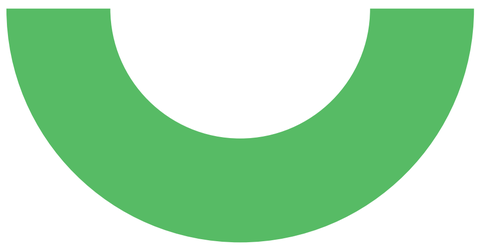
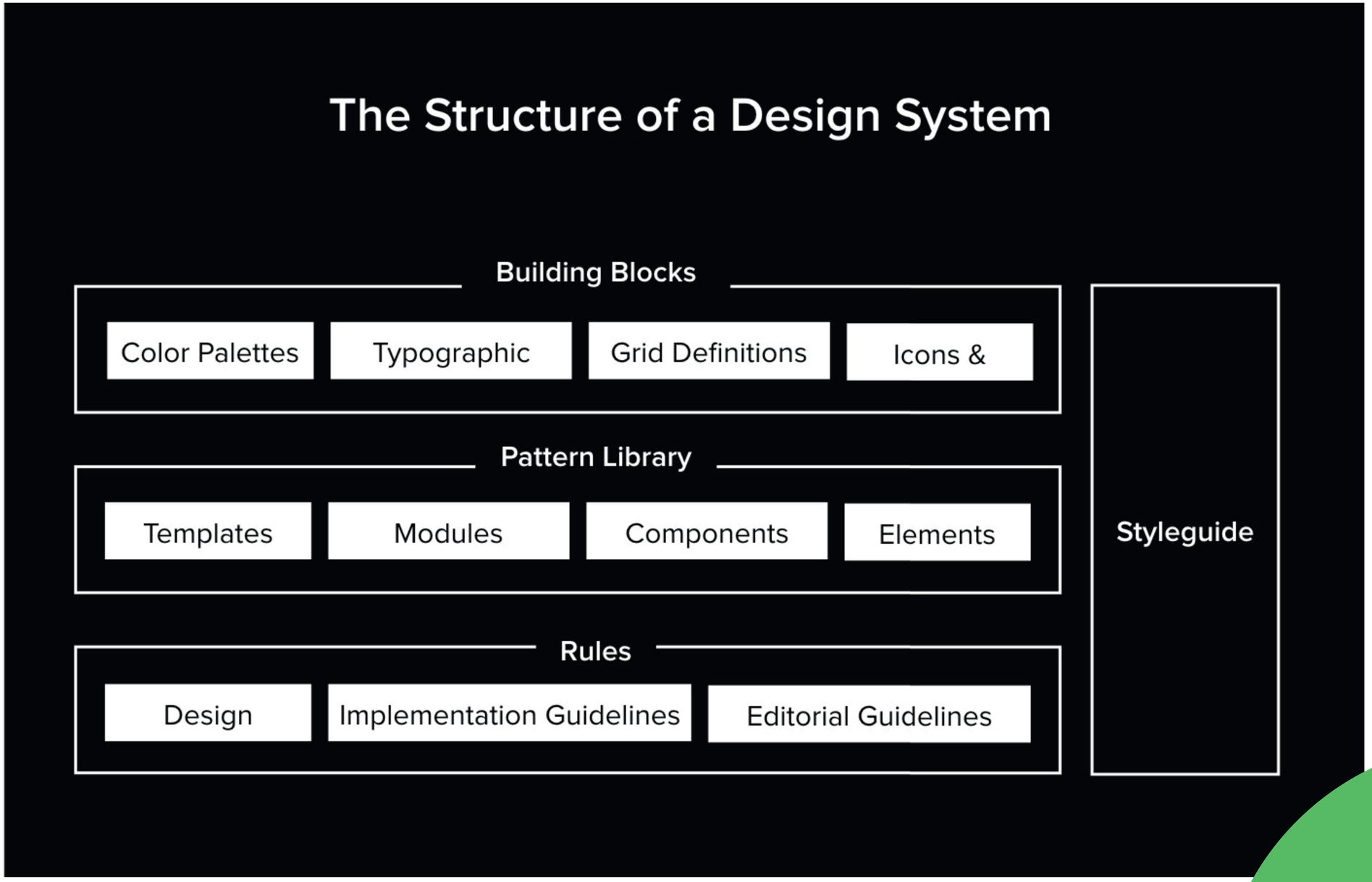


Design Systems para Angular

O que são design systems?

É o conjunto completo de padrões do projeto, documentação e princípios, juntamente com o kit de ferramentas de design e código para atingir esses padrões"

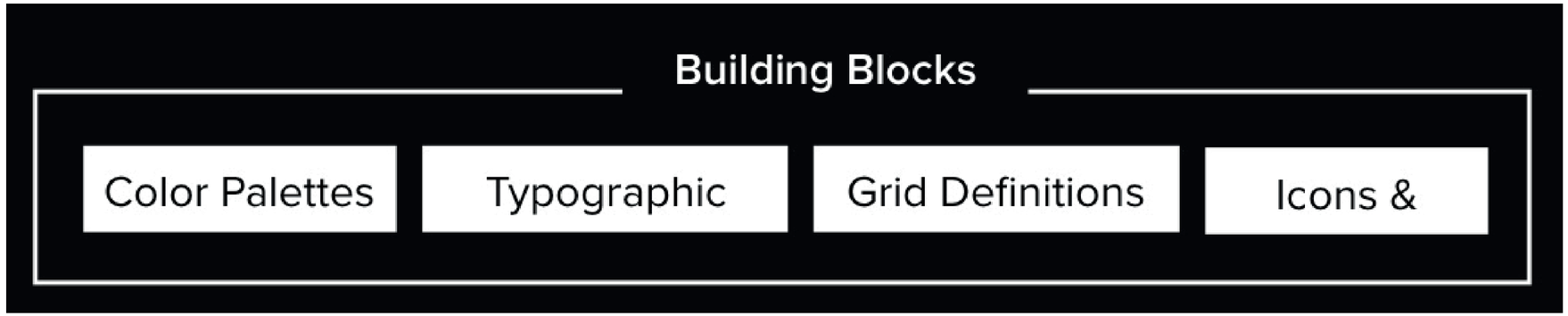
O que são design systems?



O que são design systems?

Building Blocks

Components básicos, como paleta de cores, tipografia, ícones, padrões de espaçamento, etc.



O que são design systems?

