CAIO INCAU

Coordenador de Engenharia na Wirecard

@iCaioIncau

@caioincau

wirecard

A EXPERIÊNCIA DA WIRECARD

ARQUITETURA DE UM TIME FRONT-END DE ALTA PERFORMANCE

VAMOS INVESTIR EM USABILIDADE E CRIAR UM SISTEMA NOVO.

Diretoria da Wirecard

SISTEMA ANTERIOR

- Ruby on Rails
- JQuery (pelo menos três versões)
- Bootstrap
- Super acoplado e pouca coisa era reusável

O QUE ESTÁ APLICAÇÃO IRIA PRECISAR?

- Manipular dados sensíveis
- Alta escalabilidade
- Suporte cross browser
- Tempo curto para um projeto grande
- Fácil de reusar e incrementar os componentes
- Doze equipes de back-end gerando insumo para o projeto

ESTRUTURA DA TALK

- Processos e práticas da equipe enxuta
- Arquitetura Front-End simples e escalável
- Deploy, build e otimizações

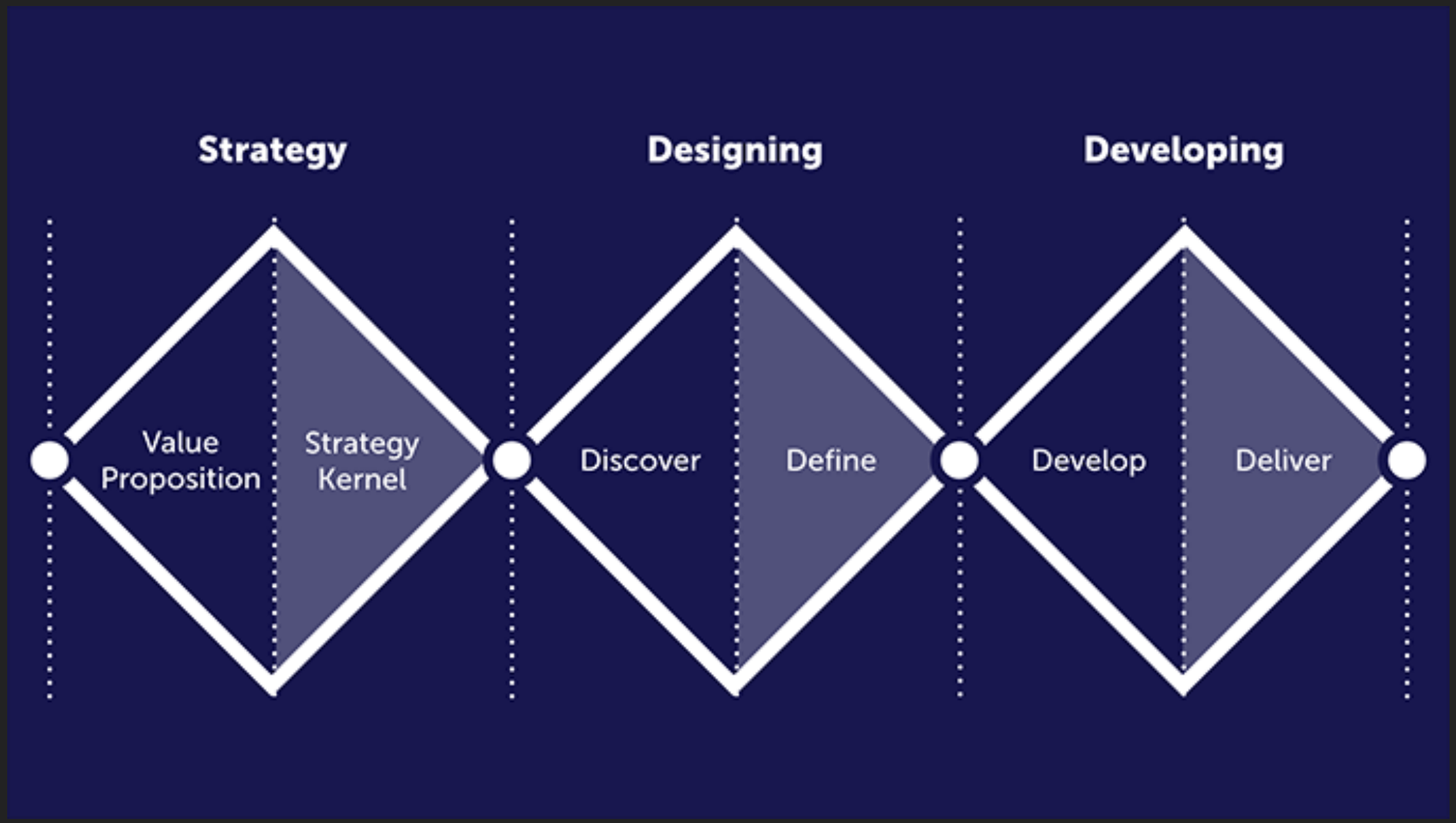
ESTRUTURA DA EQUIPE

- 2 UX Designers
- 2 UI Designers
- 4 Front-ends
- 1 QA
- 1 PO

FOCO NO PROCESSO

TRANSPARÊNCIA DO WIP

TRIPLE DIAMOND PROCESS



QA É UM PROCESSO, NÃO UM PAPEL

"SIMPLICIDADE É O ÚLTIMO GRAU DE SOFISTICAÇÃO".

Leonardo da Vinci

DADA A PREMISSA DE MANTER A STACK SIMPLES, QUAL FRAMEWORK FAZ MAIS SENTIDO PARA NÓS?

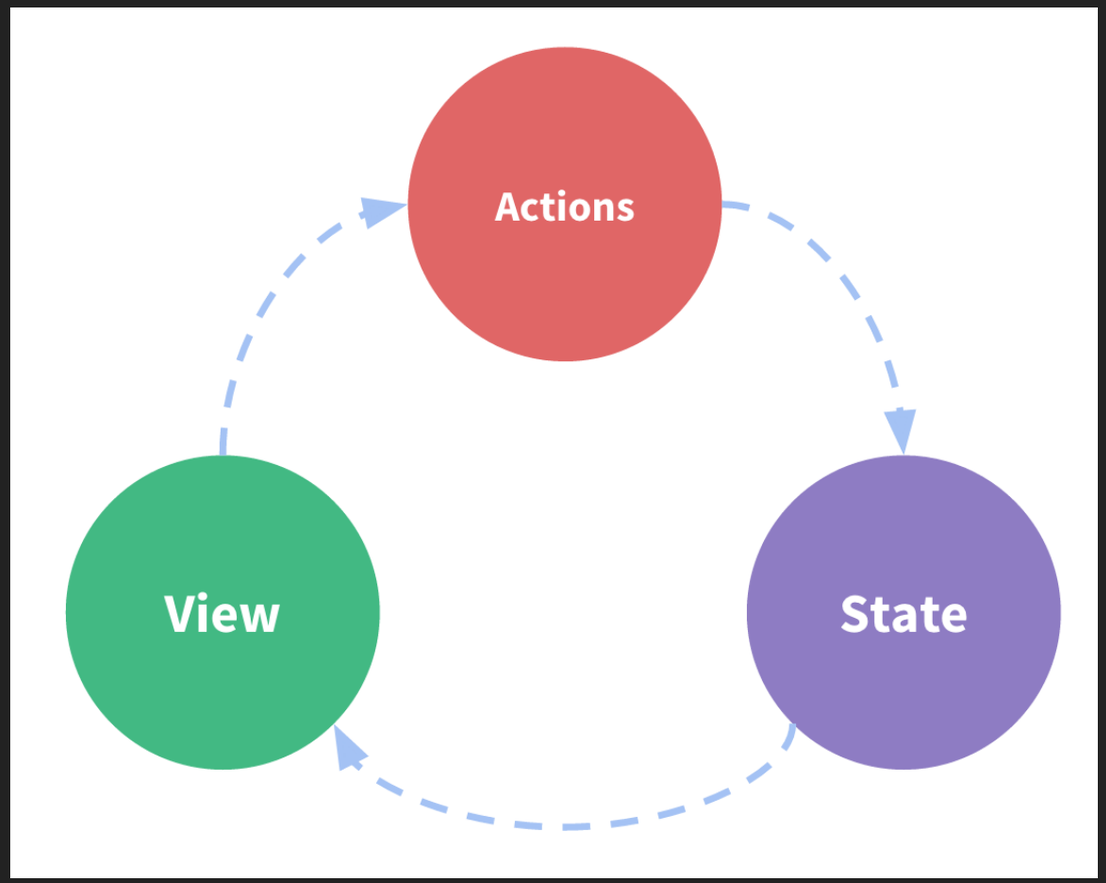
VUE.JS

- Suporta IE9
- Curva de aprendizado pequena
- Melhor performance
- Boa documentação
- Framework progressivo
- Open Source
- Comunidade super ativa

