
Newton
Quem se prepara, não para.

Arquitetura de Sistemas

7º período

Professora: Michelle Hanne

Método de Análise de Arquitetura

Newton
Quem se prepara, não para.

- O **SEI** (Software Engineering Institute) desenvolveu um método de análise dos prós e contras de uma arquitetura (ATAM, *architecture trade-off analysis method*)

Método de Análise de Arquitetura

Newton
Quem se prepara, não para.

1. **Coletar cenários:** é desenvolvido um conjunto de casos de uso para representar o sistema sob o ponto de vista do usuário.
2. **Levantar requisitos:** restrições e descrição do ambiente, parte da engenharia de requisitos.

Método de Análise de Arquitetura

Newton
Quem se prepara, não para.

3. Descrever os estilos/padrões de arquitetura: o(s) estilo(s) de arquitetura deve(m) ser descrito(s) usando uma das seguintes visões de arquitetura:

- **Visão de módulos** para a análise de atribuições de trabalho com componentes e o grau que foi atingido pelo encapsulamento de informações.
- **Visão de processos** para a análise do desempenho do sistema.
- **Visão de fluxo de dados** para análise do grau em que a arquitetura atende às necessidades funcionais.

Método de Análise de Arquitetura

Newton
Quem se prepara, não para.

4. **Avaliar atributos de qualidade:** considerando cada atributo isoladamente. O número de atributos de qualidade escolhidos para análise é uma função do tempo disponível e da relevância para o sistema em questão.

5. **Identificar a sensibilidade dos atributos de qualidade:** em relação a vários atributos de arquitetura para um estilo de arquitetura especifico.

Método de Análise de Arquitetura

Newton
Quem se prepara, não para.

6. Criticar arquiteturas candidatas: usando a análise de sensibilidade realizada na etapa 5. O SEI descreve esse método da seguinte maneira -

Uma vez determinados os pontos de sensibilidade da arquitetura, encontrar o ponto de balanceamento é simplesmente identificar os elementos da arquitetura para os quais vários atributos são sensíveis.

Método de Análise de Arquitetura

Newton
Quem se prepara, não para.

As técnicas de revisão de arquitetura mais usadas no setor são: raciocínio baseado na experiência, avaliação de protótipo, revisão de cenário e uso de checklists.

Método de Análise de Arquitetura

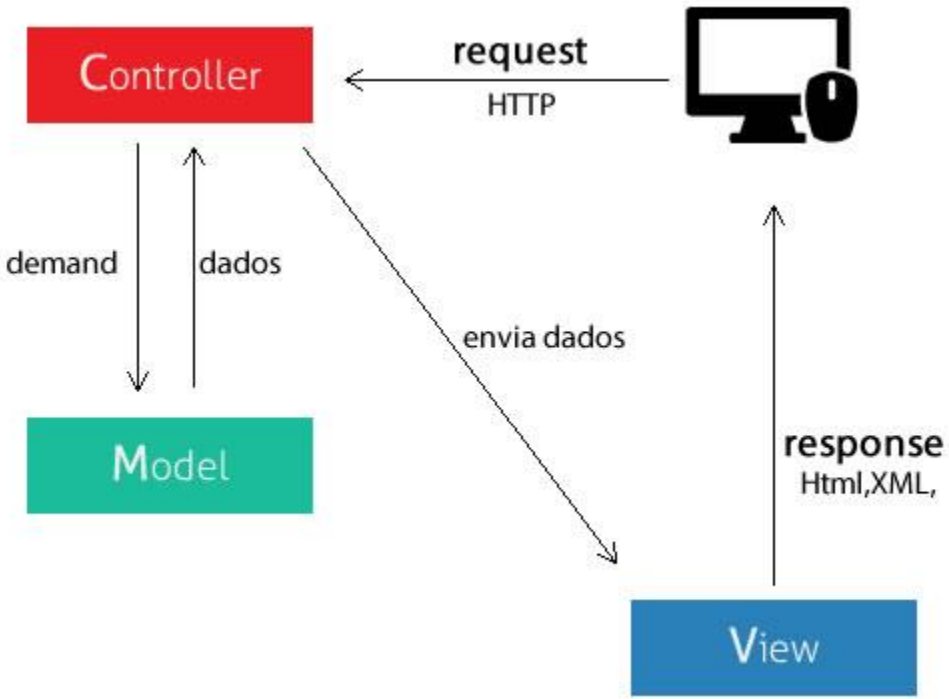
Newton
Quem se prepara, não para.

- A arquitetura de software representa a estrutura e a organização dos **componentes de software**, suas propriedades e as **conexões entre eles**.
- **Os componentes de software** incluem **módulos de programas** e as várias **representações de dados** manipulados pelo programa.

Padrão de Arquitetura MVC

Newton
Quem se prepara, não para.

Model-view-controller (*MVC*) é um padrão de arquitetura de software que divide a aplicação em três camadas: **Model, View e Controller.**



Modelo MVC

Newton
Quem se prepara, não para.

M (MODEL)

A camada *Model* (modelo) é responsável pela leitura, escrita e validação dos dados. Nesta camada são implementadas as regras de negócios. Sempre que você pensar em manipulação de dados, pense em model.

V (VIEW)

A camada *View* (visão) é responsável pela interação com o usuário. Nesta camada são apresentados os dados ao usuário. Os dados podem ser entregues em vários formatos, dependendo do que for preciso, como páginas HTML, arquivos XML, documentos, vídeos, fotos, músicas, entre outros.

C (CONTROLLER)

A camada *Controller* (controlador) é responsável por lidar com as requisições do usuário. Ela gerencia as ações realizadas, fala qual Model e qual View utilizar, para que a ação seja completada.