Pattern Library

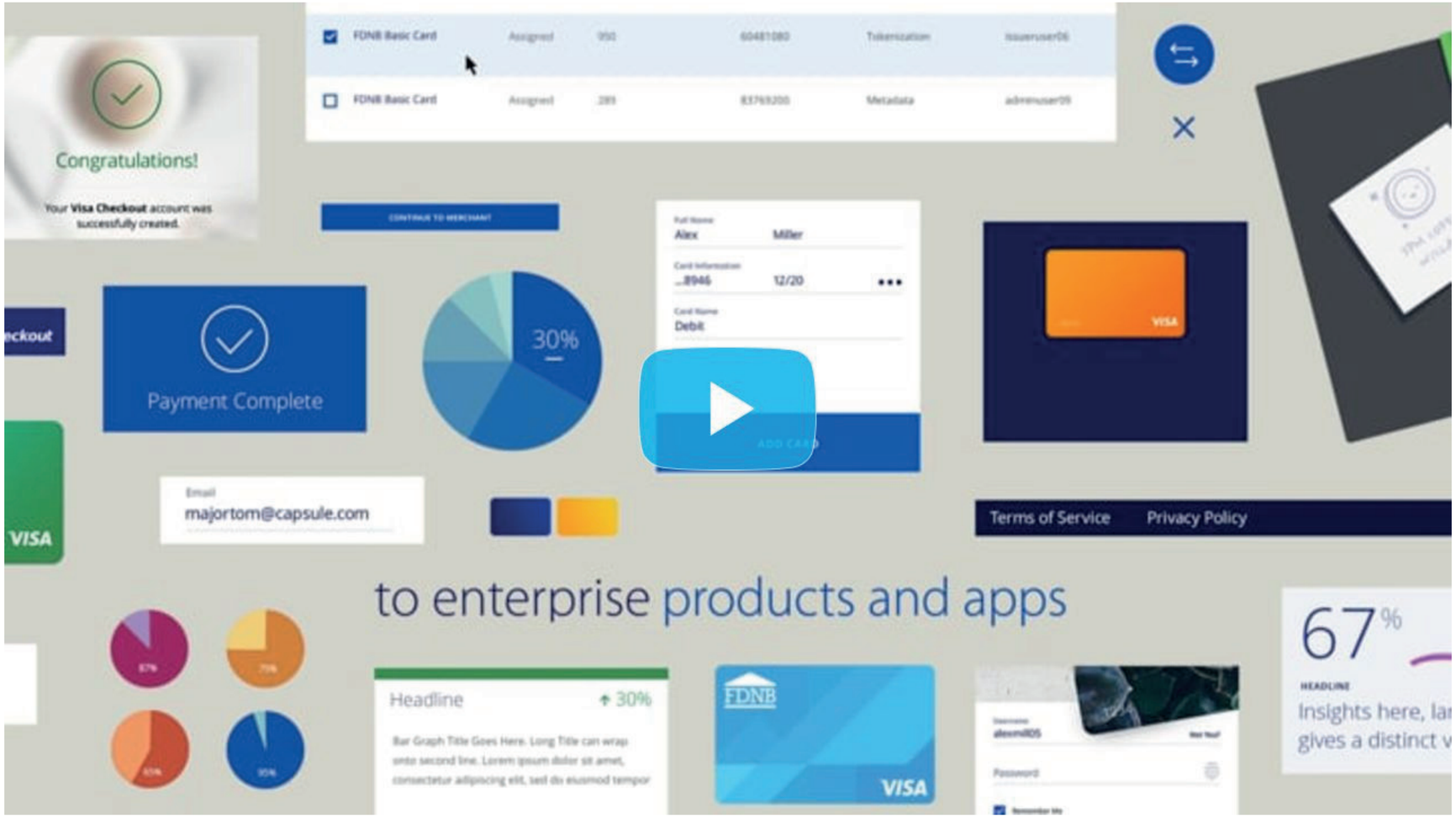
É uma subclasse no sistema de design, é o conjunto de padrões de design para uso em uma empresa.

O que são design systems?

Styleguide

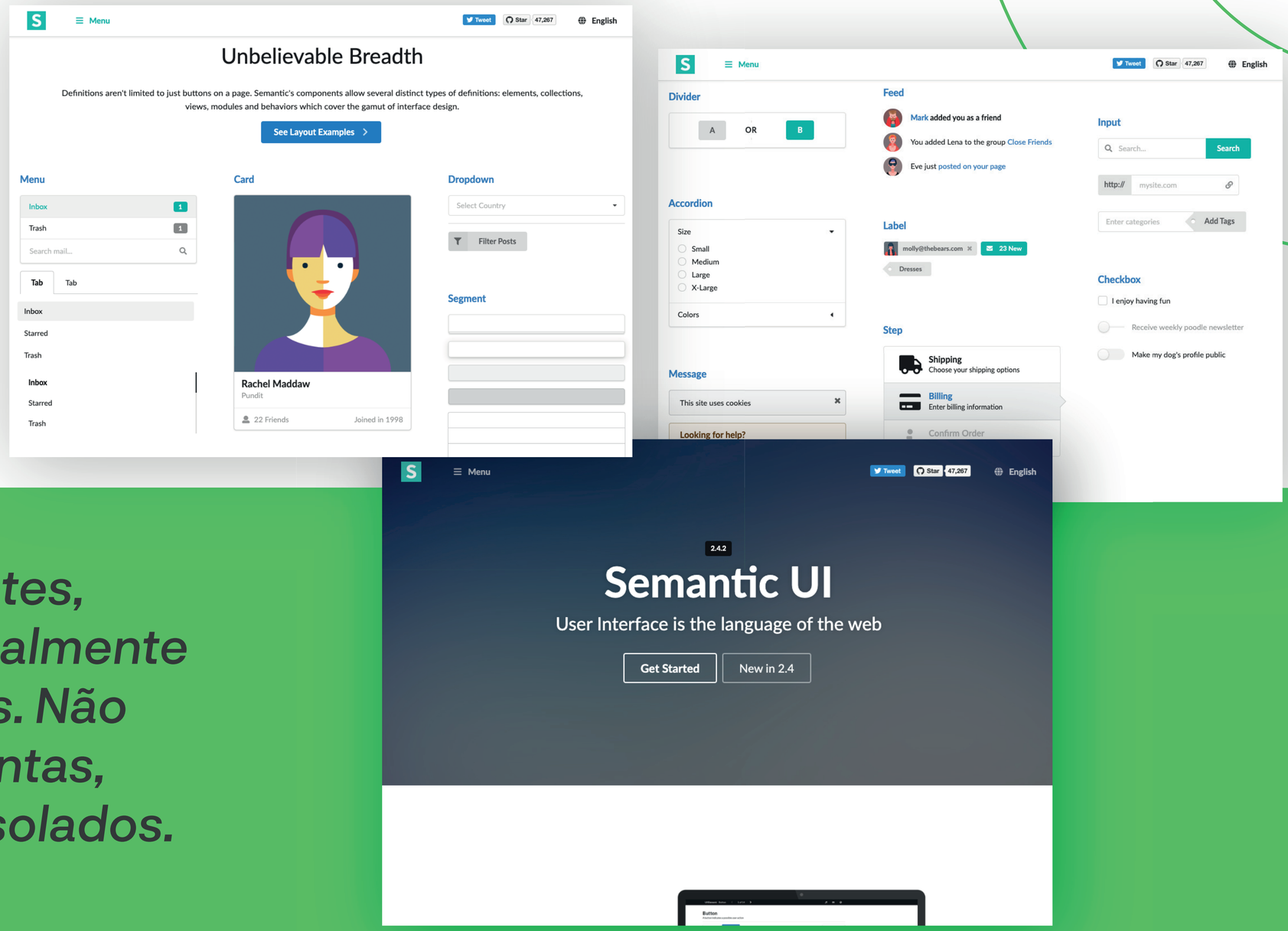
É outra subclasse no sistema de design.

Uma documentação que descreve o próprio sistema de design:
como os produtos devem parecer e se comportar, padrões de interface do usuário como tamanho de fonte, cores, botões e etc.


