SUPER APP

Estrutura do trabalho

Etapas do trabalho

Método	Super App VR	Arquitetura proposta	Plataforma de desenvolvimento	Alertas e recomendações	Próximos passos	Detalhamento da arquitetura
Visão geral do método de trabalho utilizado para avaliar o Super APP.	Visão da VR sobre seu Super APP, levando em consideração as necessidades estratégicas e principais expectativas.	Desenho da proposta de uma nova arquitetura para suportar as demandas e necessidades de negócio sobre o Super App VR.	Descrição objetiva de recomendações necessárias para nova arquitetura tendo em vista os pontos e gaps observados.	Embasamento técnico sobre os caminhos arquiteturais utilizados na proposição.	Mapeamento do cenário tecnológico e desenho da arquitetura atuais da VR.	Amarração final, sintetizando-se as principais percepções de forças e pontos de atenção para embasar um parecer final sobre a nova arquitetura.

Apresentação executiva

Método de trabalho

Método

Como conduzimos os trabalhos

PROCESSO

COMPREENSÃO

- **VISÃO VR** — ESTRATÉGIA & TECNOLOGIA
- **IMERSÃO** — NO CENÁRIO ATUAL DE ARQUITETURA
- **REFINAMENTO** — DO CENÁRIO ATUAL TECNOLÓGICA DA VR

ANÁLISE

PROPOSTA DE ARQUITETURA

FERRAMENTAS

- INCEPTION
- ENTREVISTAS
- AVALIAÇÃO DE JORNADAS
- ANÁLISE DE DOCUMENTAÇÃO
- INSPEÇÕES TÉCNICAS
- PONTOS DE CONTATO VR
- OBSERVAÇÃO DE CANAIS
- ARQUITETURAS E MODELOS DE REFERÊNCIAS

- Funcionalidades
- Arquitetura tecnológica
- Implementação
- Infraestrutura
- Segurança
- Parceiros

PERFIS

- ANDRÉ ALVES(CONSULTOR SENIOR)
- ANDRE XAVIER(CONSULTOR DEVOPS)
- ALEX GAMAS (ARQUITETO SOLUÇÕES)
- ANDERSON GAMA(ARQUITETO SOLUÇÕES)
- EDMILSON SANTANA (CONSULTOR SENIOR)
- JOSE INACIO FERRARINI (ESPECIALISTA MOBILE)
- THALES SANTOS (ESPECIALISTA MOBILE)

DURAÇÃO: aprox. 5 semanas

Super App VR

- Sonho da VR
- Diretrizes de negócio
- Arquitetura tecnológica
- Objetivos arquiteturais
- Requisitos técnicos
- Jornadas iniciais

Carteira Digital do Trabalhador: SONHO

Para Quem	Ao trabalhador brasileiro e sua família
O que	Faremos com que seu dinheiro valha mais
Como	Com o app de pagamentos, benefícios e serviços mais usado no seu dia a dia
Por que	Por possuirmos a melhor gestão de experiência e dados
Quanto	Impactando ao menos 5 milhões de lares
Quando	Em até 3 anos

Diretrizes de negócio

Nosso entendimento sobre ambições para o Super App da VR

- **Relacionamento sólido e conexão recorrente com trabalhadores** e suas famílias.
- **Funcionalmente simples,** que traga serviços, benefícios, utilidades, facilidades e agilidade.
- **Time to market**, disponibilizar novidades rapidamente.

Arquitetura tecnológica

Principais dimensões

mindset | Experimentação | Jornadas | Incorporar parceiros | Marketing Digital | Time to Market

Arranjo arquitetural plugável

Permitir a conexão controlada da plataforma com produtos e serviços de terceiros ou internos.

Implementação ágil

Viabilizar a **escalabilidade dos times** de desenvolvimento e consequentemente a agilidade para a construção.

Orientação a dados

Conhecer o comportamento e preferências dos trabalhadores e proporcionar maior engajamento em sua jornada.

Operação tecnológica simplificada

Viabilizar a **assertividade das operações** de sustentação do dia a dia.

Objetivos arquiteturais

Sustentadores da estratégia

CX fluida e atrativa

Entregar uma **experiência transparente entre os diversos produtos e serviços disponíveis no Super App**, com o refinamento constante da experiência do cliente. A arquitetura deve viabilizar a utilização de um Design System para UI e UX com componentes que **facilitem as jornadas e reduzam fricções** na experiência.

Integração de parceiros

Permitir conexão a outros atores para execução de operações e incorporação de produtos e serviços de terceiros à plataforma.

Segurança

Proteger dados, privacidade e transações dos trabalhadores, bem como recursos e a estrutura tecnológica.

Facilidades de desenvolvimento

Alavancar a produtividade do time com base em seu skill atual. Priorizaremos características da plataforma de desenvolvimento que suportem os objetivos com simplicidade e frugalidade arquitetural.

Estratégia de dados

Retroalimentar o negócio com conhecimento sobre sua base de clientes, fortalecendo e dinamizando sua relação com trabalhadores e demais atores do ecossistema.

Adoção de Mini Apps

Incorporar parceiros mais facilmente, oferecendo facilidades e agilidade para que o os atores do ecossistema se acoplem ao Super App. Neste sentido pensamos na estratégia de incorporar mini apps através de **webview** pois esta é uma tendência de mercado para a abertura do ecossistema.

Estabilidade do produto

Priorizar a **performance e reputação do Super App**. Portanto o arranjo arquitetural (dev, data, infra e ops) deverá prever a utilização de componentes e serviços considerados maduros.

Future-proof architecture

Incluir no Mindset perspectiva de que o produto deve durar para o futuro. A arquitetura deve **permitir ajustes rápidas sem degradação do arranjo e com baixa propagação de impacto**.

Requisitos técnicos

Atributos explicitamente desejados

Whitelabel

O super APP deve ser whitelabel, podendo assim ser ofertado para os parceiros emissores da VR de forma personalizada e ser facilmente redistribuído.

Integração

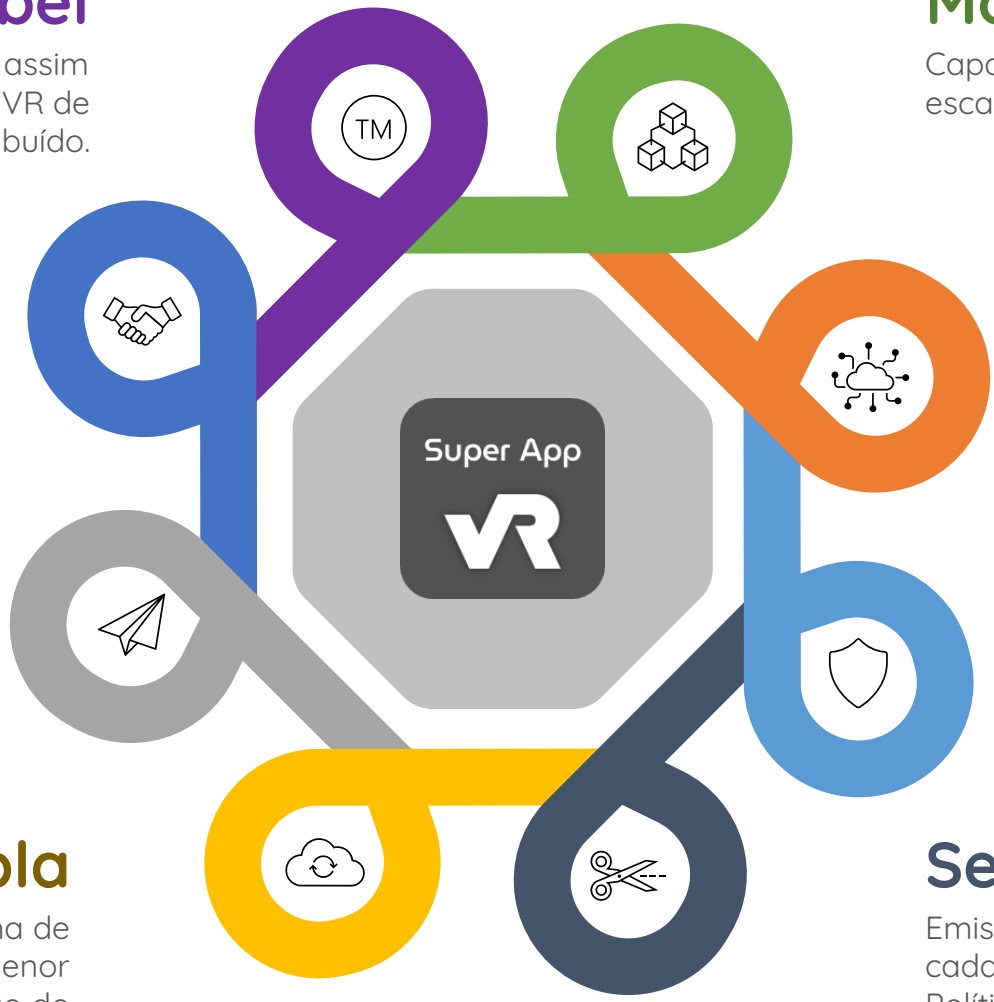
Agregar novas funcionalidades através de webview, hub de API, incorporação de outros apps e produtos da VR e de parceiros.

Push

Ter a capacidade de segregação das notificações e envio por múltiplos canais.

Distribuição ampla

O SuperApp deve ser compatível com a gama de dispositivos esperados e deverá ter o menor tamanho possível. Seu modelo de atualização do mesmo deve ser prático e ágil.



Modularização

Capacidade de ser modularizado para escalar desenvolvimento.

CI/CD específico

Integração às lojas Google Play e App Store através de um pipeline de CI/CD

Segurança

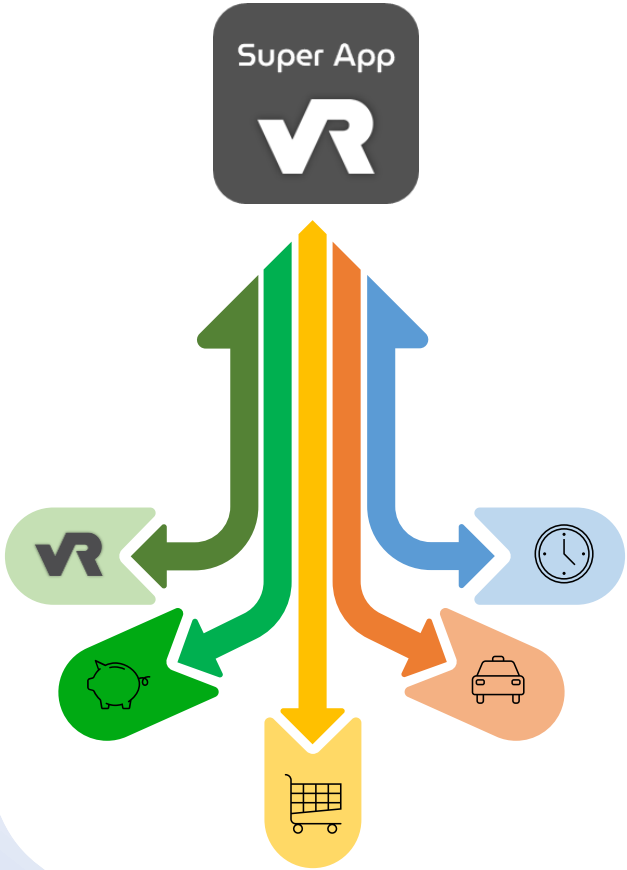
Previsão de mecanismos de segurança para as fluxos de comunicação, armazenamento local, proteção do binário e contenção de vazamentos.

Segregação

Emissores: Prover mecanismo para segregação de cadastro/login e ou integrado com emissor.
Políticas: Implementar funcionalidade de segregação de politicas, termos de uso e LGPD.

Jornadas iniciais

Ponto de partida para o projeto



Conta bandeira

Carrossel / Saldo / Busca de rede / Cartão virtual / Pagamento via QR Code.

Conta de pagamento

Serviços Financeiros / Conta Pagamento Digital / Cash-in / Pgamento de contas / PIX / TED

Elementos atuais

Marketplace

- Cashback
- Marketplace
- Fidelidade

Ponto

- Marcações de ponto
- Horas extras
- Banco de horas

Mobilidade

- Roteirização
- Deslocamento
- Vale Transporte

Elementos por vir

Arquitetura proposta

- Visão da arquitetura
- Recomendação de plataforma
- Alerta e recomendações
- Próximos passos

Visão da arquitetura

Visão da arquitetura

Elementos estruturais da arquitetura

Design system

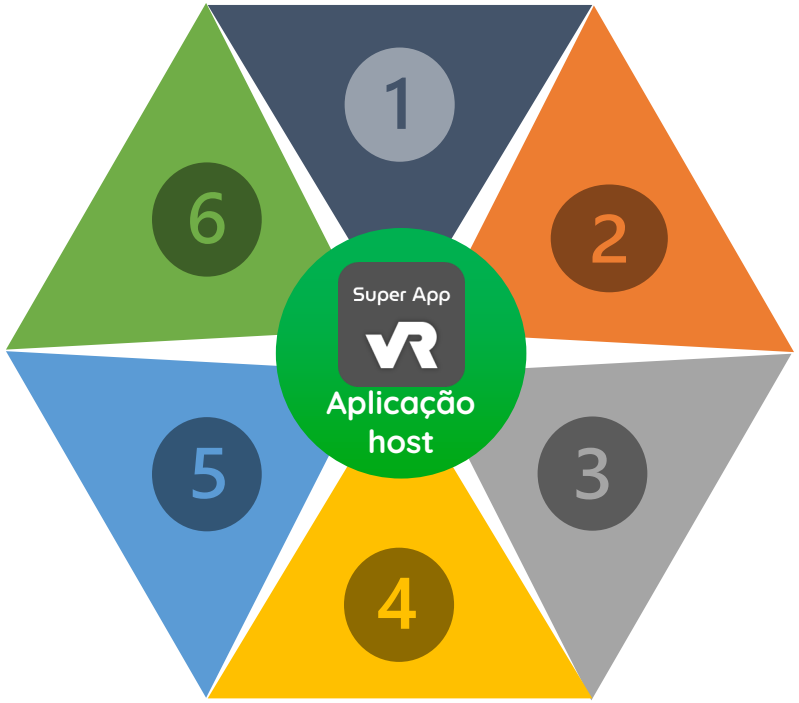
Central de componentes visuais para proporcionar a padronização e **maior fluidez na experiência do usuário**.