APESAR DA SIMPLICIDADE DO FRAMEWORK, COMEÇOU A FICAR COMPLICADO LIDAR COM DADOS NA APLICAÇÃO...

VUEX

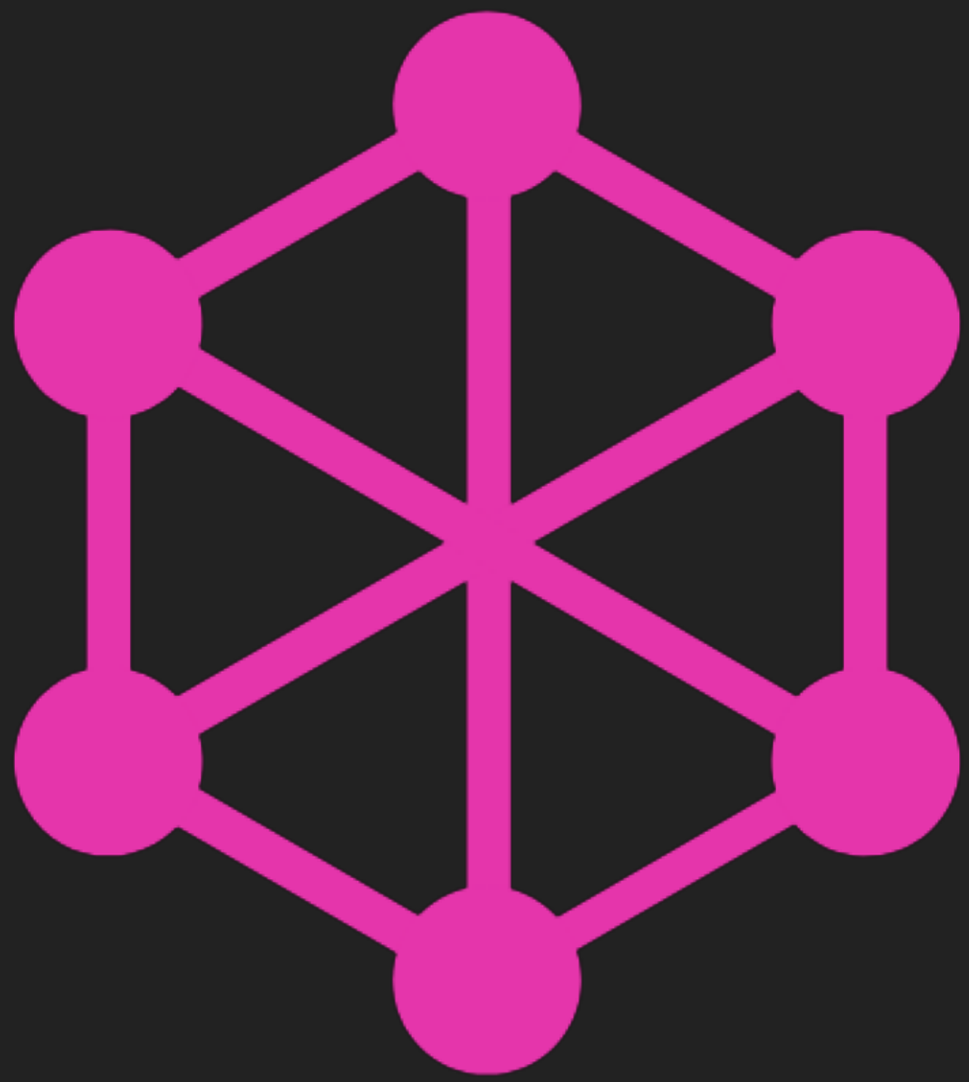
- Implementação baseada em Flux
- Zero config
- Todos os dados da aplicação centralizados em um lugar
- Fácil acesso a dados, cache e time-travel