Modelo MVC

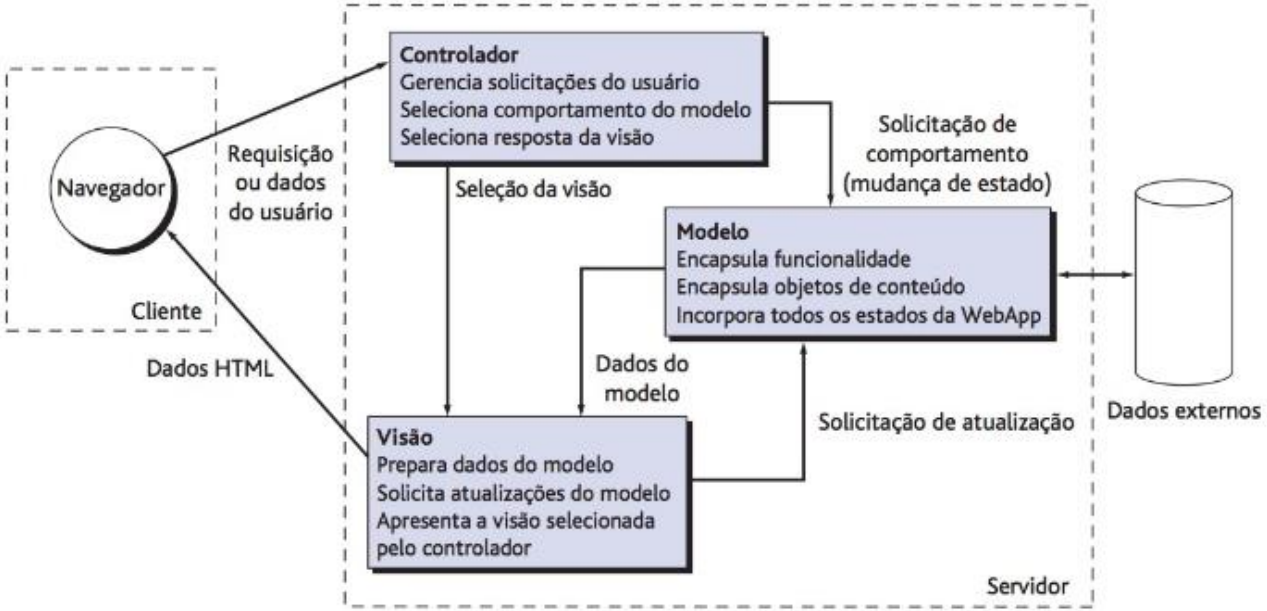


FIGURA 17.8 A arquitetura MVC.
Fonte: Adaptado de [Jac02b].

Modelo MVC

Newton
Quem se prepara, não para.

Em uma **WebApp**, a visão é atualizada pelo controlador com dados do modelo baseados nas informações fornecidas pelos usuários.

As solicitações ou os dados do usuário são manipulados pelo controlador. O controlador também seleciona o objeto visão aplicável, de acordo com a solicitação do usuário. O objeto-modelo pode acessar dados armazenados em um banco de dados (repositório de dados local ou um conjunto de arquivos independentes). Os dados trabalhados no modelo devem ser formatados e organizados pelo objeto de visão apropriado e transmitidos do **servidor de aplicações** de volta para o navegador instalado no cliente para exibição na máquina do usuário.

Modelo MVC

Newton
Quem se prepara, não para.

O diálogo das camadas

View: Fala Controller! O usuário acabou de pedir para acessar o Facebook! Pega os dados de login dele ai. **Controller**: Blz. Já te mando a resposta. Ai model, meu parceiro, toma esses dados de login e verifica se ele loga. **Model**: Os dados são válidos. Mandando a resposta de login. **Controller**: Blz. View, o usuário informou os dados corretos. Vou mandar pra vc os dados dele e você carrega a página de perfil. **View**: Vlw. Mostrando ao usuário...

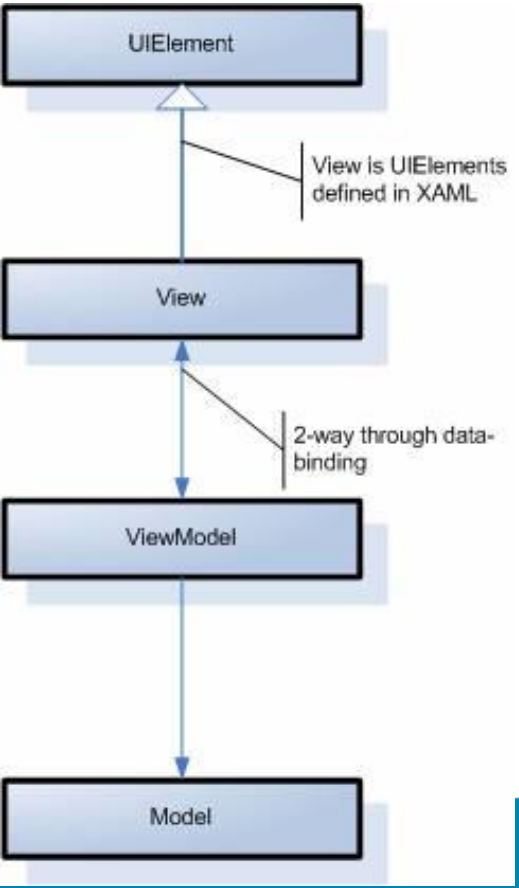
Modelo MVVM

Newton
Quem se prepara, não para.

- O padrão de projeto **Model-View-ViewModel (MVVM)** foi originalmente criado para aplicativos **Windows Presentation Foundation** (WPF) usando **XAML** para separar a interface do usuário (UI) da lógica de negócios e aproveitando ao máximo o data binding (*a vinculação de dados*).
- Aplicações arquitetadas desta forma têm uma camada **ViewModel** distinta que não possui dependências de sua interface de usuário.
- Como as classes **ViewModel** de um aplicativo não têm dependências sobre a camada de interface do usuário, você pode facilmente trocar uma interface de usuário iOS por uma interface Android e escrever testes contra a camada ViewModel.

http://www.macoratti.net/16/09/net_mvvm1.htm

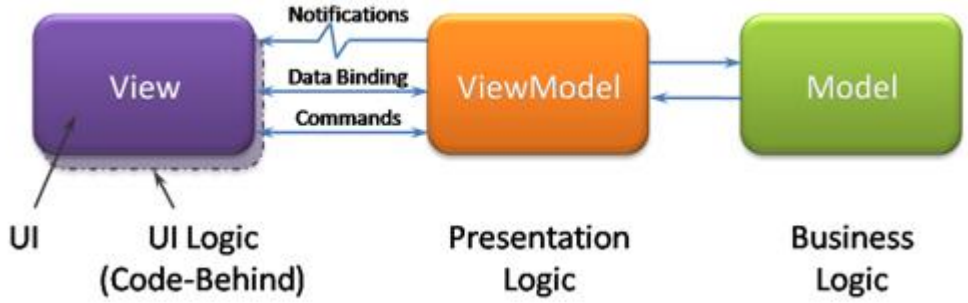
Modelo MVVM



- A camada **Model (Modelo)** não conhece a **View (Camada de apresentação)** e vice-versa, na verdade a View conhece a ViewModel e se comunica com ela através do mecanismo de *binding*.
- A View conversa com a ViewModel, que conversa com o Model, ou seja, a view não conhece o model, que não conhece a view nem a viewmodel, totalmente desacoplado e de fácil manutenção.