Central de notificações

Possibilitará a **agregação de múltiplos canais** de contato com o trabalhador e viabilizará a **segregação das notificações** para suportar a estratégia whitelabel.

CI/CD

Acelerador do processo de entrega que automatizará o fluxo de integração e **entrega continua do aplicativo diretamente na loja**.



Communication channel

Camada responsável pela **orquestração dos componentes** de integrações (módulos e webview).

Wrapper Modules

Componente que viabilizará a **escalabilidade do desenvolvimento** e gestão dos novos módulos acoplados.

SDK

Abstração que possibilita **coordenar as integrações**, além de servir como mecanismo de **controle e segurança**.

Visão da arquitetura

Convergência com as necessidades de negócio

Elementos estruturais

Camada mobile

- SDK
- Design System
- Wrapper Módulos
- Communication channel
- CI/CD do Super App
- Central de Notificações

Back-end VR

- APIs orientadas ao canal (BFF)
- Infraestrutura em cloud
- Microsserviços

Super App VR

Drivers VR

Business

- Relacionamento e conexão com trabalhadores
- Simplicidade, facilidade e agilidade
- Time to market
- Distribuição de produtos e serviços (B2B2C)

Tech

- Whitelabel
- Modularização
- CI/CD
- Segurança
- Segregação
- Distribuição ampla
- Push
- Integração

Estratégia de dados

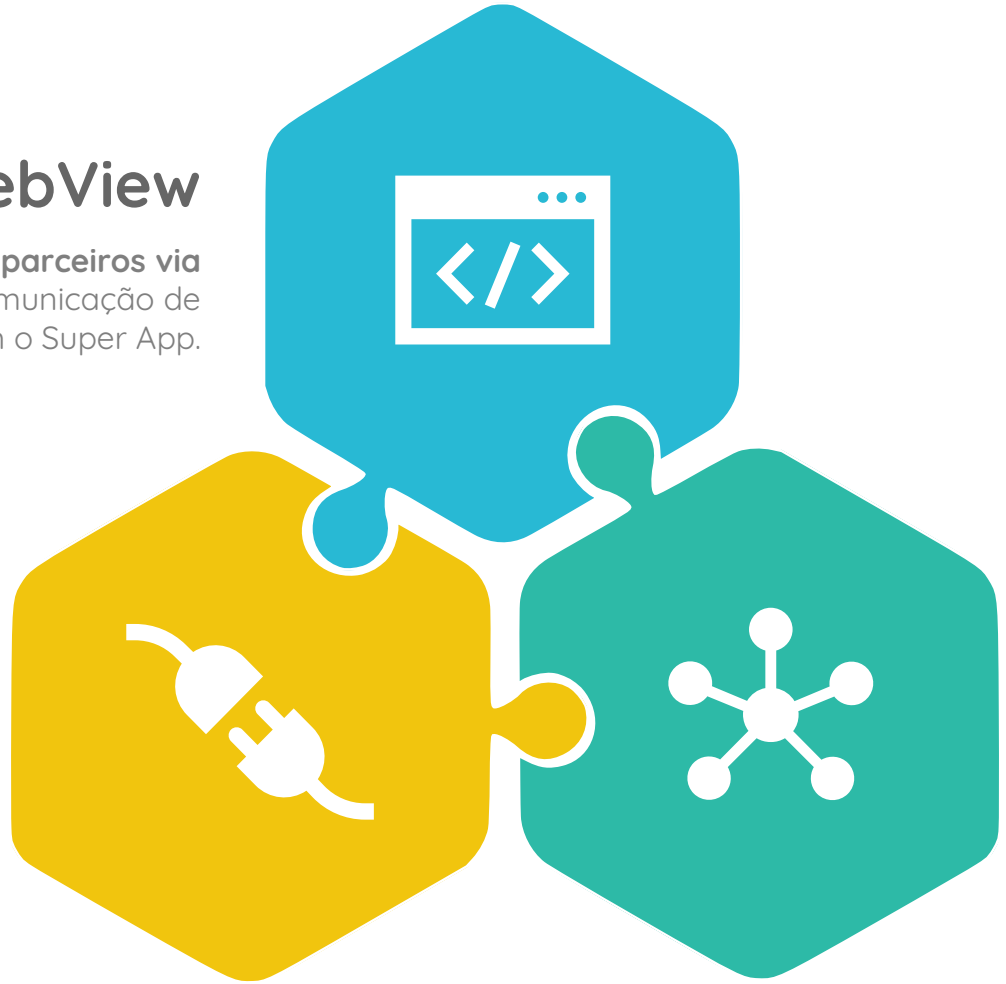
Estratégia de integração

Visão da arquitetura

Modelos de integração ao Super App

WebView

Viabilizar a **integração de parceiros via mini apps**, é possível a comunicação de portais web com o Super App.



Incorporação de App

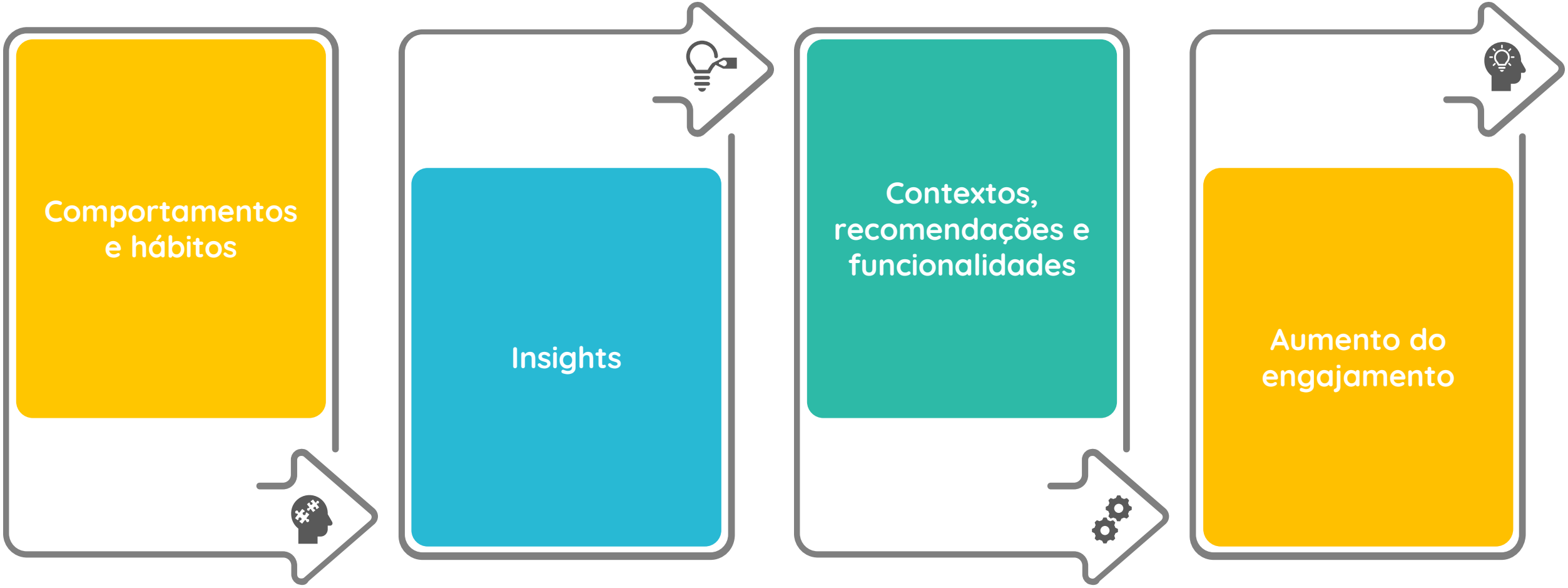
Viabilizar o **acoplamento de funcionalidades novas ou existentes** (da VR e de parceiros) que estejam na mesma plataforma tecnológica - pode ser necessário adaptá-las para integração.

Hub de API

Suporte a integração com hubs de API's (VR ou terceiros) como estratégia para **consumo ou distribuição de serviços**.

Visão da arquitetura

Estratégia de dados



Visão executiva da arquitetura

Grupos de componentes do Super App

Escopo da arquitetura mobile

Aplicação host

- White-Label
- Comnunication chanel
- SDK
- Segurança
- Captura de comportamentos e hábitos

Mini Apps

(Abordagem web)

Utilitários

(Firebase, Dynatrace, idWall, flowsense, etc)

Módulos

- Conta bandeira
- Conta pagamento
- Marketplace
- Ponto
- Mobilidade
- + produtos e serviços da VR e parceiros

Reuso e Padrões

- Design system VR
- Módulo de referência

Arquitetura de back-end atual da VR

Arquitetura BFF e API specification ↔ **Arquitetura back-end**

Dados

Ingestão de dados para métricas de negócio e técnicas

DevSecOps

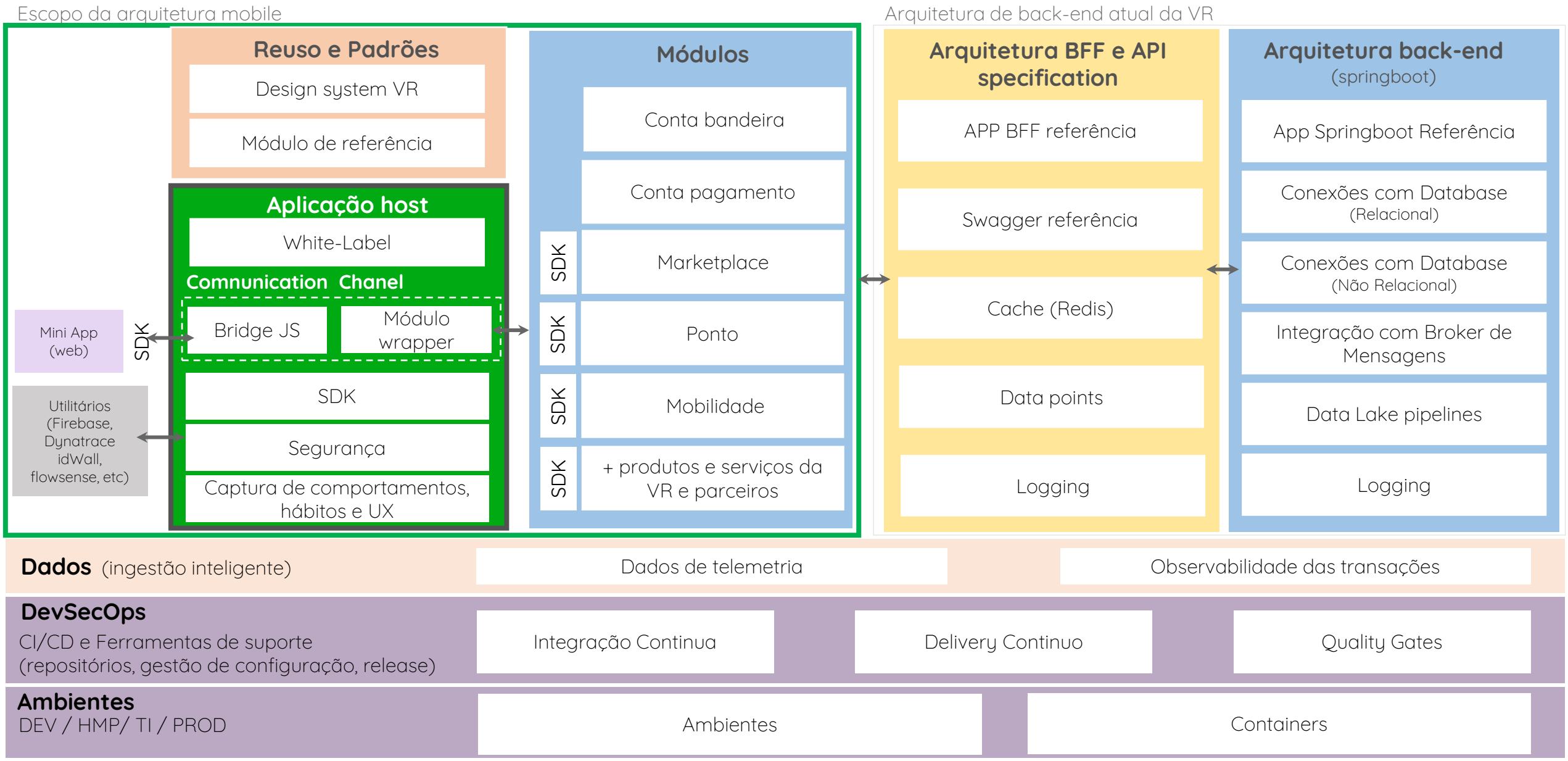
Atividades e ferramentas de automatização do fluxo de CI/CD

Ambientes

Desenvolvimento / Homologações / Labs / Produção

Visão da arquitetura

Design estrutural



Visão da arquitetura

Benefícios

Reuso de componentes

Consistência na identidade visual entre os diversos produtos e serviços.