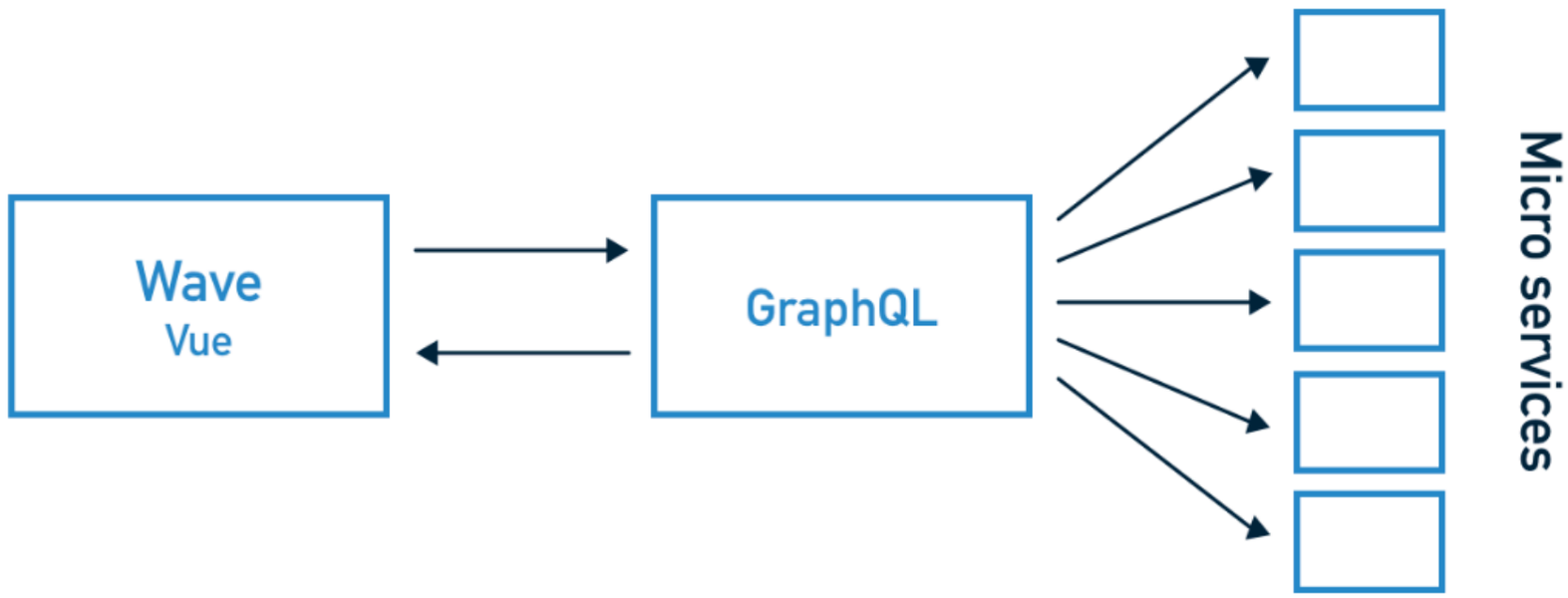
AS RESPOSTAS DA APIS NÃO ERAM CONCISAS...

GRAPHQL

- Mais de 30 APIs, precisávamos padronizar
- Mais uma camada de cache
- Redução do tráfego de dados, podíamos pegar somente o necessário
- Melhor controle sobre as respostas
- Mais resiliente



GRAPHQL



GRAPHQL É RESILIENTE "ATÉ DEMAIS". PRECISAMOS CAPTURAR OS ERROS E ATUAR SOBRE ELES...

SENTRY

- Monitoramento de erros real time
- Agimos antes dos usuários reportarem os erros
- Maior facilidade de debugar
- Seis linhas de código capturam os erros da aplicação toda.