https://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411

Modelo MVVM



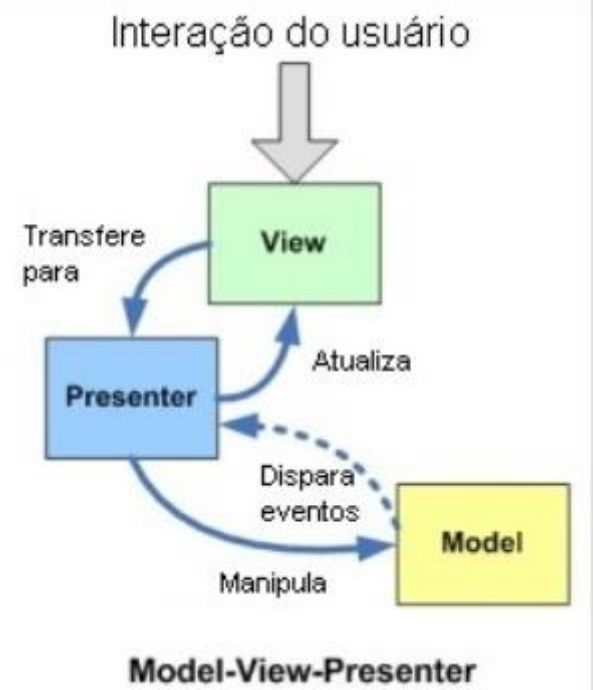
- A ViewModel é uma classe não visual, que expões para a View uma lógica de apresentação.
- A ViewModel é testável, independentemente da View ou Model.
- A ViewModel coordena as intenções entre a View e o Model.
- A ViewModel não referencia a View, na verdade não tem nenhum conhecimento sobre a mesma.

- A View é um elemento visual, como um objeto Window, Page, UserControl ou DataTemplate.
- A View referencia a ViewModel através da propriedade **DataContext**. Os controles da View são preenchidos com propriedades ou comando, expostos pela ViewModel.

- O Modelo são classes que encapsulam a lógica de negócios e os dados.
- O Modelo não referencia diretamente a View ou ViewModel.
- O Modelo provê eventos de notificação de mudança de estado, através das interfaces **INotifyPropertyChanged** and **INotifyCollectionChanged.** Isto facilita o preenchimento de dados na View.

Exemplo de Arquitetura MVVM
https://github.com/ranzate/arquitetura-mvvm-swift

Modelo MVP



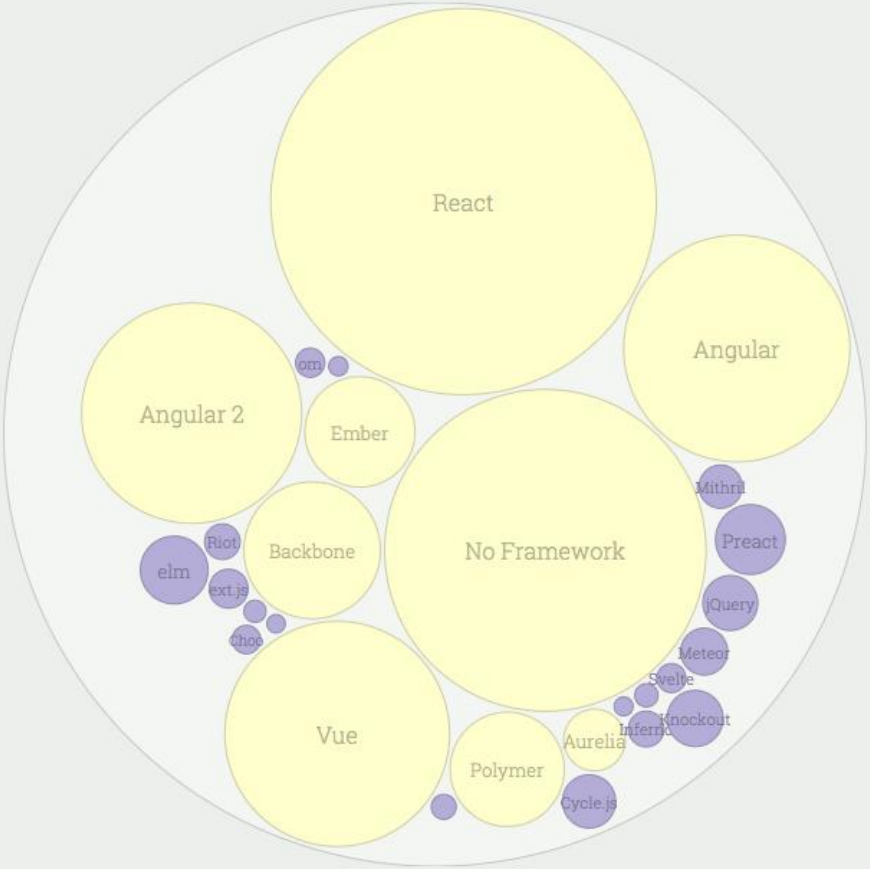
Model-View-Presenter

- **MVP (Model View Presenter)**

Em MVP a camada *Presenter* assume a função de mediadora (executada pelo *Controller* em MVC). Além disso, a *View* é responsável por manipular os eventos UI (como mouseDown, keyDown, etc.), que era o trabalho da Controller. Finalmente, a *Model* se torna estritamente um modelo de domínio.

https://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411
https://pt.slideshare.net/Celio12/padres-arquiteturais-mvc-mvp-e-mvvm