Agilidade para integrar novos parceiros

Modelo mais **plugável**

Camada específica de segurança

Aumenta **pontos de controle** e eleva a confiabilidade estrutural da solução.

Capacidade de escalar o desenvolvimento

Flexibilidade para **aumentar capacity**

Maior eficiência e isolamento para testes

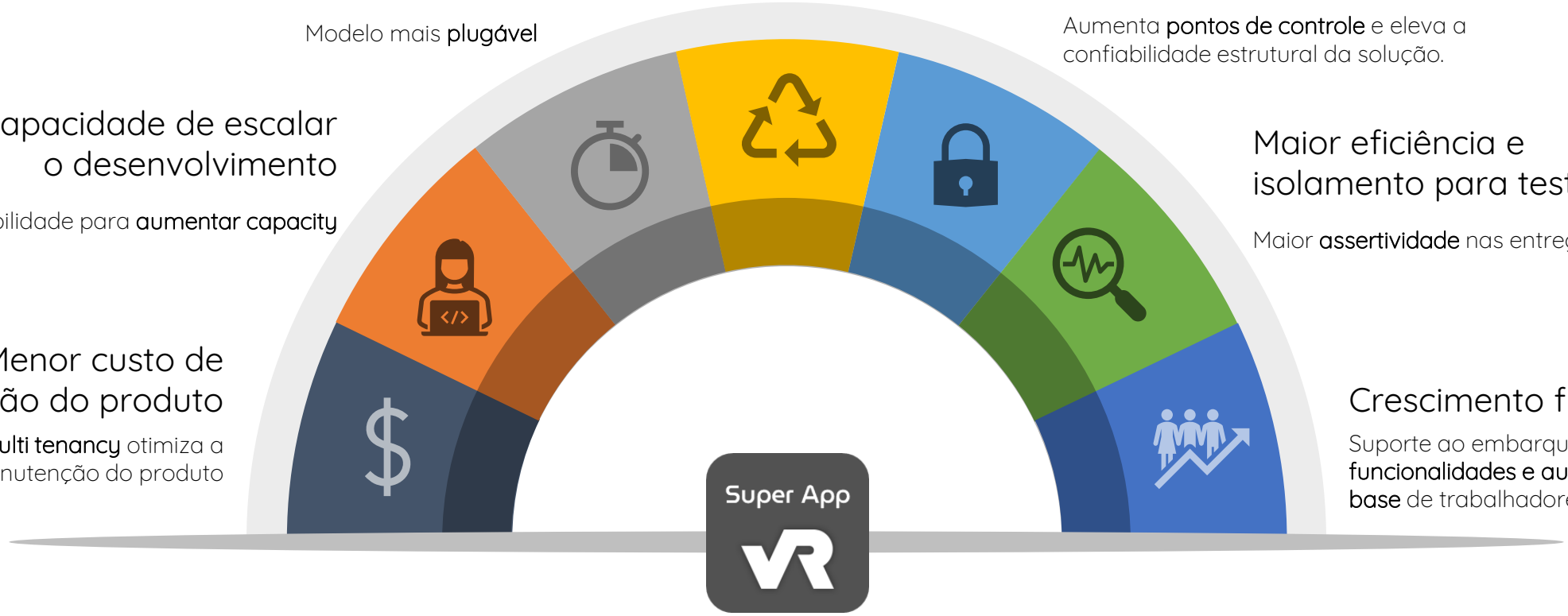
Maior **assertividade** nas entregas

Menor custo de sustentação do produto

Modelo **multi tenancy** otimiza a manutenção do produto

Crescimento funcional

Suporte ao embarque de **novas funcionalidades e aumento da base** de trabalhadores



Recomendação de plataforma

Recomendação de plataforma

Aspectos decisórios

Após análise conjunta de **aspectos técnicos, estratégicos e do contexto tecnológico da VR**, entendemos que o **REACT Native**, juntamente com componentes nativos, é a recomendação de plataforma adequada para construção do Super App.

- Custos de desenvolvimento e mudanças
- Time to market
- Mobilização de profissionais

- Modelo white label
- Escalabilidade do desenvolvimento
- Integrações externas
- Usabilidade
- Tamanho do App

- Competências atuais do time
- Reaproveitamento do legado

React Native

+

Complementos em plataformas nativas

android iOS

Comparativo tecnológico entre plataformas

Visão quantitativa ponderada

	Driver	Peso	Flutter	React Native	Nativo	Critério avaliativo
Técnicos	Whitelabel	2	4	3	4	Menor esforço de implementação.
	Modularização dentro da mesma plataforma	3	3	3	3	Capacidade de modularizar para escalar o desenvolvimento.
	Modularização entre plataformas	3	2	2	2	Capacidade de interoperabilizar aplicativos de outras plataformas.
	Integração com SDK de terceiros	3	4	3	5	Menor complexidade da integração.
	Pipeline CI/CD para a loja	1	4	4	4	Menor esforço de implantação do pipeline.
	Aderência da experiência à plataforma nativa - usabilidade	1	4	3	5	Melhor experiência de interface
	Distribuição ampla	2	2	3	4	Menor tamanho final do aplicativo
Estratégicos	Time to Market	3	4	4	2	Menor tempo entre concepção e lançamento de features. (considerando mesmo HH em cada uma).
	Custos	3	4	4	2	Menor custo em desenvolvimentos novos e alterações.
	Facilidade de mobilização de time	3	3	4	3	Disponibilidade de profissionais no mercado.
	Capability do time atual da VR em cada plataforma	3	2	4	4	Maior fit das competências técnicas do time com cada plataforma.
Contexto	Integração com ecossistema atual de parceiros	3	1	4	3	Maior facilidade para incorporar as plataformas existentes dos parceiros.
	Reaproveitamento da aplicação VR e VOCÊ (nativo)	2	3	3	4	Maior aproveitamento do código nativo legado.
	Reaproveitamento da aplicação Multi-benefícios (REACT)	2	1	4	3	Maior aproveitamento do código Reactive Native legado
	Aderência		**97**	**117**	**111**	

Alertas e recomendações

Elementos estruturais da arquitetura

Pontos de atenção

Uma **estrutura básica precisará será construída**.

Heterogeneidade da arquitetura dos parceiros.

Encapsular mini-apps e módulos para aumentar controle e segurança.

Arquitetura

Desenvolvimento

Serão necessárias **3 competências tecnológicas** (React Native/Android/iOS).

Reaproveitamento de funcionalidades será detalhado caso a caso.

Novo fluxo de dados demandará cuidados específicos.

Dados

Operação tecnológica

Ajustes de **capacidade e competências operacionais**, (custos, monitoramento, riscos de segurança, etc).

Elementos estruturais da arquitetura

Deixando claro...

Para integrar múltiplas soluções, inclusive de empresas distintas, observar que ...

- Mesmo sendo uma arquitetura plugável, haverá a necessidade de **orquestrar as adaptações no ecossistema** para viabilizar as integrações ao Super App.
- **Integrações não são seamless** e podem não ser triviais. Prevemos mecanismos e procedimentos para facilitar, contudo, podem existir condições não previstas.
- Pode haver a necessidade de solucionar **incompatibilidades estruturais pontuais entre as arquiteturas de terceiros e o Super App**, com consequente **risco de atrasos** para integrar tais atores ao ecossistema.
- É imprescindível **detalhar os esforços** para validar os prazos do roadmap, tanto para produtos e serviços da VR quanto de parceiros.

Próximos passos

Próximos passos

Ações recomendadas para implementar a arquitetura

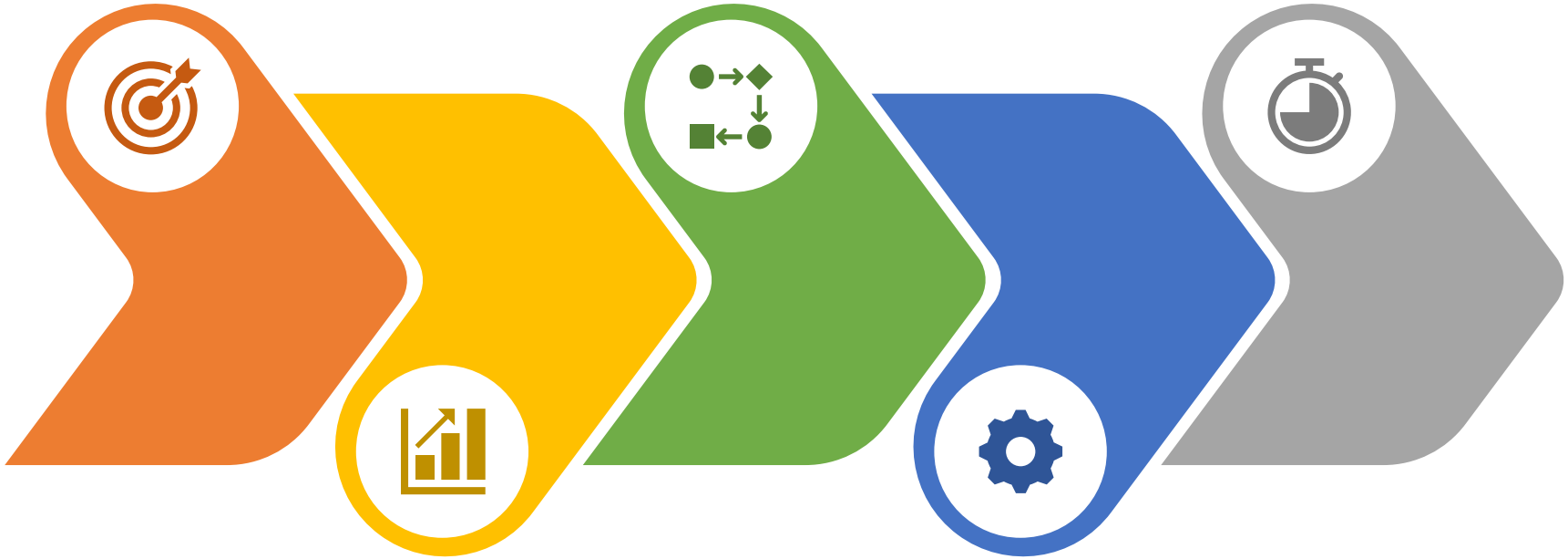
Detalhar a implementação.

Avaliar o investimento para construção da estrutura básica.

Revisar processo de desenvolvimento.

Construir aplicações de referência.

Embarcar produto para maturar os conceitos das integrações.