Congratulations!
Your Visa Checkout account was successfully created.
FDNB Basic Card
FDNB Basic Card
Payment Complete
30%
Alex
Miller
12/20
Debit
majortom@capsule.com
Terms of Service
Privacy Policy
to enterprise products and apps
Headline
30%
FDNB
VISA
67%
HEADLINE

Templates vs Bibliotecas de UI

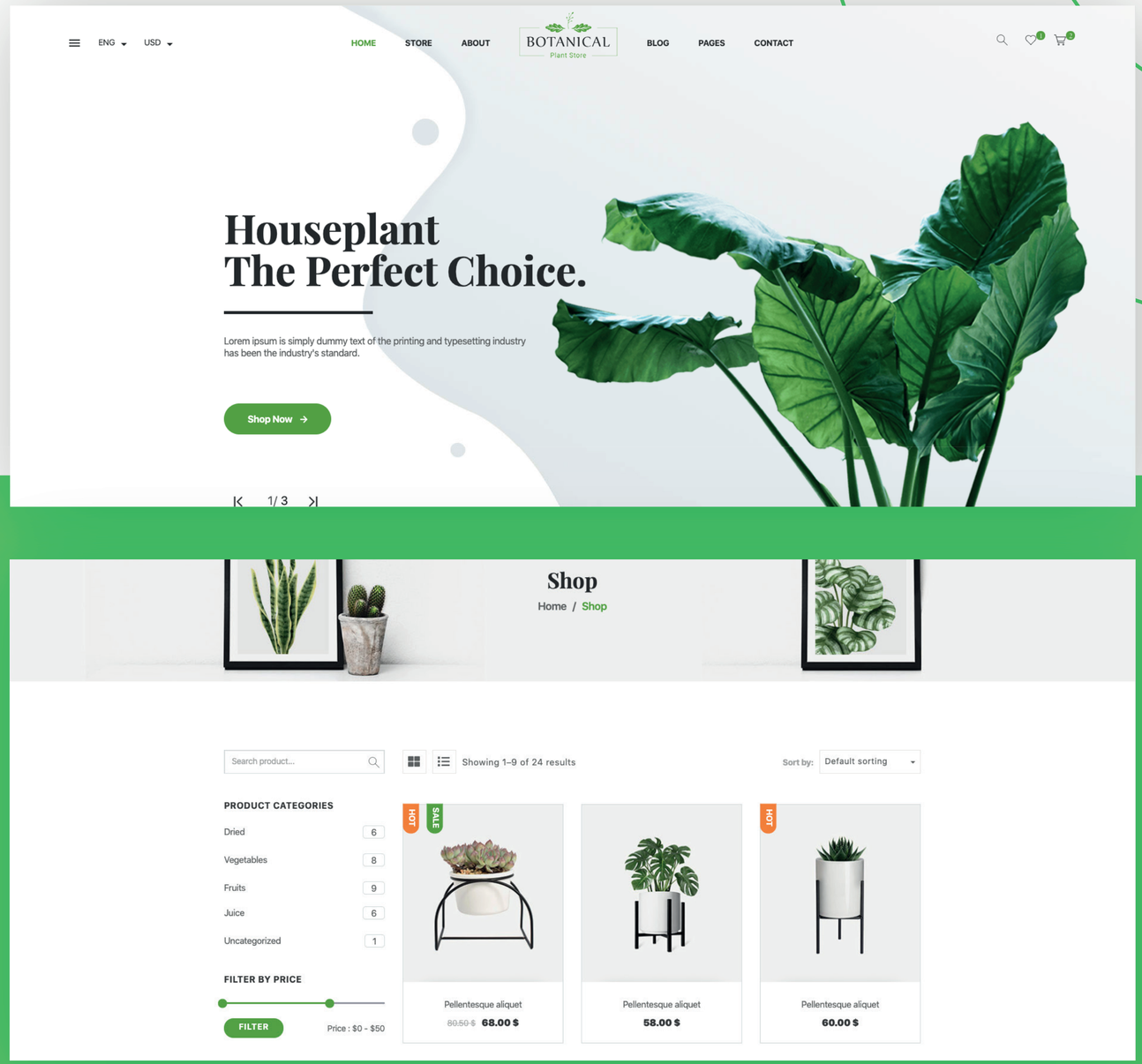
O que são Bibliotecas?



Conjunto de componentes, estilos e padrões. Normalmente customizáveis e básicos. Não envolvem telas pré-prontas, apenas componentes isolados.

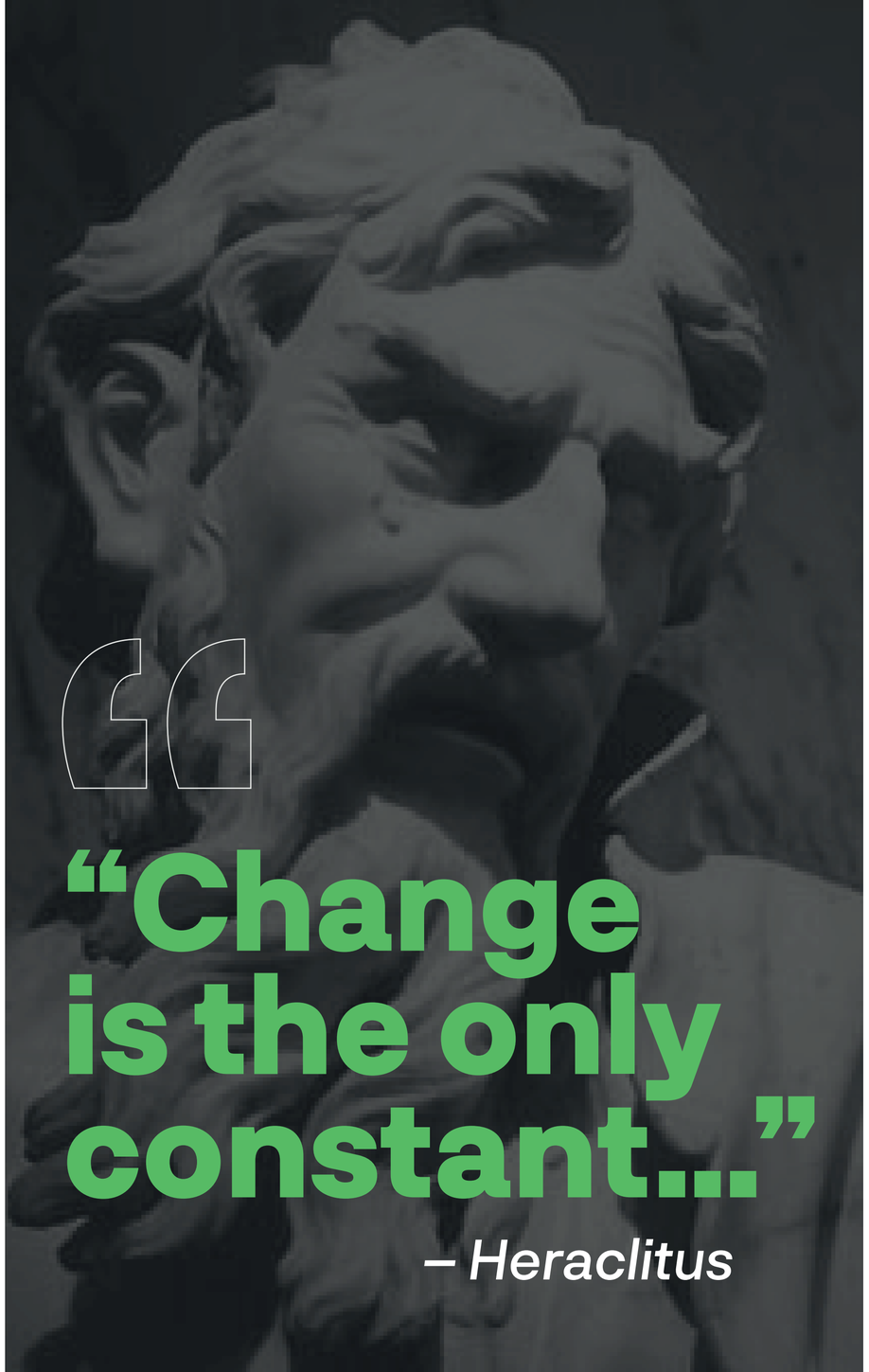
O que são Templates?