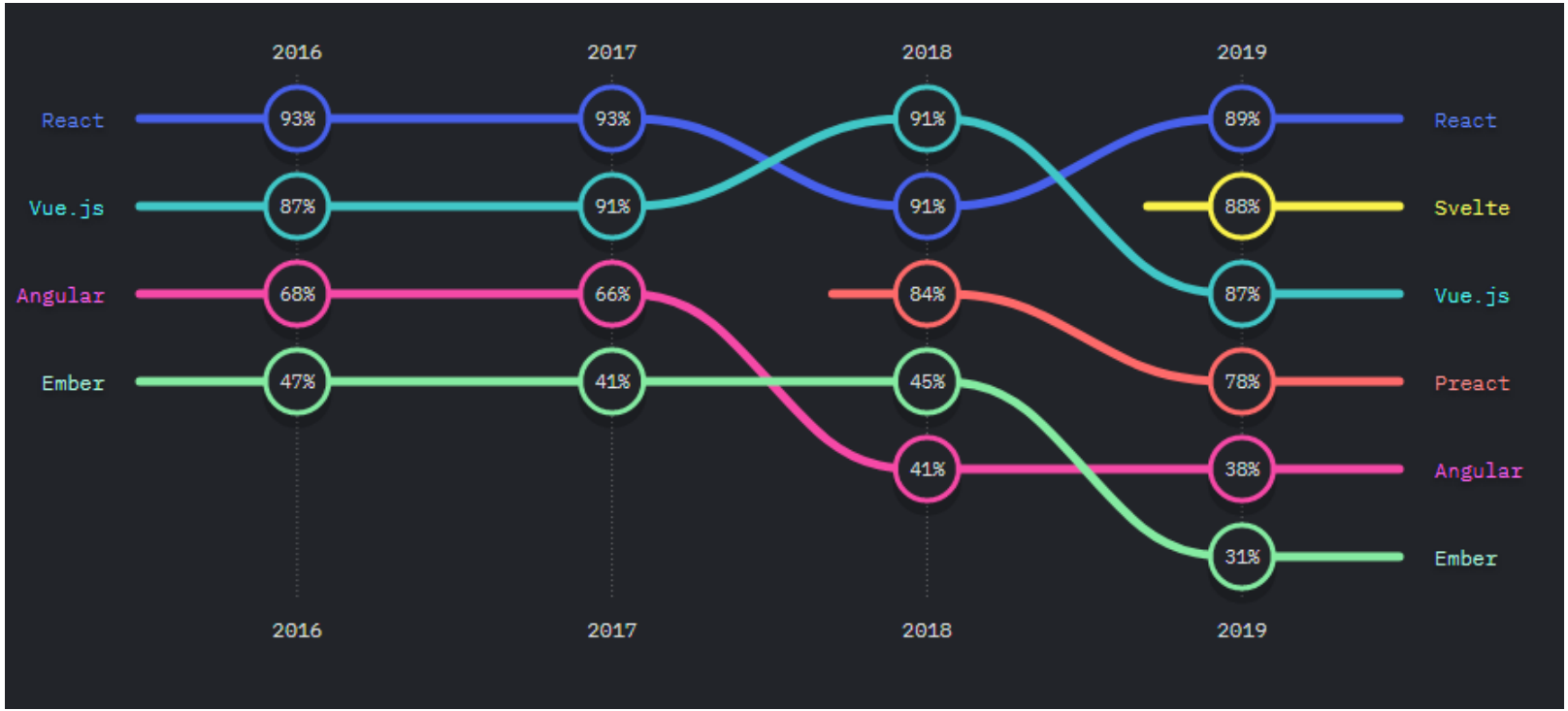
Escolhas de Tecnologias para MVC



Fonte:
https://medium.com/@marco_s_mendes/tecnologias-web-em-2018-o-que-escolher-ao-come%C3%A7ar-um-novo-projeto-f396a1d76859

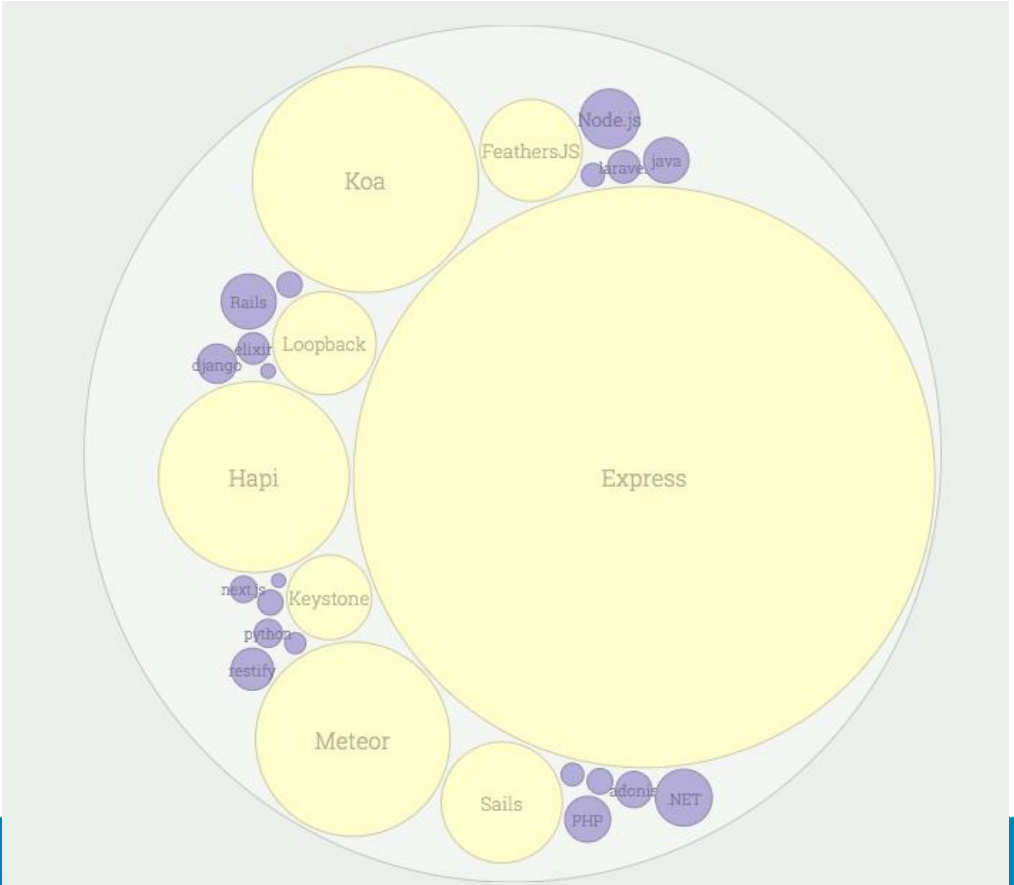
Escolhas de Tecnologias para MVC

Newton
Quem se prepara, não para.