Super App VR

Fim da apresentação executiva

Detalhamento da arquitetura

Design estrutural da arquitetura

Blueprint

Escopo da arquitetura mobile

Arquitetura de back-end atual da VR

Reuso e Padrões

- Design system VR
- Módulo de referência

Aplicação host

- White-Label
- Comnunication Chanel
  - Bridge JS
  - Módulo wrapper
- SDK
- Segurança
- Captura de comportamentos, hábitos e UX

Mini App (web)

SDK

Utilitários (Firebase, Dynatrace idWall, flowsense, etc)

Módulos

- SDK Conta bandeira
- SDK Conta pagamento
- SDK Marketplace
- SDK Ponto
- SDK Mobilidade
- SDK + produtos e serviços da VR e parceiros

Arquitetura BFF e API specification

- APP BFF referência
- Swagger referência
- Cache (Redis)
- Data points
- Logging

Arquitetura back-end (springboot)

- App Springboot Referência
- Conexões com Database (Relacional)
- Conexões com Database (Não Relacional)
- Integração com Broker de Mensagens
- Data Lake pipelines
- Logging

Dados (ingestão inteligente)

- Dados de telemetria
- Observabilidade das transações

DevSecOps

CI/CD e Ferramentas de suporte (repositórios, gestão de configuração, release)

- Integração Continua
- Delivery Continuo
- Quality Gates

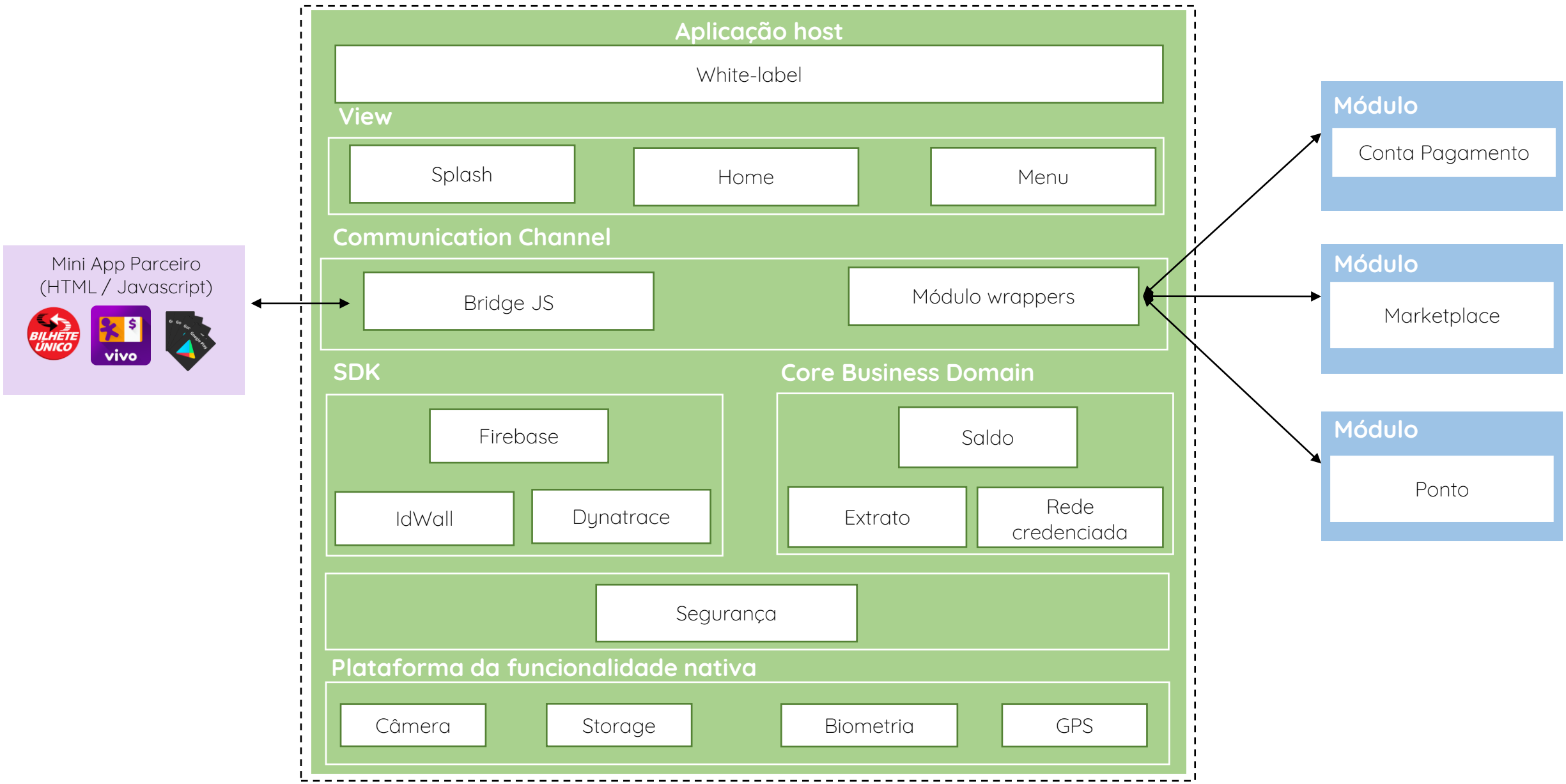
Ambientes

DEV / HMP/ TI / PROD

- Ambientes
- Containers

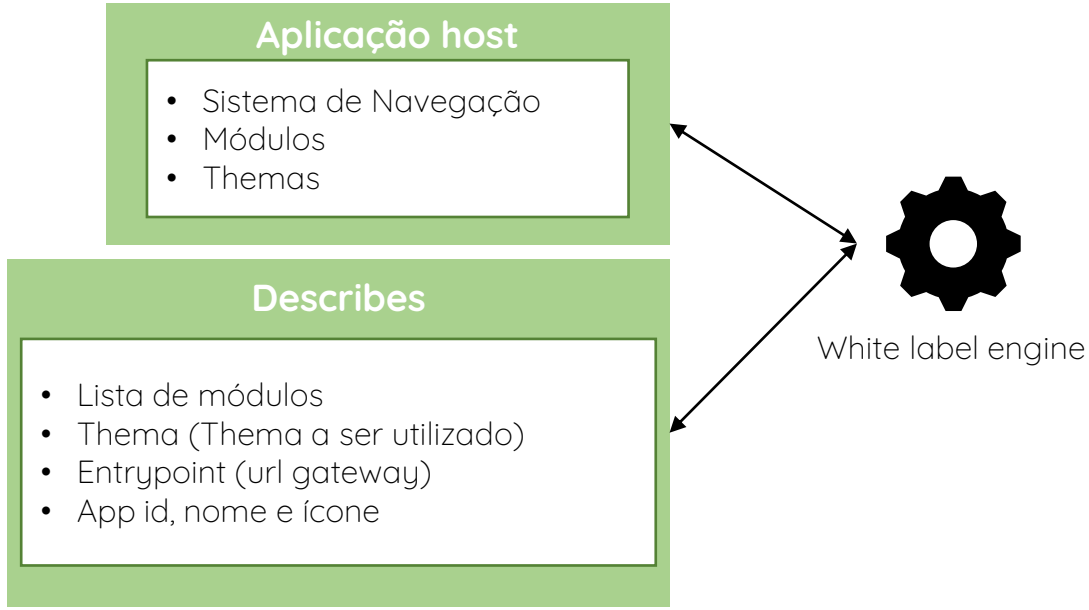
Detalhamento da arquitetura

Estrutura central do Super App



Detalhamento da arquitetura

White-label

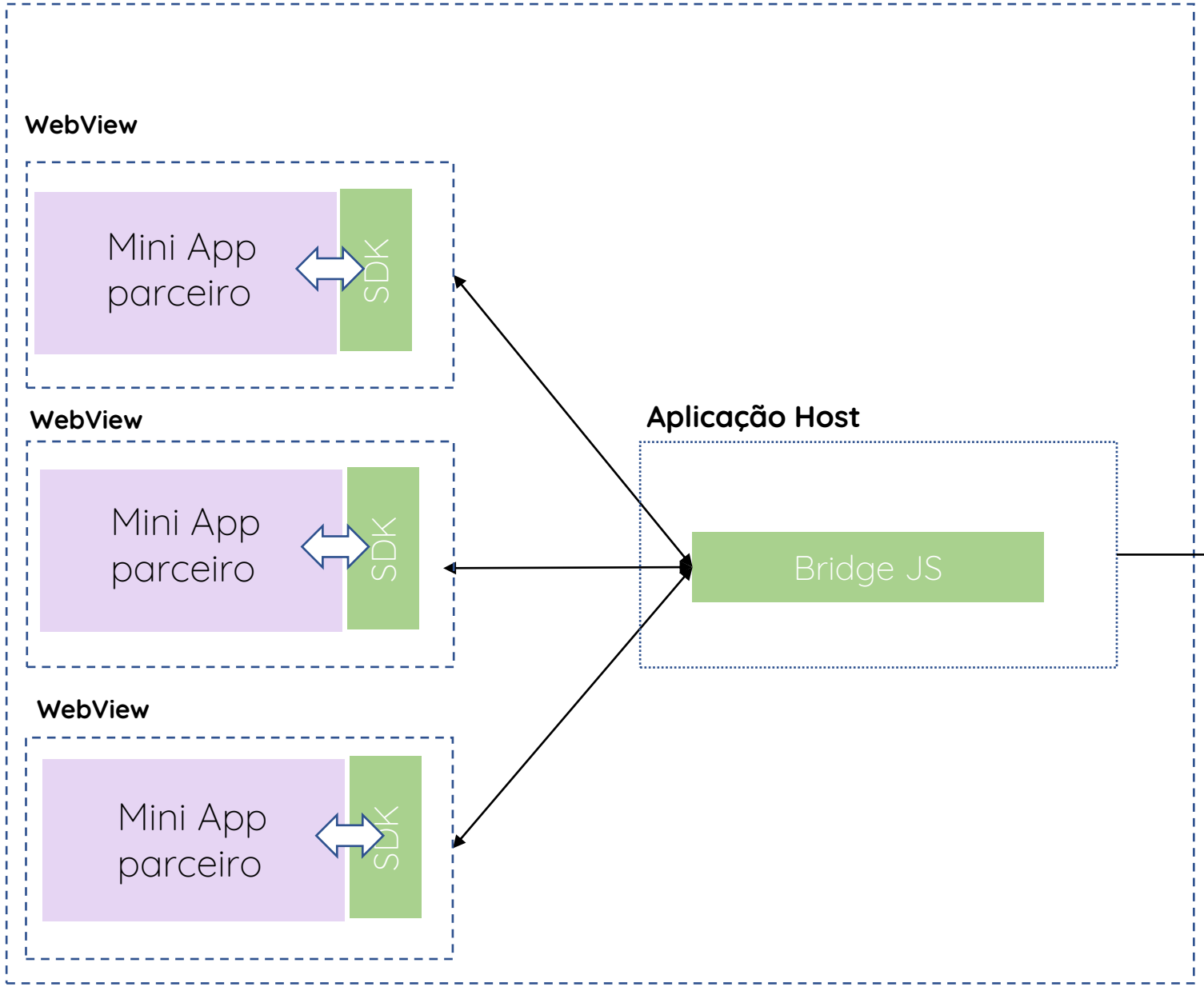


A abordagem multi-tenancy foi pensando sendo sustentável por um design system consistente, que possa ser facilmente alterado com mudanças simples de variáveis entre os diferentes flavours, e que cada tela possa ser alterada de um flavour para outro seja corretamente modularizada em tempo de codificação, visto que o design system vai guiar toda a arquitetura interna do Super App, de forma a minimizar o esforço na hora de geração de um outro flavour, seja por mudanças simples de cores, mudanças maiores como formatos de botões, e até carregamento ou não de módulos inteiros dentro do app.

Para isso foi adotada a estratégia de definição de um módulo base da aplicação e de um guia de estilo, passando por uma paleta de cores apontando por "cor do título" ou "cor principal" que pode ser bem distintas entre cada flavour, da mesma forma os assets devem ser apontados de forma genérica refletindo seu uso interno no app.

Detalhamento da arquitetura

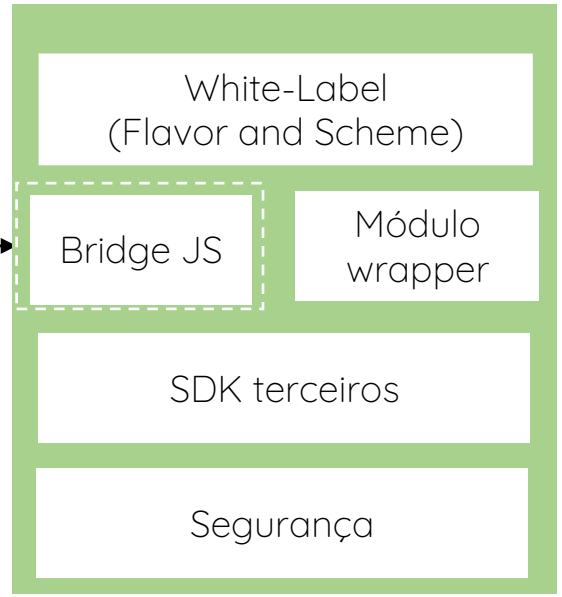
WebView



Para estabelecer a comunicação com os Mini App e o SuperApp VR, o parceiro deverá comunicar-se com – Bridge JS – através de um SDK disponibilizado pela VR.

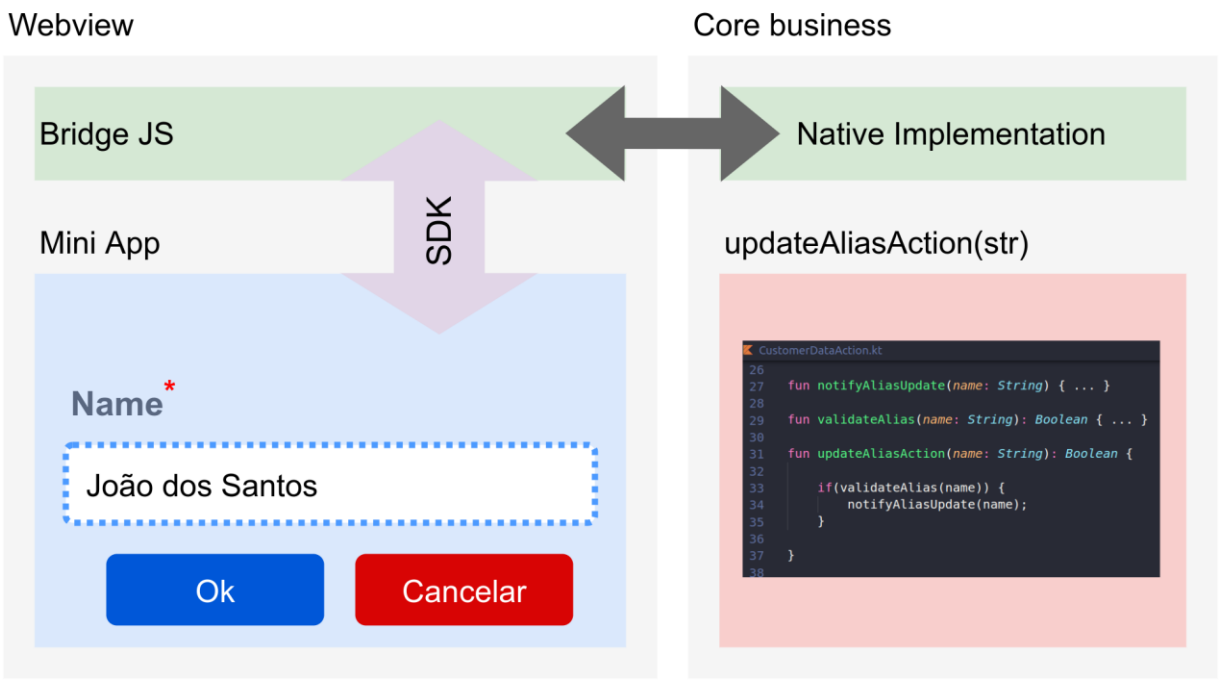
SDK é responsável, por:

- Disponibilizar ações pré-estabelecidas.
- Disponibilizar callbalck pré-determinados.
- Estabelecer controles de segurança.