Comumente feitos em cima de alguma biblioteca. Focam em telas pré-prontas, componentes mais complexos e estilização própria. Podem também estar focados em algum nicho de negócio, como e-commerce e dashboards.



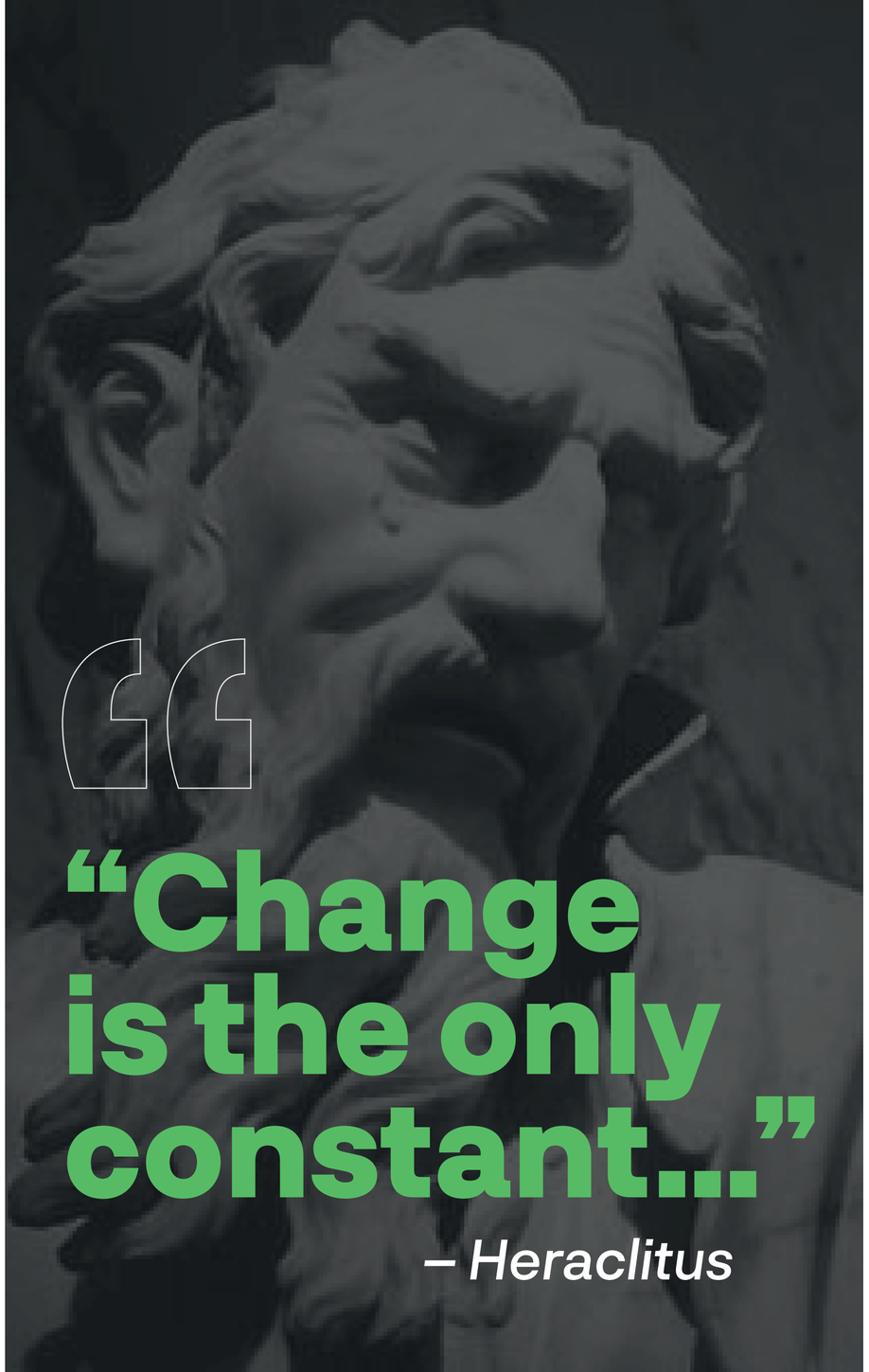
Feito em Casa vs Dependência

Pense bem antes de depender de alguém



- *O jeito como fazemos tecnologia muda muito rápido;*
- *As dependências nos ajudam a focar no código específico à nossa aplicação, nossas regras de negócio*