https://2019.stateofjs.com/front-end-frameworks/

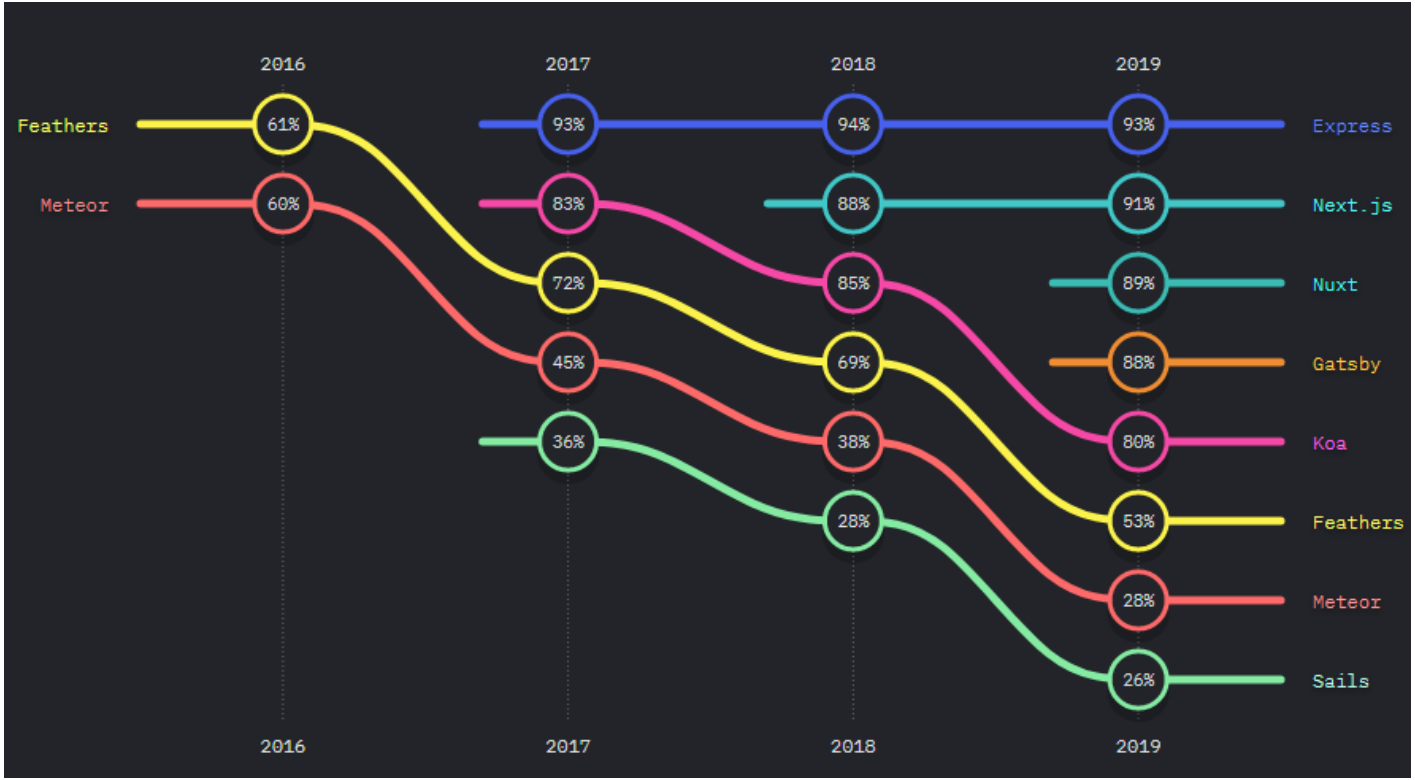
Escolhas de Tecnologias para MVC



Fonte:
https://medium.com/@marco_s_mendes/tecnologias-web-em-2018-o-que-escolher-ao-come%C3%A7ar-um-novo-projeto-f396a1d76859

Escolhas de Tecnologias para MVC

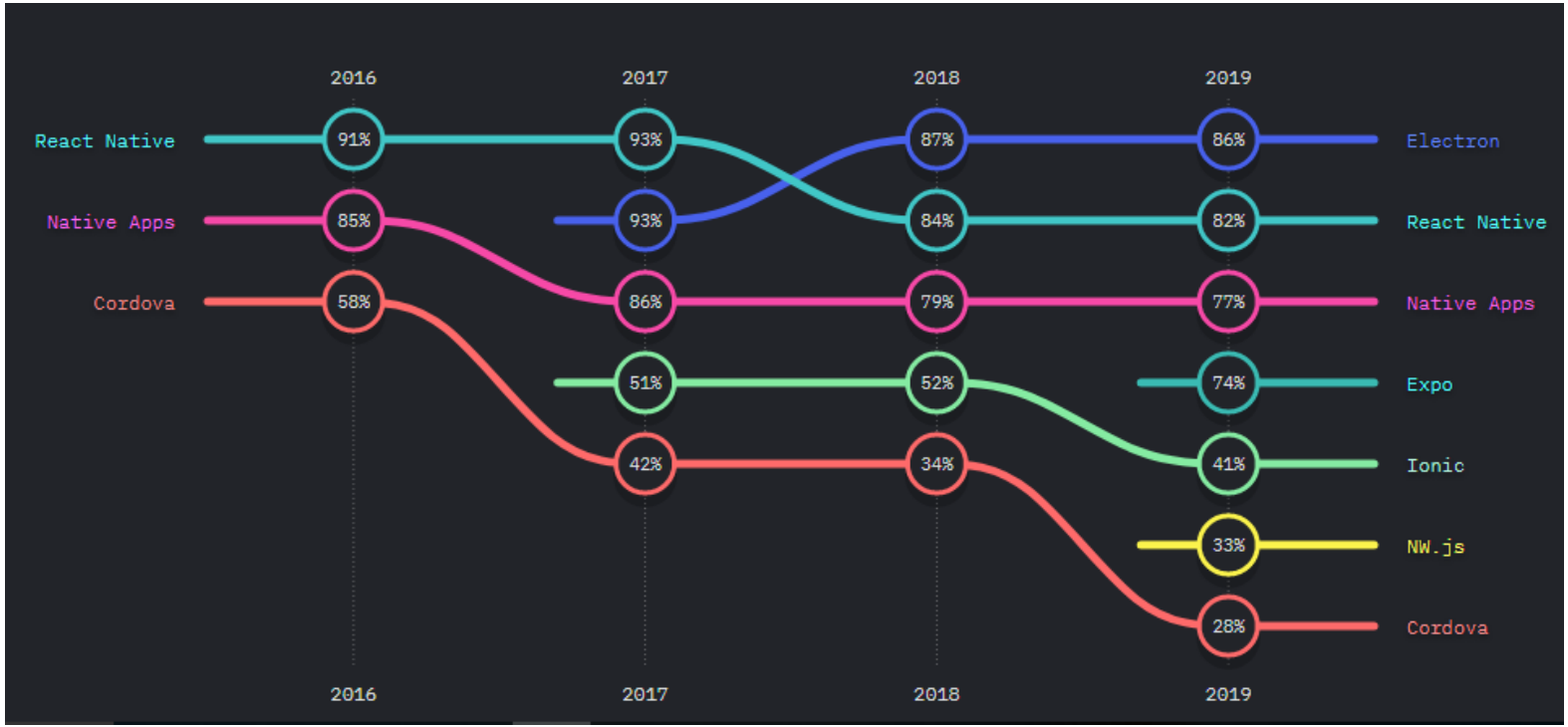
Newton
Quem se prepara, não para.