Detalhamento da arquitetura

WebView - exemplo



A comunicação para Mini Apps e o Super App VR, foi prevista através de uma integração com o parceiro, via WebView e a ponte (Bridge JS) do aplicativo host.

A integração de parceiros também será possível através de conexões com “aplicações web”. Para isso, foi proposta uma solução de integração baseada em Webview. Os serviços que foram se integrar através desse recursos vão se comunicar com o aplicativo host por meio de uma bridge js, que é responsável pela orquestrações das requisições feitas pelo mini app, e garantir mecanismos de segurança.

Para estabelecer um padrão de comunicação todo mini app que for integrado ao aplicativo host, precisa implementar no seu produto “aplicação web” um contrato de comunicação através de um SDK que será fornecido pela VR.

Detalhamento da arquitetura

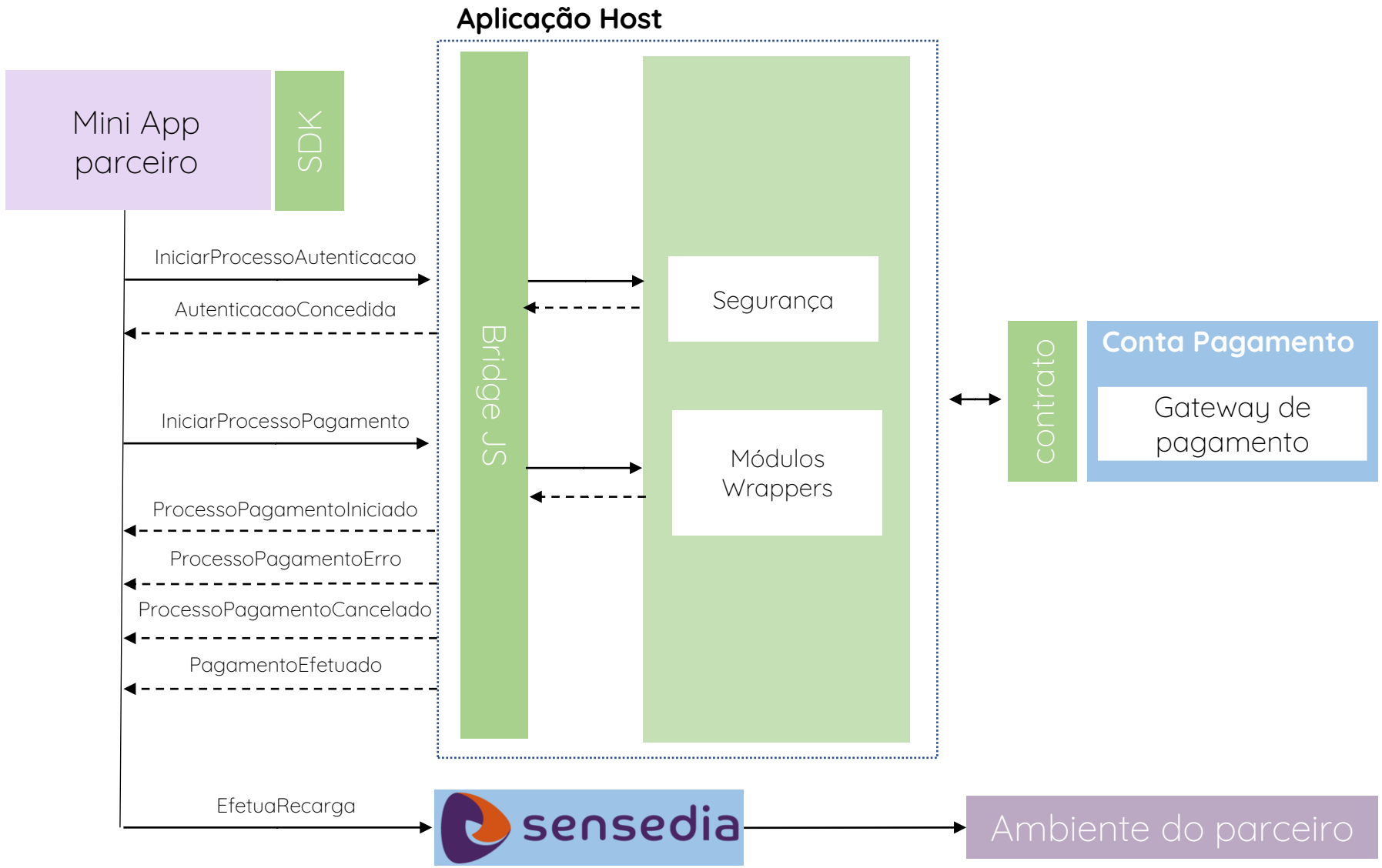
Exemplo de integração de Mini App com WebView

Supondo o cenário em que o mini-app de um parceiro efetua recarga de celulares seja integrado ao Super App VR.

O Fluxo se inicia com a ação do usuário escolhendo os valores e número do aparelho
e confirmando a intenção de recarregar.

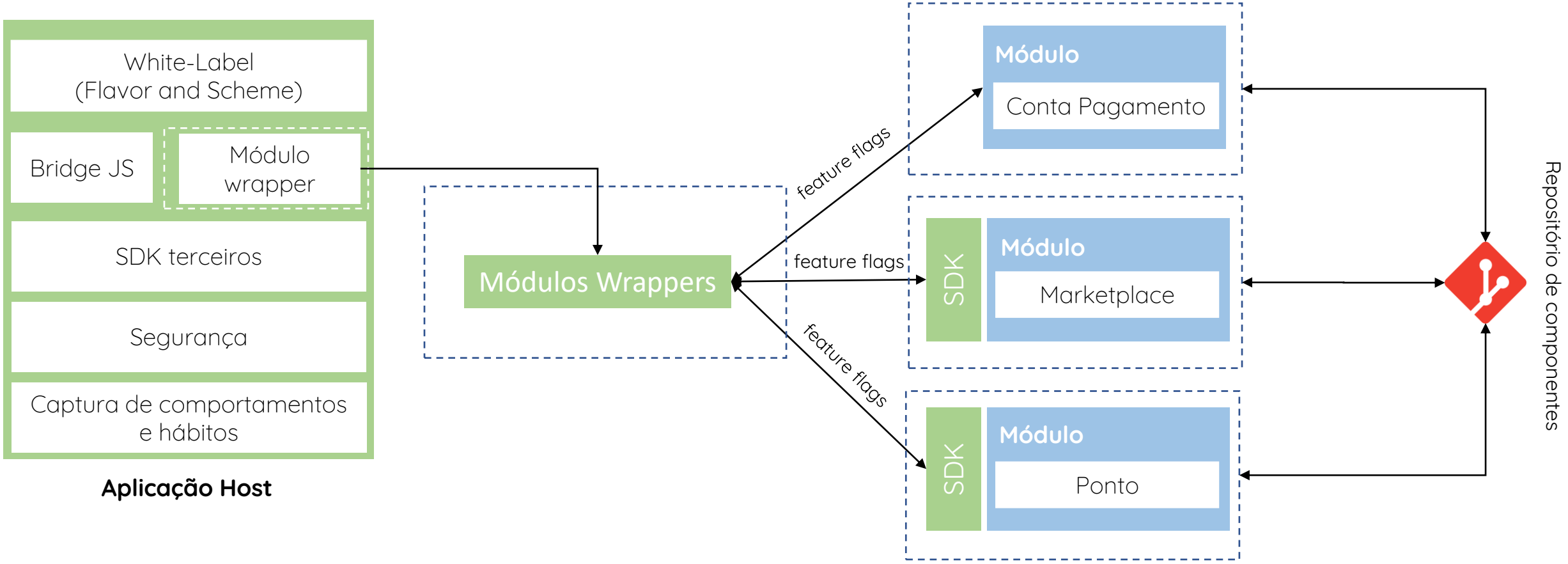
Em seguida a mensagem com os dados necessários para a recarga é enviada para o Super App (aplicação host) obedecendo o contrato com a SDK.

A Bridge JS entrega a mensagem para o módulo responsável por pagamentos, que por sua vez debitará o valor correspondente da carteira do cliente e retornar para o MiniApp efetuar a chamada da recarga através do API GW da VR.



Detalhamento da arquitetura

Modularização



Cada módulo precisará ter um contrato definido com a aplicação host para garantir que o host sempre o inicialize e para compartilhamento de métodos entre módulos, possibilitando assim o reuso de funcionalidades.

Os componentes visuais dos módulos e da aplicação host devem seguir um padrão de reuso estabelecido através de um design system (repositório de componentes).

Detalhamento da arquitetura

Modularização

Para dar suporte aos aspectos estratégicos e do contexto tecnológico da VR, foram observados os seguintes detalhes no arranjo da arquitetura da modularização.

O tempo de build da aplicação mobile degrada a produtividade a medida que novas funcionalidades são agregadas no aplicativo.

A necessidade de escalabilidade do desenvolvimento de novas funcionalidades, de forma independente.

Ter maior eficiência e isolamento nos testes de novas funcionalidades

Para endereçar essas necessidades, foi adotada a estratégia de modularização do aplicativo, através do conceito de (Multi-Module Architecture), nesse modelo a **orquestração dos módulos é de responsabilidade da aplicação host**, responsável por inicializar os módulos a partir de mecanismos de configurações (feature flags) e prover toda **infraestrutura para acoplamentos** destes módulos.

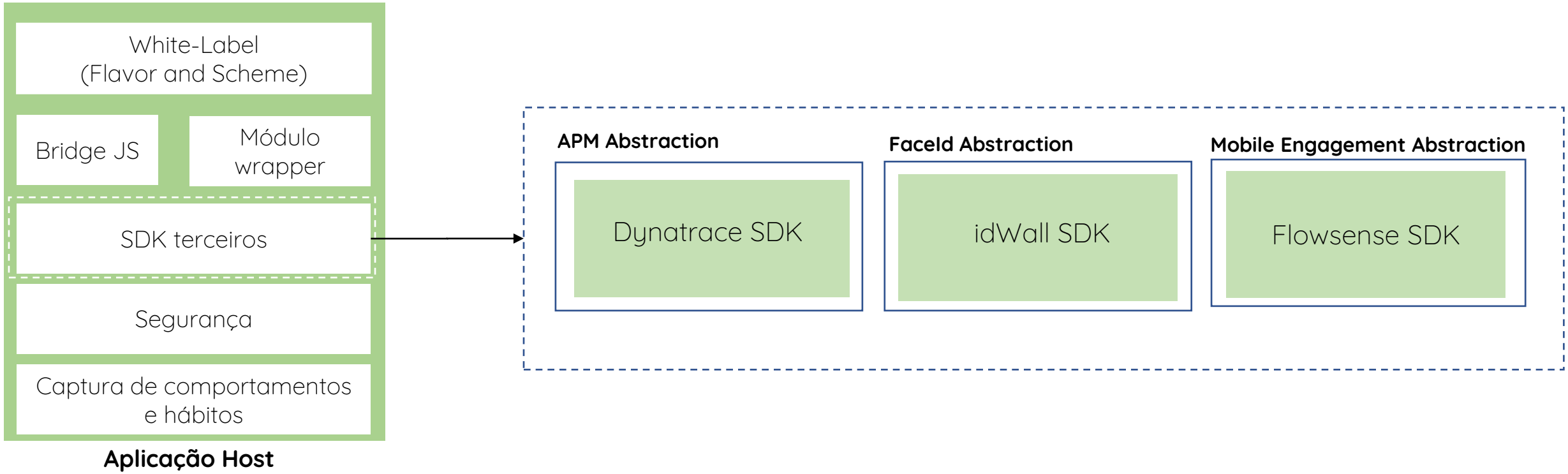
Nessa abordagem, os módulos não precisam conhecer ou ter dependências entre si, o conceito de módulo empregado deverá ser o feature module (processo capaz de ativar ou desativar grupos de funcionalidades sem necessidade de uma nova release de software).

Os módulos devem se conectar ao host, **preferencialmente através de um contrato (SDK),** com isso, será garantido mecanismos de segurança pela aplicação core.

