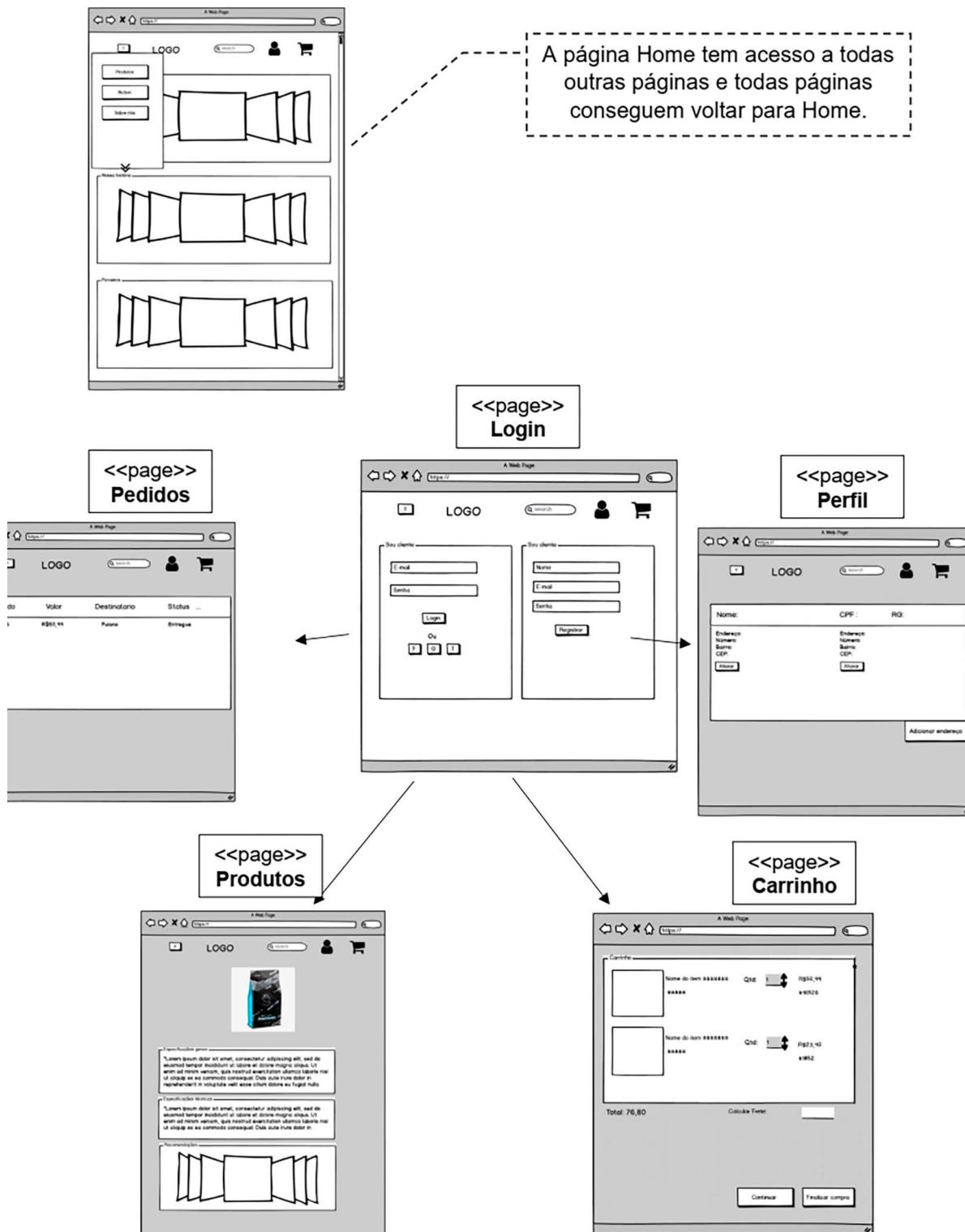
**Figura 12 – Exemplo de wireframe**

Fonte: Getty Images

---

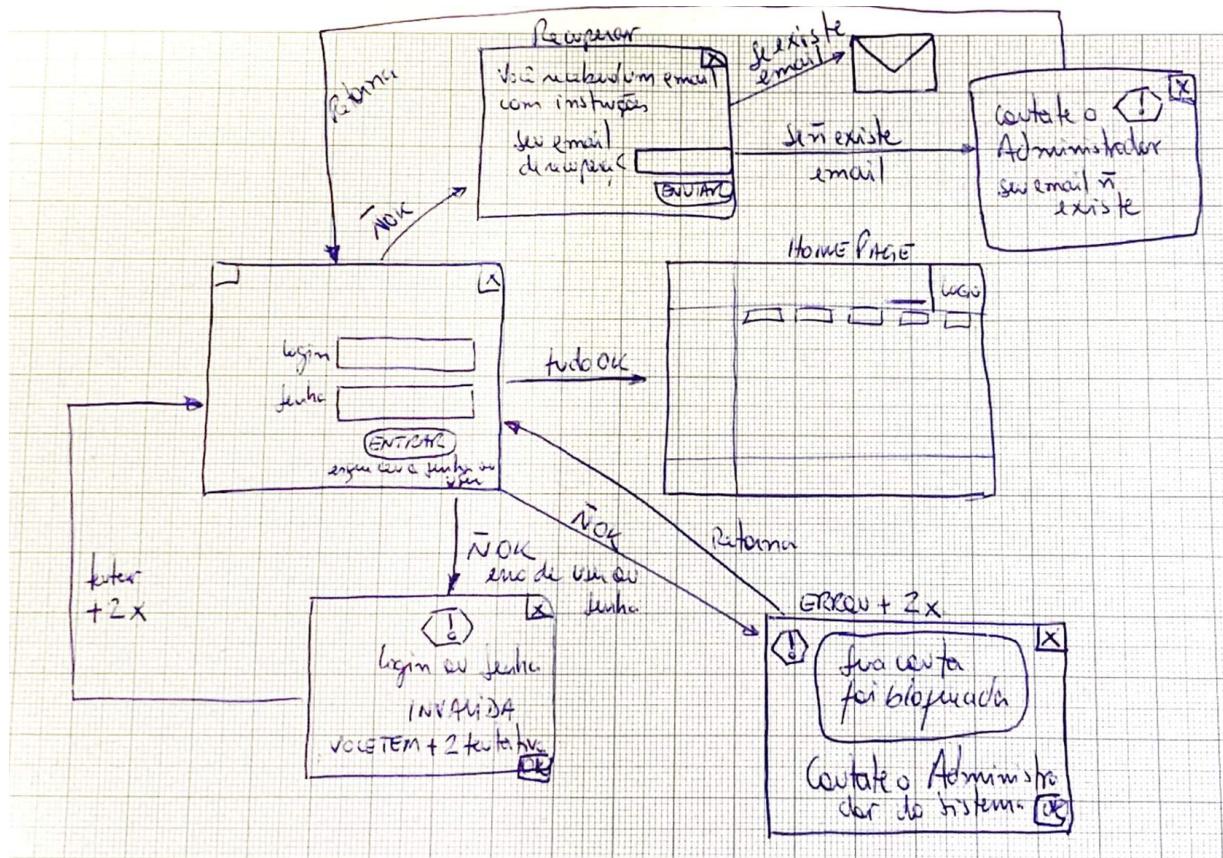
O exemplo a seguir ilustra um aplicativo, de modo que a partir da página central todas as demais vão para alguma outra página ou camada do aplicativo, sendo um tipo de mapa navegacional:

## Mapa Navegacional



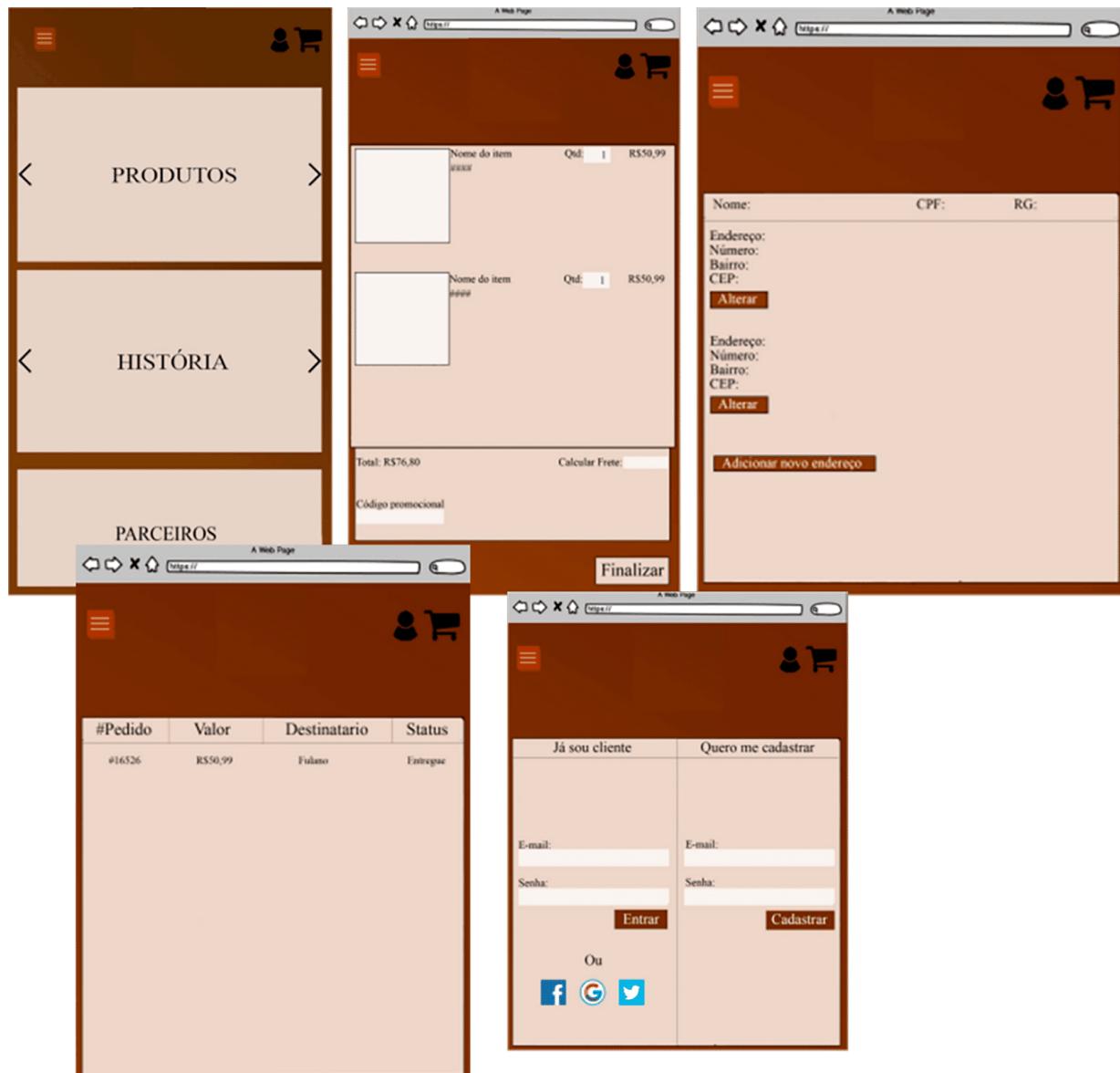
**Figura 13 – Exemplo simples de mapa navegacional**

Caso queira, você poderá fazer em um nível um pouco mais profundo e sofisticado, tal como no exemplo a seguir – realizado para cada setor (da tela de login à *home page*):



**Figura 14 – Exemplo de mapa navegacional feito à mão**

Depois disso é chegado o momento de preparar os *mockups* de média/alta qualidade já usando as cores para a aplicação. Elabore um PDF ou uma apresentação PPT com eles – a seguir há um exemplo do que você deverá fazer baseando-se no projeto:



**Figura 15 – Exemplo de mockup feito para aplicação**

Fonte: Reprodução

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

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 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.com2016/10/26/modeling-in-the-agile-age>>. Acesso em: 01/10/2021.

DRUCKER, P.F. **Clássicos Harvard Business Review**: A Teoria do Negócio. Ed. Actual: São Paulo, 2019.

LEWIS, M. **A jogada do século**: A história do colapso financeiro de 2008 que deu origem ao filme A Grande Aposte. Best Seller Ed.: São Paulo, 2011.

MAGRETTA, J. **What's your story?** 2002. Disponível em: <https://www.afr.com/politics/whats-your-story-20020614-jk7vu> Acesso em: 01/10/2021.

OSTERWALDER, A.; PIGNEUR, Y.; BONELLI, R. Business Model Generation: **Inovação Em Modelos De Negócios**. São Paulo: Alta Books, 2011.

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: 01/10/2021.

---

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