https://2019.stateofjs.com/back-end/

Escolhas de Tecnologias para MVC

Newton
Quem se prepara, não para.



https://2019.stateofjs.com/mobile-desktop/

Node.js

Newton
Quem se prepara, não para.

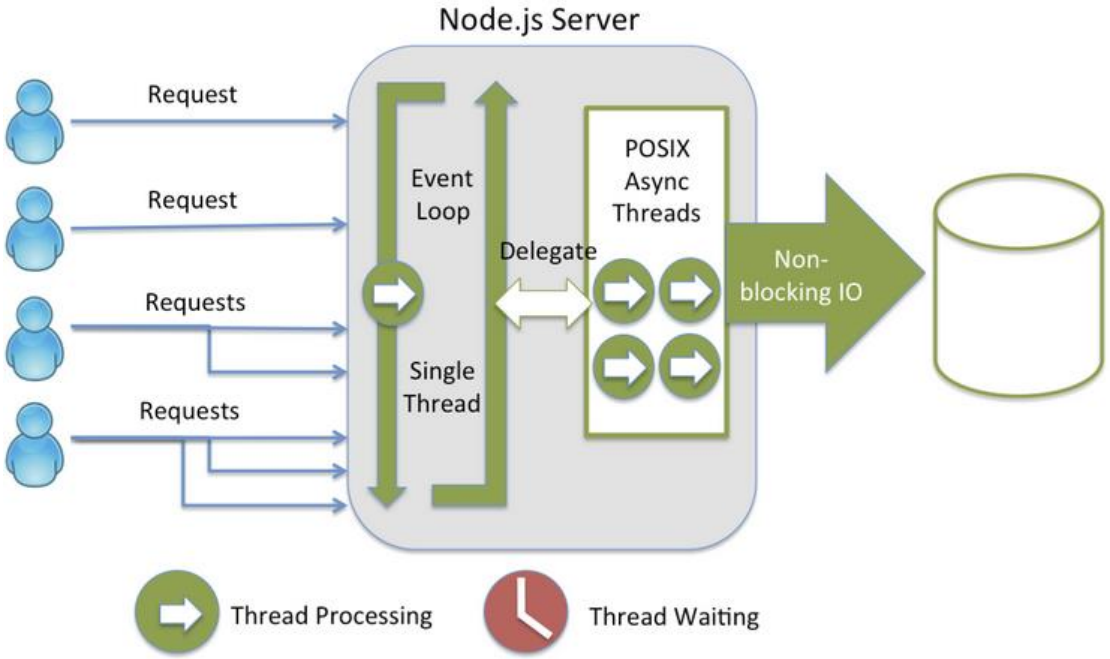
- O Node.js é um ***runtime environment*** de código aberto executado na **engine V8 do Chrome usando JavaScript**, com alto potencial de escalabilidade sem estar atrelado a plataformas ou dispositivos.
- Tecnologia usada para desenvolver aplicativos *server-side (back-end).*

Node JS

Newton
Quem se prepara, não para.

- Leveza e flexibilidade fazem do Node.JS uma tecnologia indicada para a implementação de serviços e componentes de arquiteturas como a de **microsserviços e serverless**.
- Além disso, conta com suporte das principais empresas de produtos e serviços Cloud do mercado, como a **AWS**, **Google Cloud** e **Microsoft Azure.**
- **Indicações:**
  - Aplicações em Tempo Real (ex. chat)
  - Ambientes Escaláveis (grande numero de conexões concorrentes)
  - Camada de Entrada do Servidor : O Node.js faz pouco processamento de dados e apenas passa a requisição para frente, se comunicando com serviços de *backend*.
  - API com NoSQL.

Node.js



- cada requisição é processada de uma só vez (Event Loop), que delega para o Async Thread o processo de pull de threads (ex. 4 threads alocadas)