No processo de desenvolvimento, a equipe responsável por um determinado módulo, poderá ter o **desenvolvimento desacoplado** de outros módulos, precisando apenas ter o aplicativo host para se conectar e terá a capacidade de realizar todo o ciclo de compilação e testes de forma independente.

Detalhamento da arquitetura

SDK para integração com terceiros



Para facilitar a manutenibilidade e minimizar eventuais quebras de compatibilidade (a exemplo da depreciação de componentes), deverão ser construídas abstrações para cada SDK de terceiros incorporado.

Com essa abordagem também é possível minimizar o impacto da troca de ferramenta de parceiros (por exemplo: troca de Dynatrace por AppDynamics).

Integração com SDK ou estruturas de terceiros podem ser um grande risco de segurança para o Super App. Como eles são compilados em conjunto e executados na mesma sandbox, os SDK têm os mesmos direitos que seu aplicativo. Isso significa que um SDK malicioso pode buscar a localização de seus usuários se você solicitar essa permissão ou até mesmo ler dados armazenamento no aplicativo.

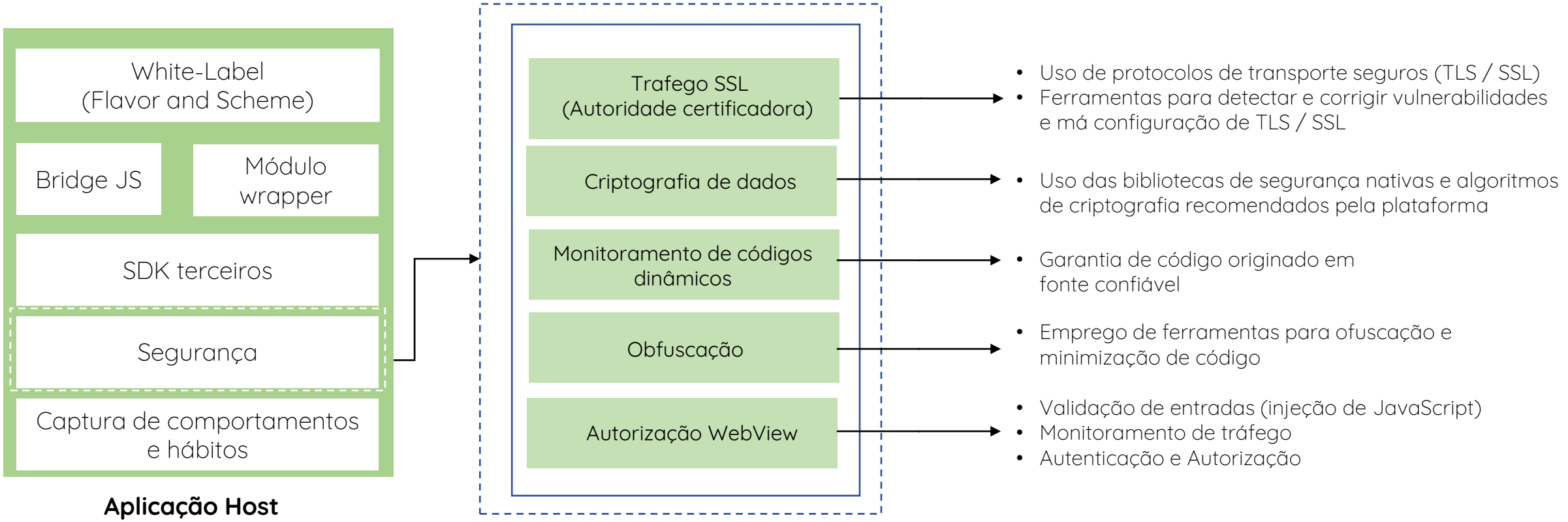
Na arquitetura proposta incorporamos uma camada de abstração para conectar as diversas plataformas via SDK. O objetivo dessa abstração é diminuir possíveis adaptações no Super APP decorrentes de mudanças na utilização dos SDK, possibilitar a migração entre fornecedores com o menor impacto possível, além de integrar esses SDK com a camada de segurança.

A camada de abstração deve garantir uso otimizado aos sistemas de APIs dos SDK, evitar vazamento de memória por parte do SDK, adicionar controles de segurança e garantir performance na implementação das chamadas ao recursos das implementações dos terceiros.

Detalhamento da arquitetura

Segurança

- Proteção dos dados dos usuários.
- Proteção dos recursos do sistemas (ex.: vazamento memoria, câmera, biometria).
- Fornecer isolamento do aplicativo do sistema, de outros aplicativos e dos usuários.



Detalhamento da arquitetura

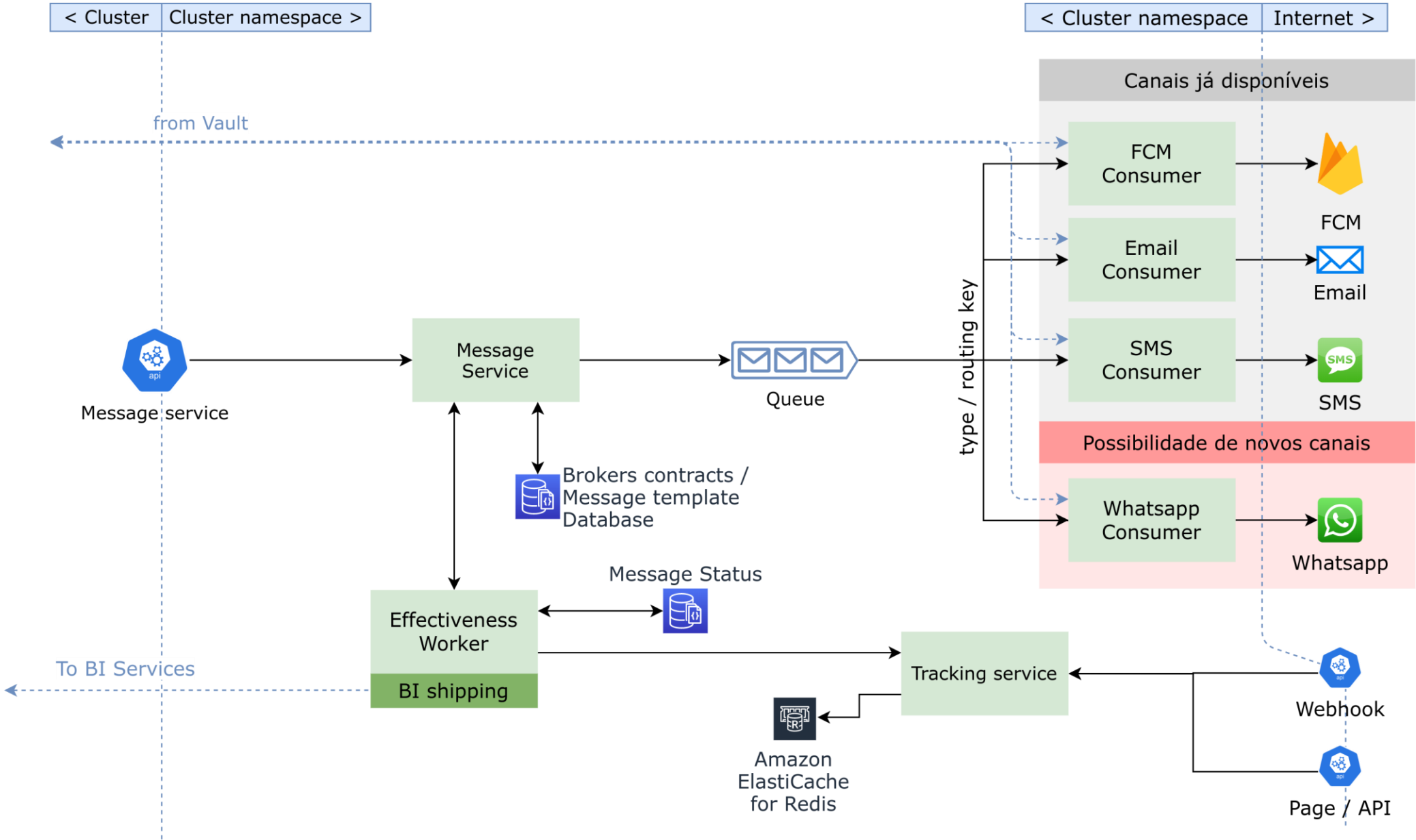
Segregação de notificações

Proposição arquitetural para separação do envio de mensagens por canal de comunicação e segregação de contratos e credenciais por tenants.

Esta é nossa releitura sobre o “Notification Center” da VR pois entendemos que este é um elemento viabilizador da estratégia de Whitelabel.

Como a arquitetura é flexível e multi canal, fica a possibilidade de agregar novos canais tais como whatsapp, telegram, etc.

Também é importante ressaltar que este elemento é parte de do backoffice e pode necessitar de aprofundamento no acoplamento.



Macro Arquitetura

Segregação de notificações

A estratégia de segregação de notificações foi pensando para **prover isolação do sistema de comunicação** com o usuário, através de uma interface (contrato) que é conhecida, documentada e gerenciável.

O controle de utilização e segregação de tenants é realizada através de identificação de aplicação / cliente chamador. Desta forma permite que os assets necessários para a construção da mensagem final sejam selecionados.

A solução foi concebida para ter um **desacoplamento dos canais de comunicação**, através de consumidores, com possibilidade de expansão horizontal. Esses consumidores são responsáveis pela entrega de mensagens de um determinado canal, que são associados a uma chave de roteamento(routing_key) específica do canal dependendo do tipo da mensagem. Os consumidores podem escalar horizontalmente para ampliar a capacidade de entrega geral do sistema.

A topologia das filas pode ser reconfigurada para beneficiar a performance de entrega de um determinado tenant.

Macro Arquitetura

Segregação de notificações

Como abordagem para garantir segurança, cada instância que tenha responsabilidade de se comunicar com brokers externos deve ter acesso as credenciais necessárias realização deste trabalho. Estas credenciais devem estar disponíveis somente leitura em um serviço de vault.

O acesso ao serviço de entrada "Message service" bem como a comunicação entre os serviços internos ao namespace deve ser efetuada por conexão segura. Esse acesso, deve reforçar a utilização de credencial que contenha os dados necessários para identificar a aplicação(cliente) e do tenant, bem como possuir escopos e papéis compatíveis com o negócio.

Para gestão da efetividade da comunicação das notificações com o usuário. Foi sugerido :

- Webhook – API que recebe a notificação (callback) de confirmação de leitura. O broker escolhido para lidar com as mensagens deve ser capaz de informar via webhook quando o destinatário efetuar a leitura da mensagem.
- Página de redirecionamento – Página que após renderizada, notifica o serviço de leitura com o token adequado e redireciona o fluxo de navegação para uma outra URL.
- API – Chamada diretamente para notificar leitura da mensagem. Quando o canal for Aplicativo móvel o próprio aparelho deve fazer a chamada síncrona ou não, batch ou individual à API.

Detalhamento da arquitetura

Fluxo de dados

Pontos de coleta

Assegurar que o Super App colete telemetria sobre comportamentos, a utilização dos produtos e serviços.

Pipeline de dados

Viabilizar o fluxo seguro de informações de comportamentos, hábitos e usabilidade desde o dispositivo até a entrega na camada analítica da retaguarda VR.

Governança

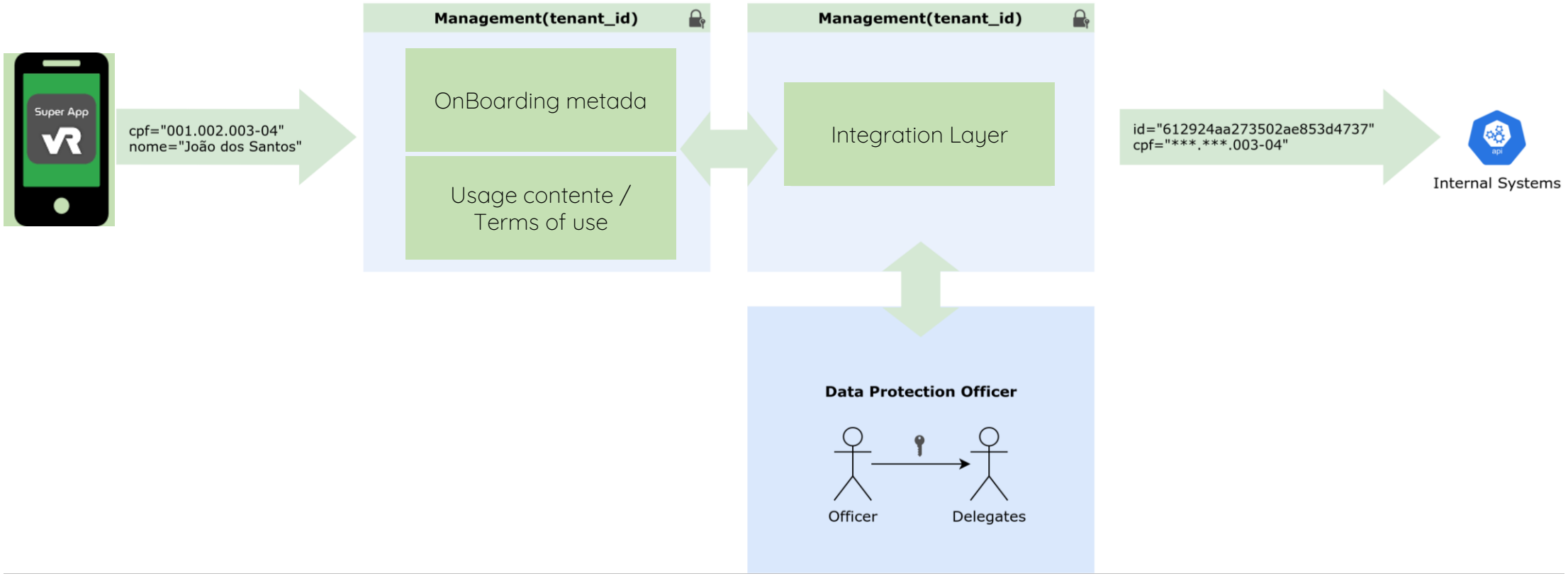
Permitir que a governança corporativa de dados tenha visibilidade e imponha controles de conformidade sobre os dados manipulados no Super App.