CÓDIGO SENTRY

```
Raven.config(`https://${KEY}@sentry.io/${APP}`)
  .addPlugin(RavenVue, Vue)
  .install();

Vue.config.errorHandler = (err, vm, info) => {
  Raven.captureException(err)
}
```

PRECISAMOS DE TESTES PARA GARANTIR QUE ESTES ERROS NÃO VOLTEM ACONTECER, ALÉM DE PREVINIR NOVOS...

JEST E CYPRESS

- Jest para testes unitários, integração fácil, pouca configuração
- Cypress para testes exploratórios, testes feitos em JS pela própria equipe de dev
- Qualidade como um processo, QAs escrevem critérios de aceite e ajudam no roteiro de testes, mas qualidade é obrigação de todos

BUILD

WEBPACK

- Build customizável
- Otimizações nativas out of the box
- Diferentes builds por ambiente
- Code-split fácil
- Remove código inutilizado
- Build com suporte a browsers mais antigos(Babel)

DOCKER

- Maior facilidade de desenvolvimento
- Podemos pegar exatamente a imagem que está em produção
- Facilita o deploy do GraphQL, dado que usamos o ECS e podemos escalar com facilidade

docker

CI E DEPLOY

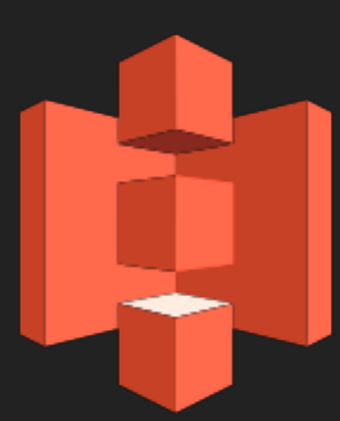
CIRCLE CI

- Fácil de configurar
- Podemos aproveitar a imagem docker que já criamos
- Garante os testes
- Usamos CD, merge na master, deploy para ambiente de integração, adicionou tag, vai para produção

PARA ONDE VÃO OS ASSETS GERADOS ?

AMAZON S3

- Fácil escalar
- Barato
- Simples de configurar, quase um copy e paste de arquivos
- 99,999999999% (11 9s) de disponibilidade



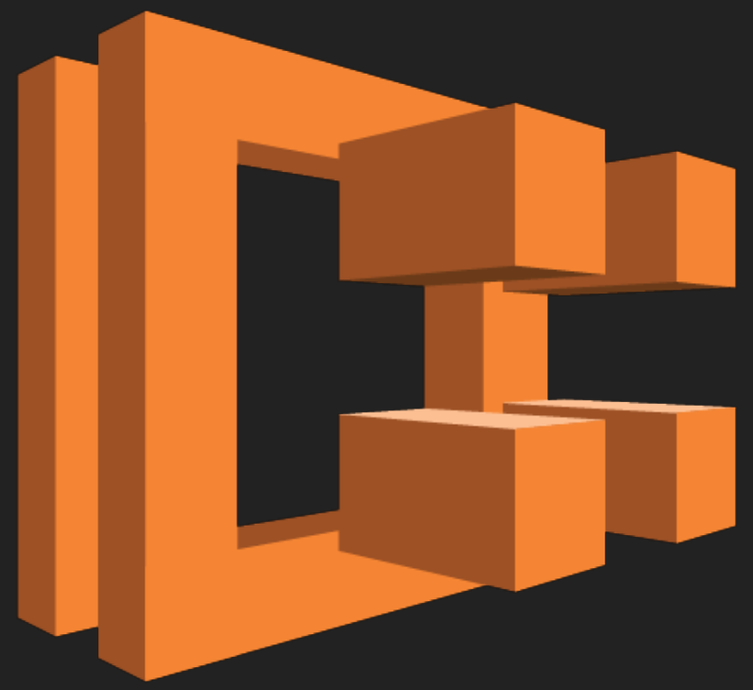
AMAZON CLOUDFRONT

- Baixa Latência
- Alta velocidade de transferência
- Configuração em poucos clicks pelo painel da Amazon

E PARA ONDE VAI O GRAPHQL?

AMAZON ECS

- Altamente escalável
- Alta performance
- Deploy integrado no CI através de command lines
- Monitoria grátis e clara
- Se recupera sozinho de falhas



AWS ECS

DÚVIDAS?

OBRIGADO