https://www.youtube.com/watch?v=KtDwdoxQL4A

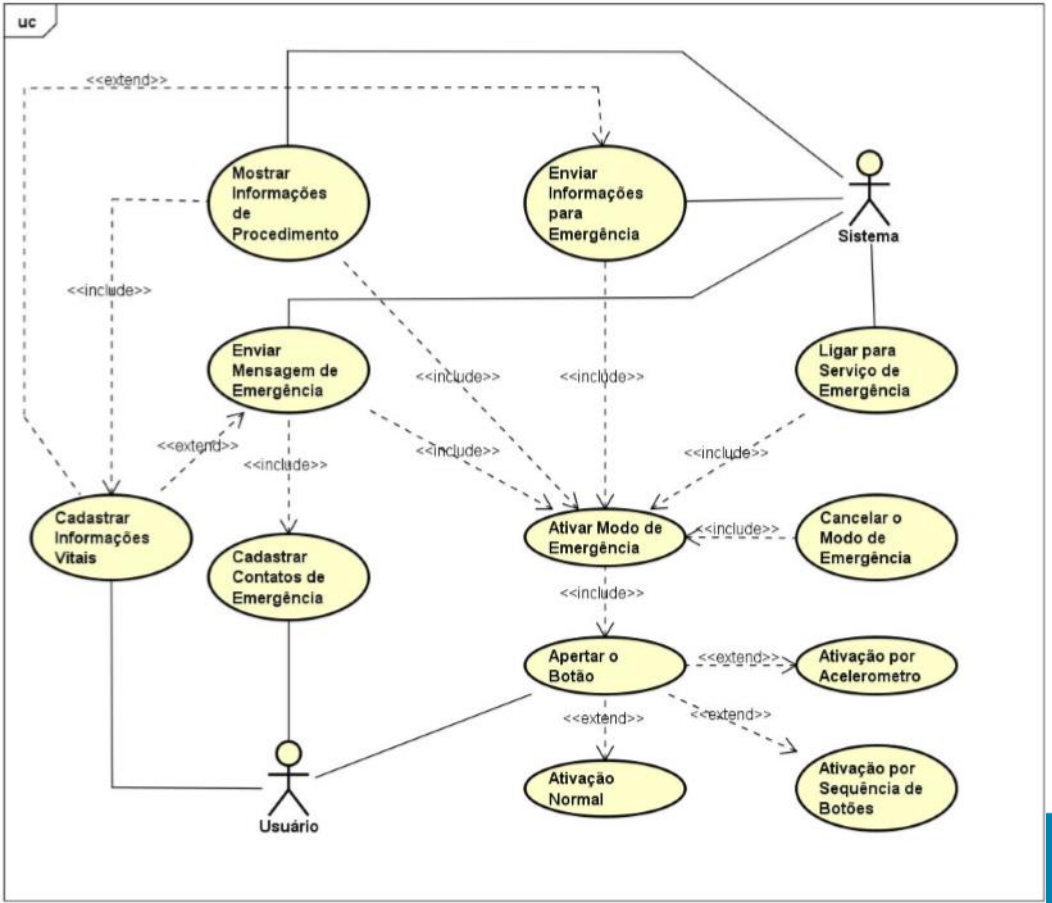
Node com MVC

Newton
Quem se prepara, não para.

Exemplo de Documentação de Arquitetura

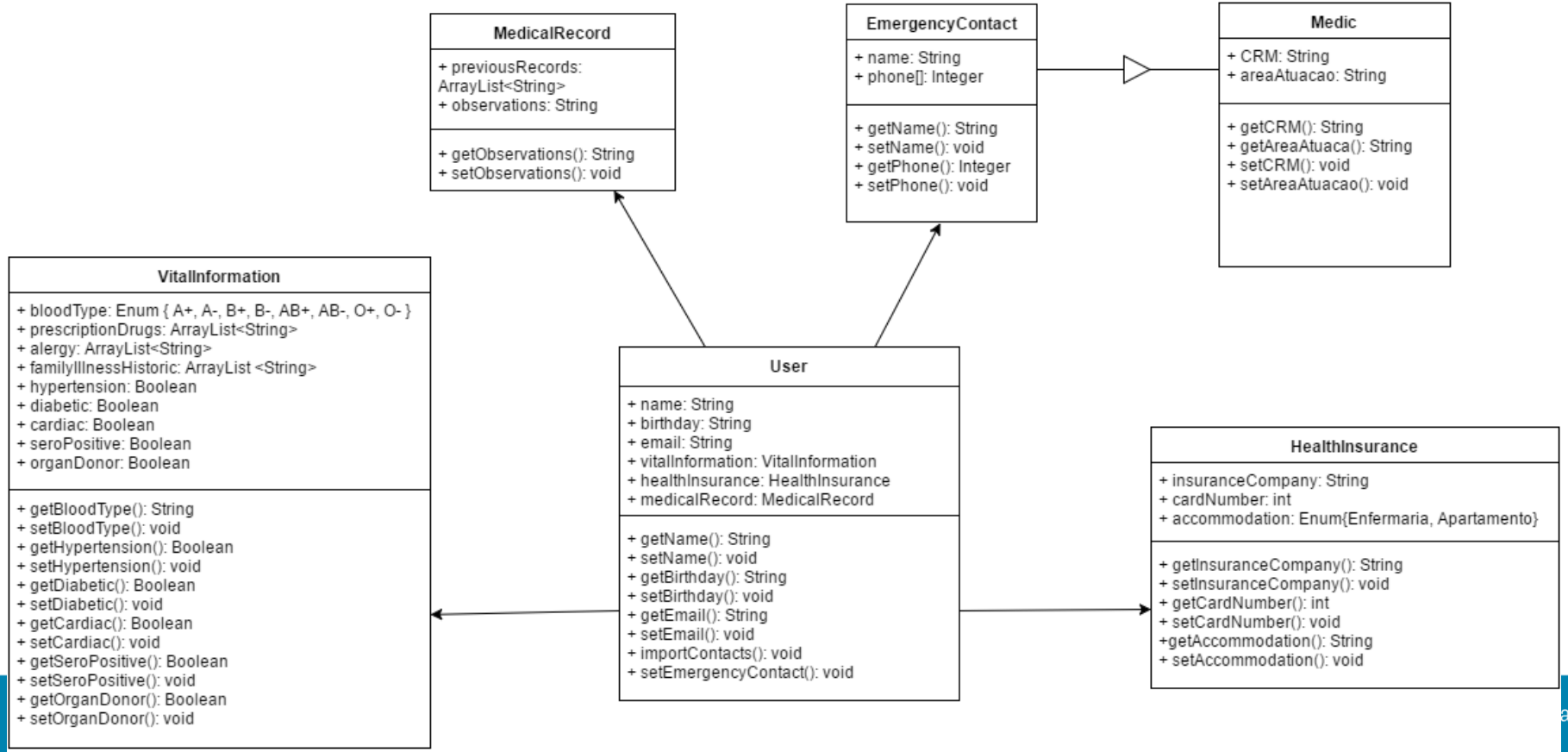
- https://github.com/LeonardoKalyn/Red-Button/wiki/Documento-de-Arquitetura-de-Software-(DAS)

Node com MVC


uc
<<extend>>
Mostrar Informações de Procedimento
Enviar Informações para Emergência
Sistema
<<include>>
Enviar Mensagem de Emergência
<<include>>
<<include>>
Ligar para Serviço de Emergência
<<extend>>
<<include>>
<<include>>
<<include>>
Cadastrar Informações Vitais
Cadastrar Contatos de Emergência
Ativar Modo de Emergência
<<include>>
Cancelar o Modo de Emergência
<<include>>
Apertar o Botão
<<extend>>
Ativação por Acelerometro
<<extend>>
<<extend>>
Ativação Normal
Ativação por Sequência de Botões
Usuário