Detalhamento da arquitetura

Segregação de LGPD e dados de usuários

Proposição arquitetural para separação do onboarding, dados e termos de uso por tenants.

SOLUTIS



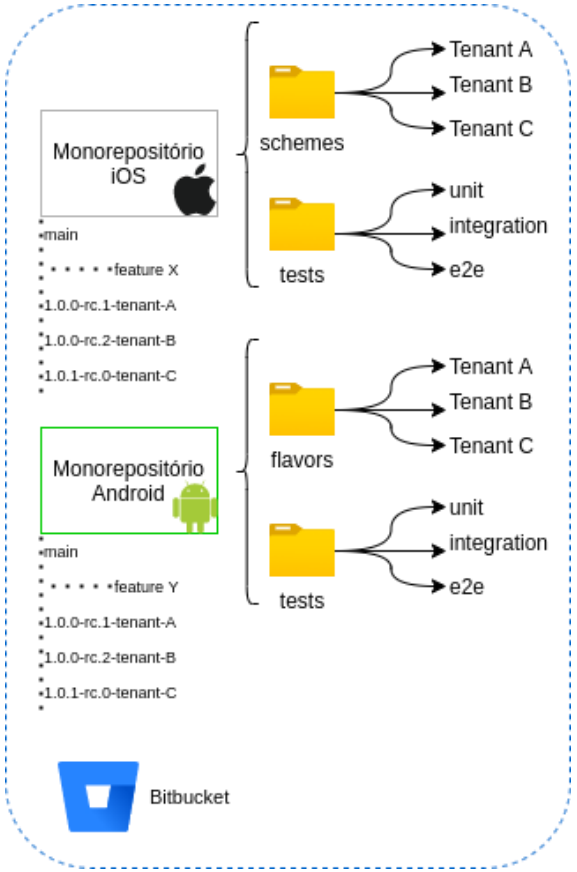
Detalhamento da arquitetura

Mobile CI/CD

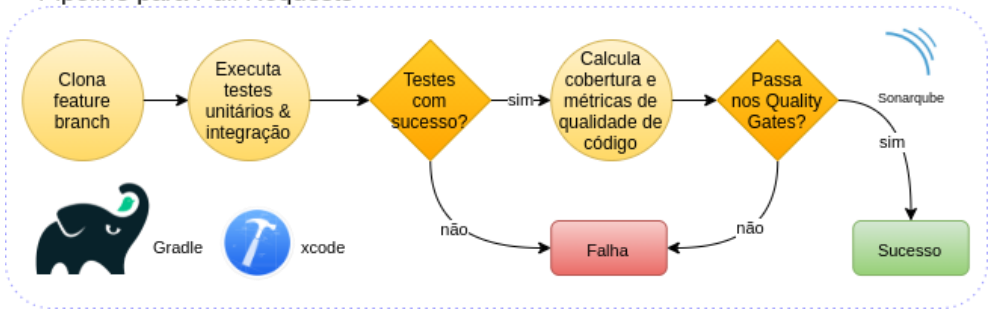
Proposição de fluxos padrão para integração & delivery (CI/CD) de SuperApp whitelabel nas lojas Google e Apple

SOLUTIS

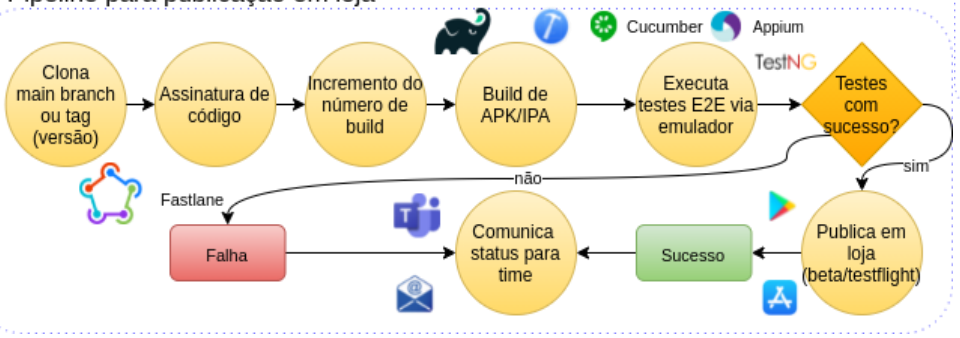
A proposição sugere a organização do SuperApp em monorepositórios utilizando Schemes(iOS) e Flavors(Android) que irão definir cada tenant do whitelabel. Os scripts para testes unitários, de integração e E2E (end to end) estão juntos contidos em cada repositório. Por padrão, o pipeline é acionado automaticamente por commit em branch master e tags (versões) e o build é feito para todos os tenants. Para um build de um tenant específico, é necessário o acionamento manual com configuração de parâmetros.



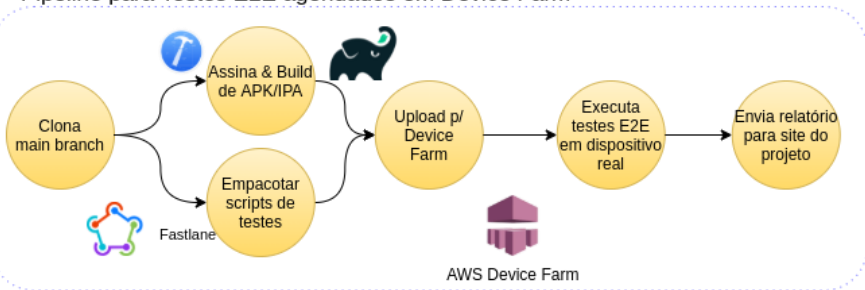
Pipeline para Pull Requests



Pipeline para publicação em loja



Pipeline para Testes E2E agendados em Device Farm



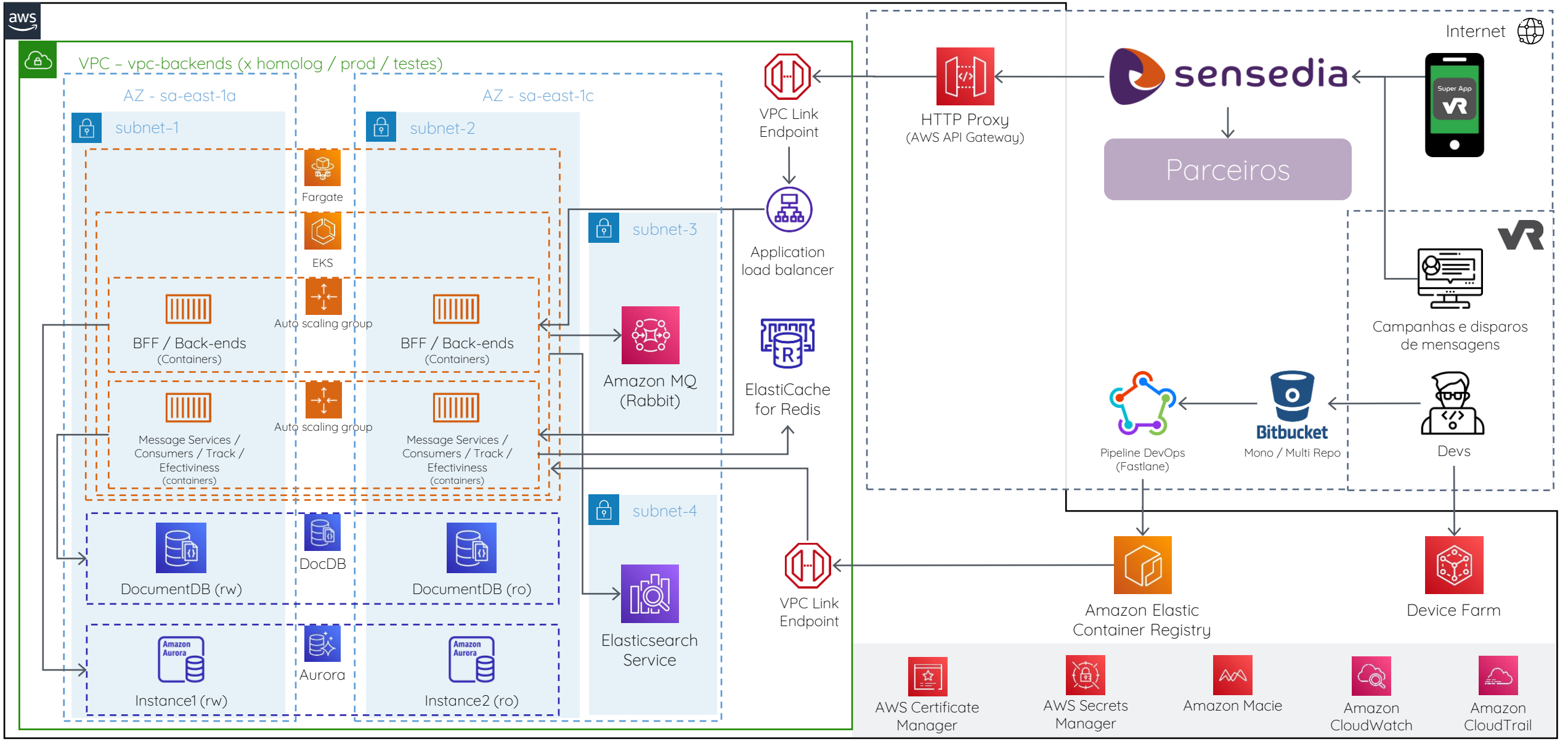
Detalhamento da arquitetura

Comparativo de CI/CD SaaS x On-Premises

Critério de avaliação	SaaS	On-Premises
Menor risco de acesso indevido ao código-fonte da aplicação e dados sensíveis (IDs e tokens p/ publicação nas lojas de apps).	2	5
Menor custo de implantação do serviço (infraestrutura, pessoas, ferramentas).	5	2
Menor custo de operação do serviço (infraestrutura, pessoas, ferramentas).	5	2
Maior flexibilidade na escolha de ferramentas e configuração.	2	5
Maior facilidade para aumentar a capacidade quando necessário.	5	3
Média	3,8	3,4

Detalhamento da arquitetura

Visão de infraestrutura em AWS e SaaS (Sensedia e Bitbucket)



Detalhamento da arquitetura

Posicionamento das tecnologias

SOLUTIS

Acessórios

- ElastiCache for Redis
- VPC Link Endpoint
- Application load balancer
- AWS Secrets Manager
- Amazon CloudWatch

Canais

Super App

Integração

sensedia Gateway de API

Segurança

BFF

Java BFF Mobile

Java BFF Portal

Microserviços

Java Onboarding

Java Cartão Virtual

Java QR Code

Framework

spring

Legacy back-end

- 4time WAS
- IBM Barramento
- Java Rede Credenciada
- Java MC Gestão

Database

- Amazon RDS
- ORACLE DATABASE
- AWS RDS MySQL
- AWS RDS PostgreSQL
- Amazon MQ (Rabbit)
- Elasticsearch Service

Segurança

- SSL
- JWT
- RH-SSO

DevSecops

- Amazon EKS
- docker
- ELK
- dynatrace
- sonarqube
- Maven
- JUnit 5
- JIRA
- Postman
- VS Code
- eclipse
- Bitbucket
- bitrise
- cucumber

Plataforma de desenvolvimento

Plataforma de desenvolvimento

Comparativo cross platform x plataforma nativa

Driver	Flutter	React native	Nativo
White label	Flutter se utiliza de recursos nativos, através da sua estrutura conhecida como "Plataform Chanel" utilizando a mesma abordagem de implementação da solução nativa. (flavor e targets)	Utilizando uma abordagem de fontes de assets para cada parceiro, gerenciado por plugins do (compilador javascript), porém com um considerável esforço de customização dos componentes.	Utilizando da abordagem de product flavors gerenciado pelo build.gradle no Android, no iOS é feito através de targets porém demanda custo de implementação no iOS.
	Justificativa nota: O Flutter possui um mecanismo de reuso de componentes que garante maior aderência que o react para esse driver. A implementação da solução em Cross Plataform em Flutter é similar para as versões em iOS e Android.		
Modularização dentro da mesma plataforma	Pode ser alcançada através de "Abordagem Federada" que é utilizado pelo time do Flutter como solução de implementação de Plugins da plataforma.	Pode ser alcançado através de uma arquitetura "Multi-Module" onde o app core em react é responsável pelas gestão das bibliotecas nativas.	Modularização pode ser concebida através de controle dependências de aplicação (Gradle and Cocoapod)
	Justificativa nota: Todas as três opções (flutter, react e nativo) atendem a modularização com um esforço de implementação similar.		
Modularização entre plataformas	Flutter pode ser incorporado a aplicações nativas existentes como bibliotecas ou módulos. Esse é um suporte nativo da plataforma.	React pode embarcar códigos nativos dentro da sua plataforma conforme especificado em sua documentação.	É possível embarcar códigos em javascript dentro de uma plataforma nativo, porém o nível de integração é complexo.
	Justificativa nota: Todas as três opções (flutter, react e nativo) enfrentam dificuldades similares para adicionar código embarcado de outra plataforma. Implementação não trival para orquestração entre os módulos.		
Integração via SDK	Para utiliza-se de reuso através de SDK, é necessário implementação de canais (plataform channels) para ambas as plataformas iOS e Android e para cada um dos SDK a serem integrados, tornando o processo bem oneroso.	Para utiliza-se de reuso através de SDK, é necessário implementação de pontes (react native bridges) para ambas as plataformas iOS e Android e para cada um dos SDK a serem integrados, tornando o processo bem oneroso.	A integração com os SDK ocorre da mesma forma que qualquer outra biblioteca, pois é um recurso natural das plataformas.
	Justificativa nota: Implementação bastante custosa nas plataformas cross, diferentemente da plataforma nativa que a integração é um processo já suportado pelo Android e iOS.		