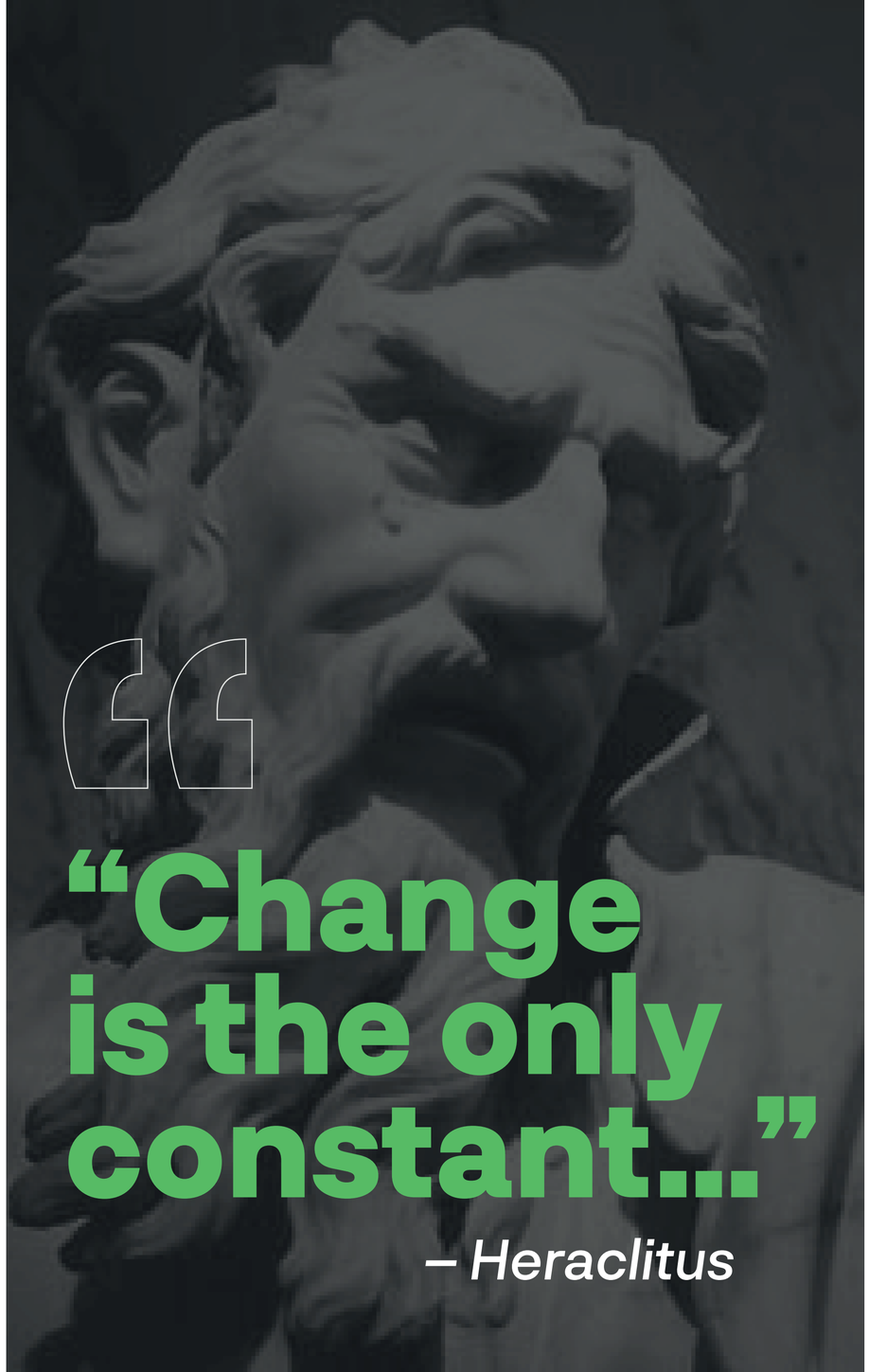
Pense bem antes de depender de alguém



Tu conta ou eu conto?

- *Nos agarramos a frameworks, bibliotecas e ferramentas como se não fossem mudar nunca.*
- *Depender de bibliotecas que resolvem coisas básicas presentes em todo de aplicacão é normal (autenticação, criptografia, http, validação de forms, etc.)*

Pense bem antes de depender de alguém



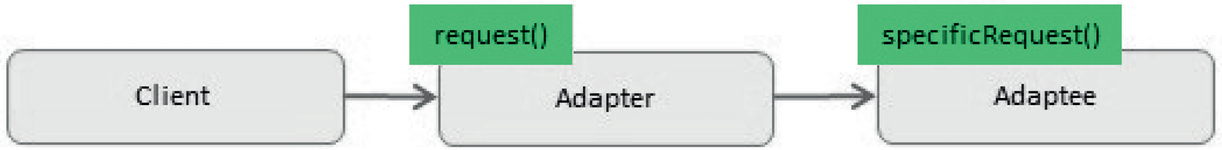
- *É melhor que nós foquemos no código específico à nossa aplicação, nossas regras de negócio;*

- *Bibliotecas podem mudar drasticamente, serem abandonadas, descontinuadas, não acompanharem as mudanças do contexto, quebrarem, conflitarem com versões de outras bibliotecas (dependency hell), etc.*

Pense bem antes de reinventar a roda.

Tens braço para resolver as broncas?

Tem hora pra tudo



Adapter pattern diagram from Dofactory.com

Antes de adicionar uma dependência, se pergunte:

- *A biblioteca tende a mudar de uma forma muito impactante?*
- *Uma mudança grande na biblioteca causaria muito dano na sua aplicação?*

Tem hora pra tudo

O que procurar?

- *A biblioteca já está por aí há muitos anos*
- *Já teve muitas major releases*
- *É utilizada por aplicações comerciais*
- *É financiada por uma empresa grande*
- *Tem uma comunidade ativa*
- *Estrelas no github? =D*

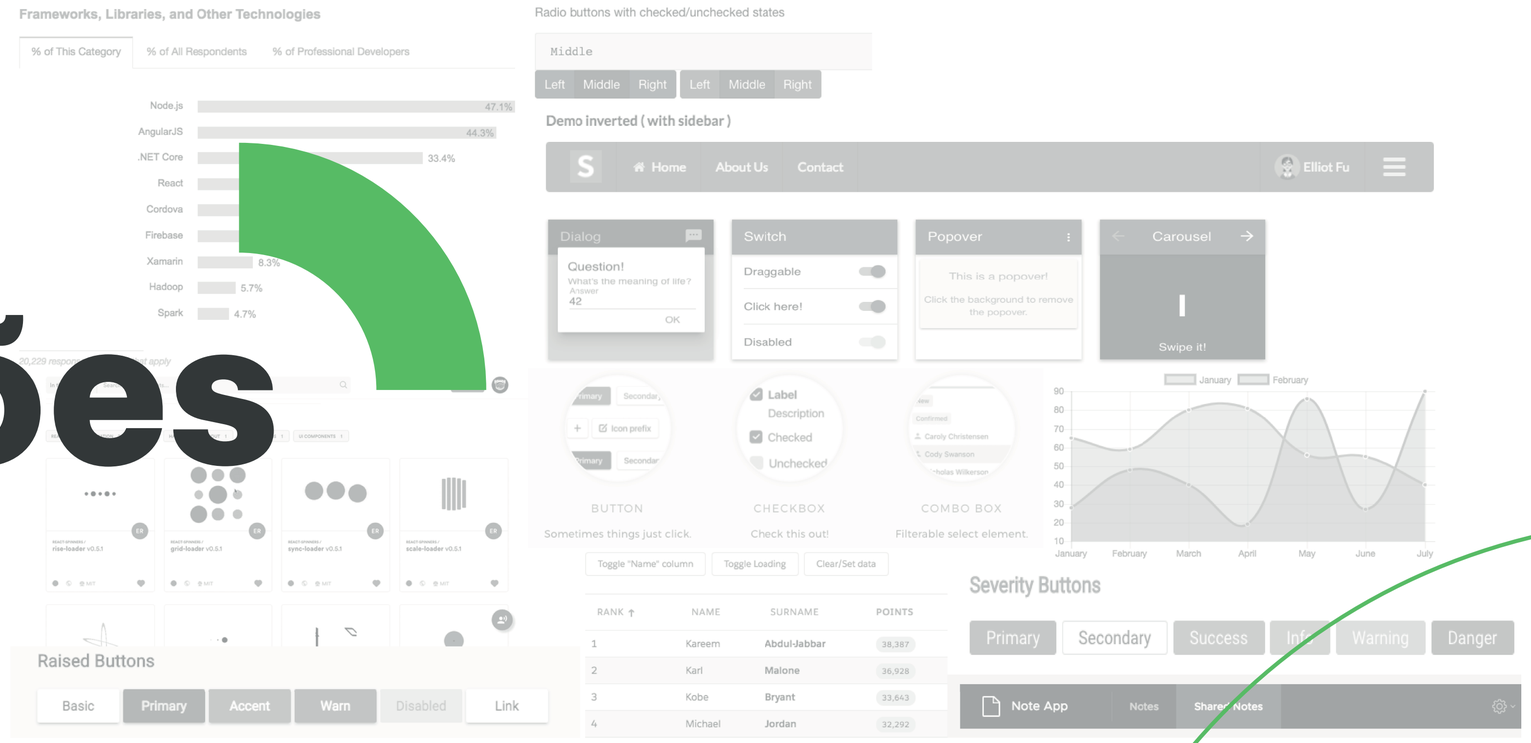
Tem hora pra tudo

Provavelmente dará certo se:

- *Só for usada numa porcão pequena da sua aplicação*
- *O código dependente não está tão atrelado às suas regras de negócio*
- *Pouco esforço seria necessário para removê-la do projeto*

Referência:
https://www.freecodecamp.org/news/code-dependencies-are-the-devil-35ed28b556d/

Opções



https://blog.bitsrc.io/11-angular-component-libraries-you-should-know-in-2018-e9f9c9d544ff

Tem muita coisa boa nesse mundo

Cuidado com as #fakenews

Não se iludir

- *Escolher uma biblioteca dessas de UI consiste em duas coisas: estilo (css) e componentes (js).*

Jogando fácil

- *Pragmaticamente: Bootstrap, NG-Bootstrap e vamos já pensando na transição e nos próximos passos.*

Casos de uso

Modal, Tooltip, Dropdown, Select, Toast, Tabs, WYSIWYG, Date Picker, Calendar, Carousel, Pagination, Progress Bar

Simplificando os conflitos com os designers:

É tudo uma questão de custo

1. *Exercitar o poder da estimativa;*
2. *Negociar com o designer versões simplificadas quando necessário (algumas vezes não dá);*
3. *Prover alternativas e seus custos.*

Quando usar cada coisa: *pergunta ao designer*

Arquitetura e Padrões

Todo mundo faz, todo mundo entrega, todo mundo mantém

(por pouco tempo, com poucas pessoas, com pouca complexidade, com poucos usuários, com muitos recursos, etc.)

Software, Entropia, TO

The second law of thermodynamics, in principle, states that a closed system's disorder cannot be reduced, it can only remain unchanged or increase. A measure of this disorder is entropy. This law also seems plausible for software systems; as a system is modified, its disorder, or entropy, tends to increase. This is known as software entropy. - Wikipedia

Software, Entropia, TO

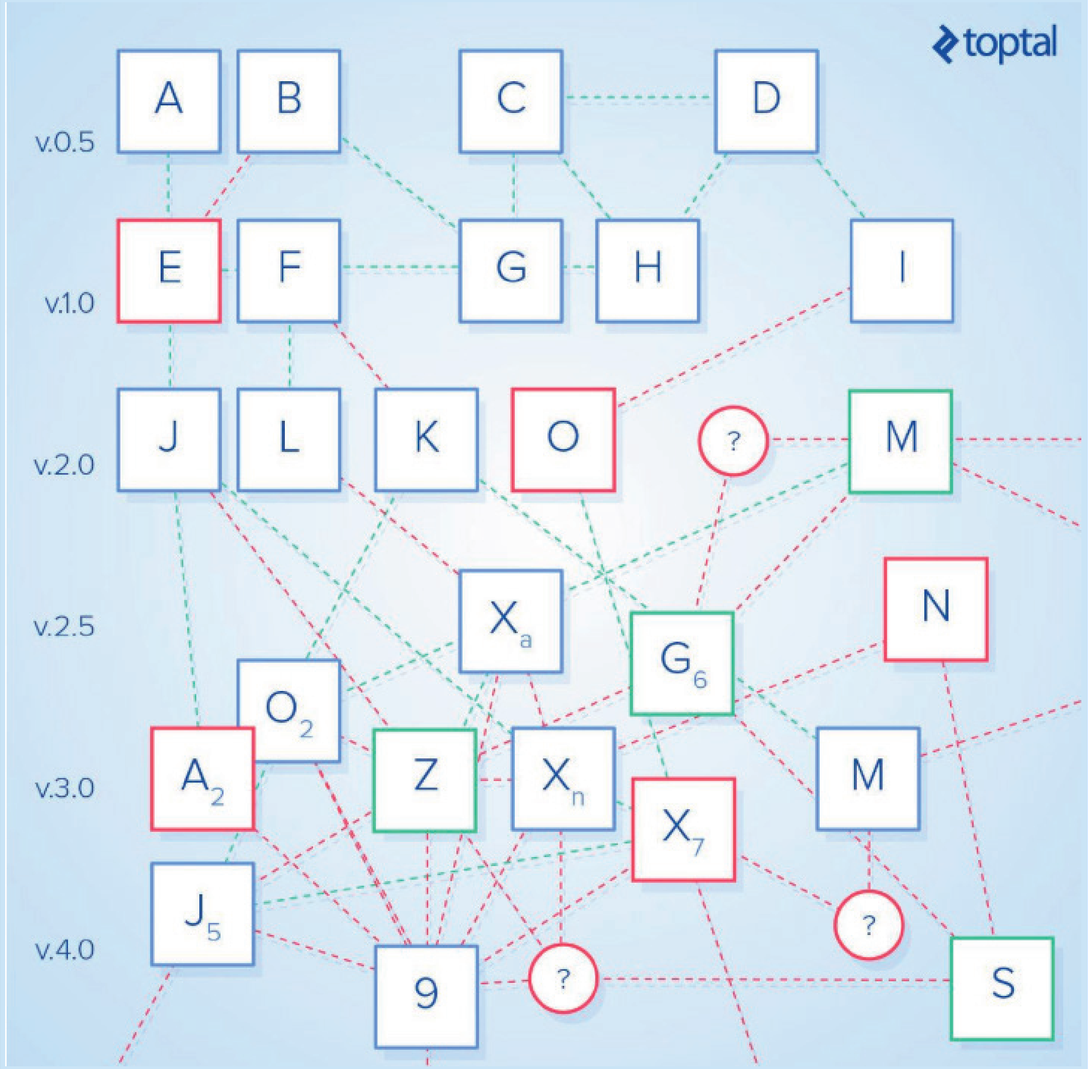
Software entropy is the risk that changing existing software will result in unexpected problems, unmet objectives, or both.

Software, Entropia, TO

It is the invisible hand that breaks component interactions that weren't in scope, causes production servers to inexplicably crash, and withholds a timely and cost-effective hotfix

Software, Entropia, TO

Expectativas de cumprimento de sprints e ciclos de desenvolvimento



Many systems with high levels of entropy rely on specific individuals, particularly if there are junior members of the development team.

Software, Entropia, TO

Expectativas de cumprimento de sprints e ciclos de desenvolvimento

- *Se a expectativa de vida do software é baixa, a probabilidade da entropia do software impactar negativamente alguma coisa é baixa;*
- *Falta de conhecimento é uma das principais causas de entropia (uma pessoa saindo do time é conhecimento indo embora);*
- *Todo ciclo de desenvolvimento tem uma probabilidade de aumentar a entropia (nunca de diminuir);*

Software, Entropia, TO

Expectativas de cumprimento de sprints e ciclos de desenvolvimento

- *Se focarmos sempre em refatoração, estaremos sempre mitigando os problemas causados pelo ciclo anterior;*

- *Se focarmos em cultura (conhecimento!), diminuiremos a probabilidade dos próximos ciclos aumentarem a entropia.*

“

Cedo ou tarde, adicionar uma mudança será mais caro que refazer o sistema inteiro”

Cedo ou tarde, adicionar mudança será mais caro que refazer o sistema inteiro

Duas opções:

- *Fazer ou Fazer Bem*

Spoiler:

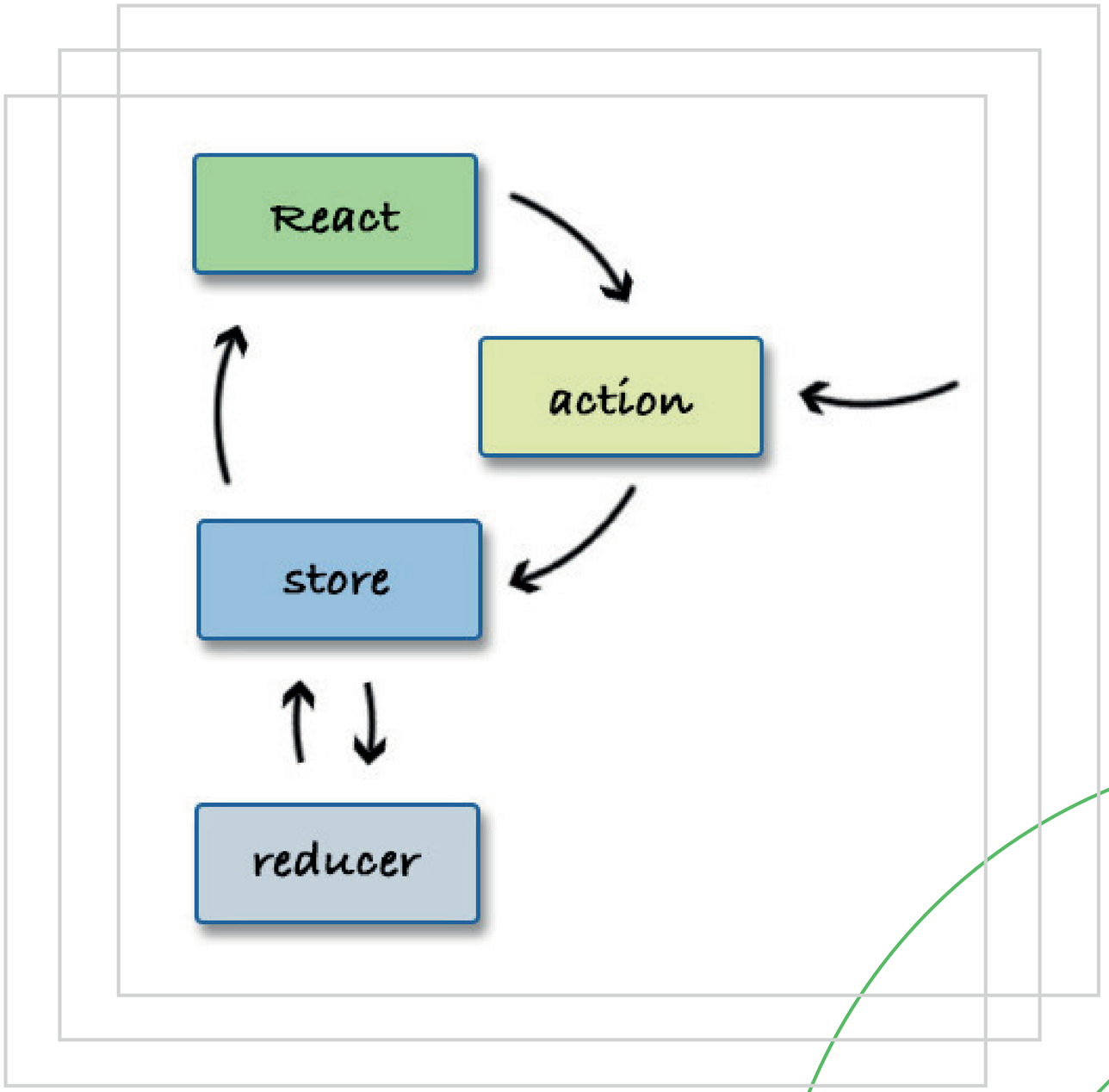
Sempre faça (o) bem!