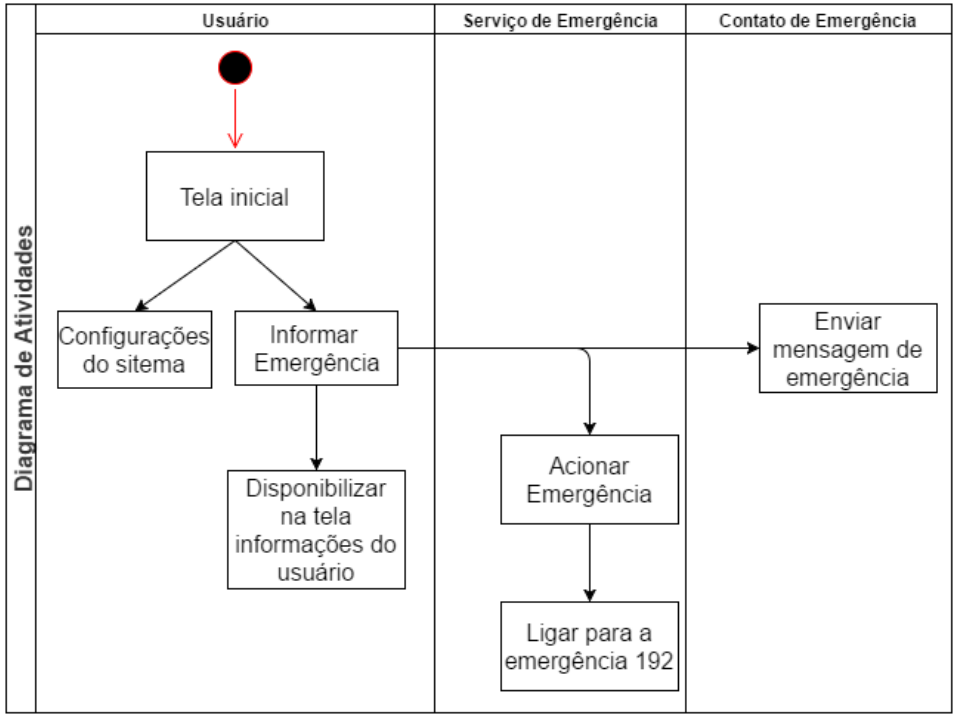
Node com MVC

Newton
Quem se prepara, não para.



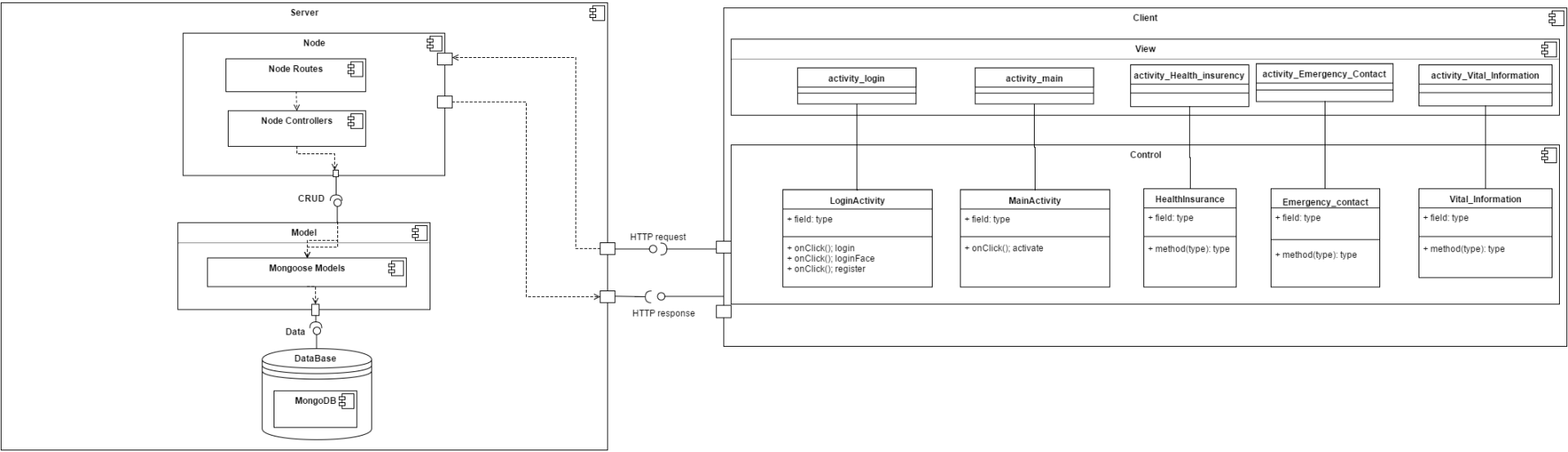
Node com MVC

Newton
Quem se prepara, não para.



Node com MVC

Newton
Quem se prepara, não para.



Criando o primeiro projeto com Node

Newton
Quem se prepara, não para.

Executar o passo a passo do projeto 1 com node:

https://github.com/ingogbe/nodeExample/wiki/1.-Criando-o-projeto-b%C3%A1sico-com-NodeJS

Preparando o Ambiente

Newton
Quem se prepara, não para.

Instalação do Github desktop e do VS studio ou Atom

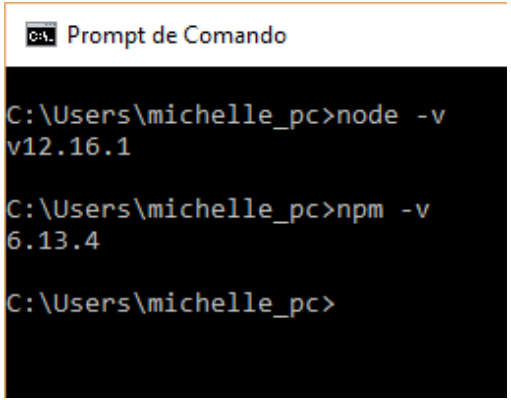
- https://desktop.github.com/
- https://code.visualstudio.com/docs/?dv=win
- https://atom.io/

Instalando o Node JS e o NPM

Newton
Quem se prepara, não para.

https://nodejs.org/en/

Testar se foi instalado corretamente



Instalando o React Native

Newton
Quem se prepara, não para.

Instalação do React Native

>npm install -g react-native-cli

Testar se está instalado

>react-native -h

Instalando o Android Studio

Newton
Quem se prepara, não para.

https://developer.android.com/studio/preview/index.html?hl=pt-br

Verificar qual a pasta onde está o SDK

Criar a Variável de ambiente: Propriedades do sistema – Configurações->Variáveis de ambiente e de sistema:

Nome: Android_home

Valor: url do SDK do android studio

Emulador Android

Newton
Quem se prepara, não para.

Genymotion:
https://www.genymotion.com/fun-zone/
Bluestacks:
https://www.bluestacks.com/pt-br/index.html

***Configurar Android SDK para rodar com o emulador**