Referência:
https://www.toptal.com/software/software-entropy-explained

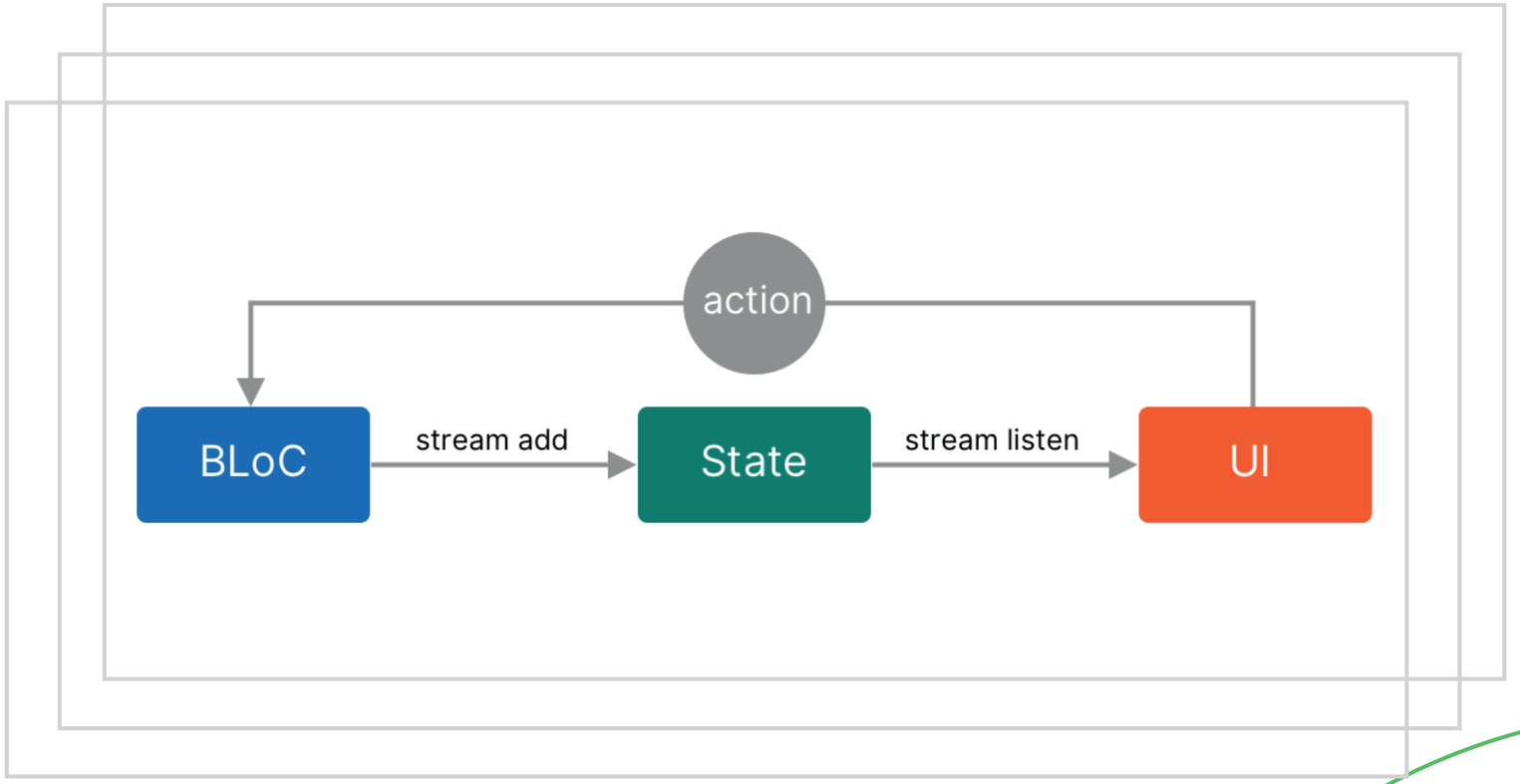
Redux, MVC, MVVM, BLoC, Facades

Um balaio de arquitetura

Redux

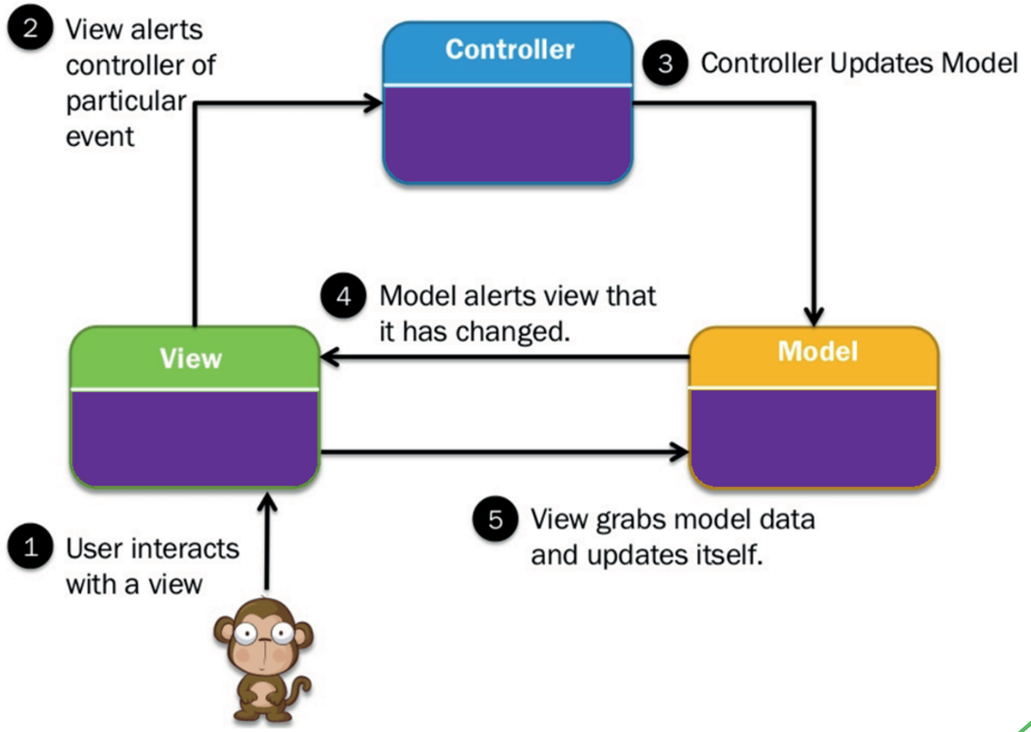


BLoC



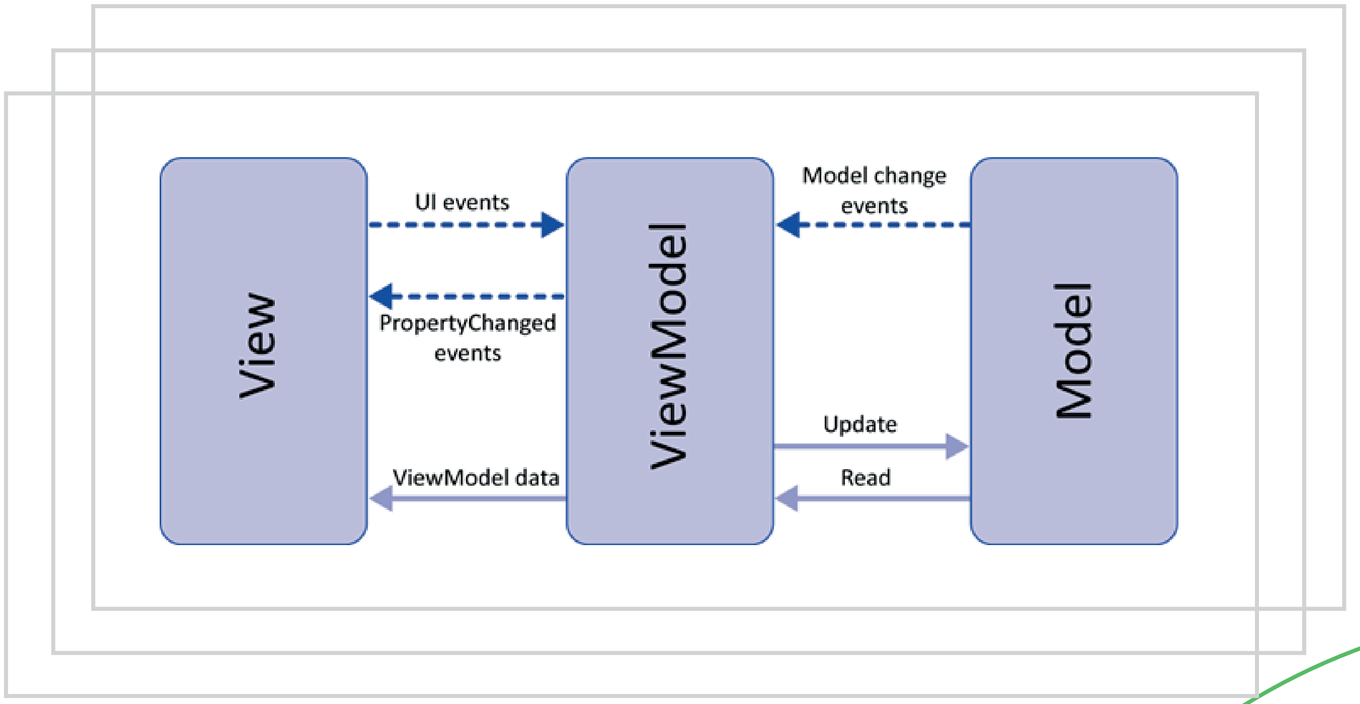
MVC

Model-View-Controller (MVC) é um padrão arquitetônico usado na engenharia de software em que o padrão isola a "lógica de domínio" (a lógica do aplicativo para o usuário) da interface do usuário (entrada e apresentação), permitindo o desenvolvimento independente, teste e manutenção de cada (separação de preocupações).



MVVM

Model-View-ViewModel (MVVM) é um padrão de design arquitetônico para implementação de interfaces de usuário. Seu foco principal é a separação de preocupação entre a View (UI) e o Model (Dados) usando uma camada intermediária chamada ViewModel para melhorar a capacidade de gerenciamento, escalabilidade e testabilidade.



Referência:
@UzumakiArtanis https://pt.stackoverflow.com/a/212181

Um balaio de prós e contras, vantagens e desvantagens

A arte de fazer o simples

Todo sistema é simples até que se prove o contrário

(sem inocência, por favor)

Tudo que você complicar será usado contra você no futuro

DRY e as abstrações prematuras

DRY e as abstrações prematuras

Como gerar uma abstração prematura e se arrepender depois:

1. *Começe a resolver um problema grande*
2. *Encontre vários problemas menores que parecem ser repetidos e terem a mesma natureza*
3. *Faça uma abstração que evita repetição de código*
4. *Ache um sub problema ligeiramente diferente e faça uma alteração para acomodá-lo*

Repita o passo 4 até se arrepender