Plataforma de desenvolvimento

Comparativo cross platform x plataforma nativa

Driver	Flutter	React native	Nativo
CI/CD para loja	Utilização de uma ferramenta SaaS para CI/CD. Ex.: Bitrise ou montagem de uma infraestrutura on-premise.	Utilização de uma ferramenta SaaS para CI/CD. Ex.: Bitrise ou montagem de uma infraestrutura on-premise.	Esse cenário é mais custo que as plataformas cross devido a construção do pipeline de dois fluxos para Android e um iOS.
	Justificativa nota: Esforço de implementação similar nas plataformas cross, no entanto, a implementação é um pouco mais custosa para a versão nativa devido a necessidade de dois fluxos de pipeline.		
Time to market	Por se tratar de uma solução cross plataforma o esforço de desenvolvimento em Flutter tende a ser menor comparado com a solução nativa, onde é necessário implementar as mesmas funcionalidades em linguagens diferentes.	Por se tratar de uma solução cross plataforma o esforço de desenvolvimento é menor comparado com a solução nativo onde é necessário implementar as mesmas funcionalidades em linguagens diferentes.	O esforço de desenvolvimento é maior devido a necessidade de implementação das funcionalidades para ambas as plataformas, Android e iOS.
	Justificativa nota: Para as plataformas cross o esforço de desenvolvimento tende a ser menor, devido a um código único e ganho de produtividade devido a profissionais especializadas em apenas uma linguagem de desenvolvimento.		
Time de desenvolvimento	Por ser uma plataforma recente a quantidade de profissionais com proficiência na linguagem Dart ainda é escassa.	Devido a similaridade com a linguagem javascript, a curva de aprendizado dos profissionais que já atuam com frontend web para react native tende a ser menor.	Necessidade de profissionais especialista em cada uma das plataformas, Android e iOS.
	Justificativa nota: O Flutter tem menos profissionais no mercado por ser uma plataforma mais recente, já as plataformas nativas sofrem pela necessidades de profissionais especializadas em iOS e Android e que na sua maioria não estão propensos a migrar para uma tecnologia cross. O React possui a vantagem da linguagem javascript que é similar a outras plataformas de desenvolvimento e com isso traz uma curva de aprendizado menor para os desenvolvedores.		
Facilidades de desenvolvimento	O Flutter tem melhor escalabilidade, pois possui hot reload integrado que se adapta bem à base de código. Flutter possui ferramentas de depuração de código e IU que funcionam prontamente em ambas as plataformas, integradas ao IDE.	React também possui hot reload. A plataforma não possui funcionalidade de integração de teste oficial, porém, como é um framework javascript pode-se utilizar framework de terceiros como Jest	Não possui hot reload, sendo necessário aguardar o build para visualizar as mudanças. Ambos, Android e iOS possuem um framework oficial de testes com um conjuntos de testes básicos já sugeridos pela plataforma.
	Justificativa nota: Flutter tem uma plataforma de desenvolvimento bastante completa com hot reload e ferramentas de testes integrados da plataforma, que foi o diferencial em relação ao React que se utiliza de produtos/frameworks de terceiros. Já a versão nativa não possuem a funcionalidade de hot reload que ajuda na produtividade do time de desenvolvimento.		

Plataforma de desenvolvimento

Comparativo cross platform x plataforma nativa - conclusão

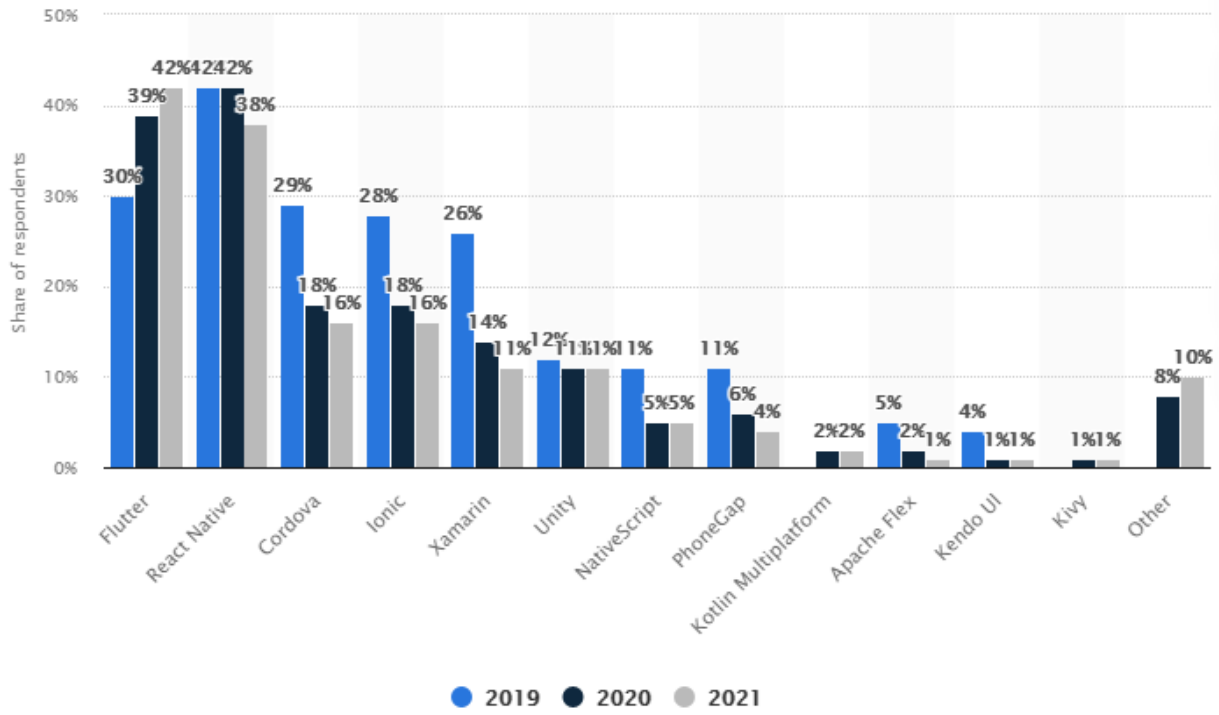
Cross Platform	Nativo
O desenvolvimento tende a ser mais genérico, não direcionado a uma plataforma especifica.	O desenvolvimento tende a ser mais direcionado à plataforma
Base de código comum entre as plataformas, ou seja, o código único para rodar em iOS e Android.	Uma base de código entre aplicativos para a mesma plataforma.
Tende a exigir maior esforço de desenvolvimento para não impactar perfórmance	Aplicativo tende a ser mais performático
Exige maior esforço de desenvolvimento para que o comportamento do código (UI) seja mais próximo ao que o usuário da plataforma está acostumado	O App naturalmente tende a se comportar e aparentar o que o usuário da plataforma está acostumado
Custo de desenvolvimento tende a ser menor	Custo de desenvolvimento tende a ser maior
Possível ambiente de desenvolvimento mais barato	Ambiente de desenvolvimento mais caro (iOS)
Equipe de desenvolvimento mais genérico	Equipe de desenvolvimento mais específico, com tendência a ser segregado pela plataforma
Novas features da plataforma dependem de implementação no Flutter	Novas features da plataforma estão naturalmente disponíveis

Plataforma de desenvolvimento

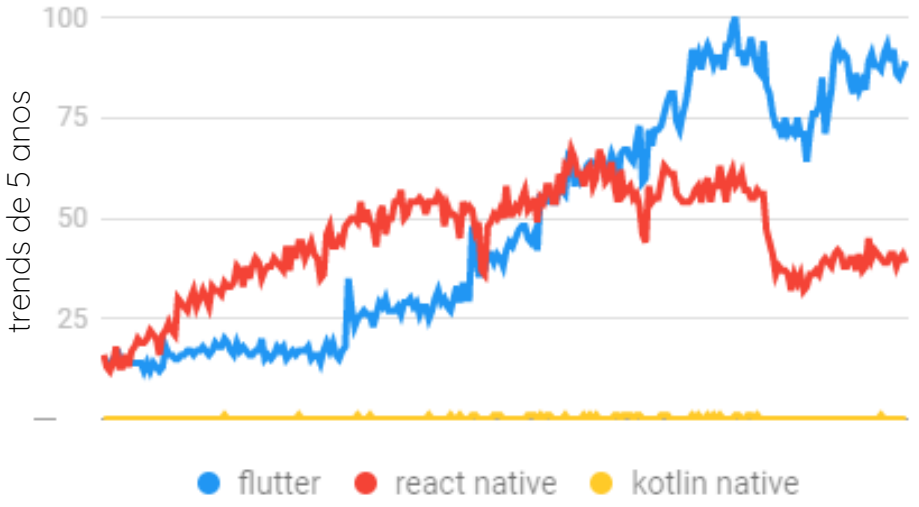
Pesquisas e estatísticas de adoção

statista

Adoção

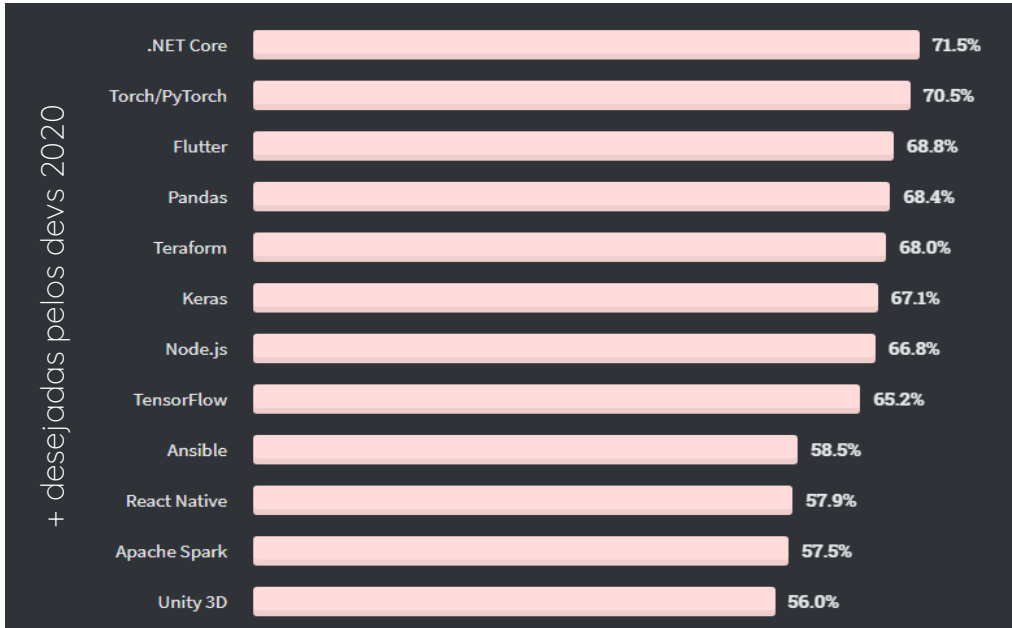


Google



stackoverflow

+ desejadas pelos devs 2020



Plataforma de desenvolvimento

Ponderações sobre a recomendação React Native + plataforma nativa

Após análise conjunta de aspectos técnicos e do contexto tecnológico da VR, entendemos que o REACT Native é a melhor opção de plataforma para o reaproveitamento potencial de iniciativas pregressas e baixa perda de benefícios tecnológicos.

Em complemento a este direcionamento, também recomendamos que alguns elementos sejam implementados em plataforma nativa para benefício da usabilidade, mecanismos de controle e segurança do aplicativo.

Super App VR =

React Native

- Aproveitar capability do time e disponibilidade de mercado.
- Incorporar legado da VR que já é mais voltado ao REACT.
- Implementar todo o código que puder ser generalizado para ambas as plataformas em REACT para reduzir custos e time to market.

+

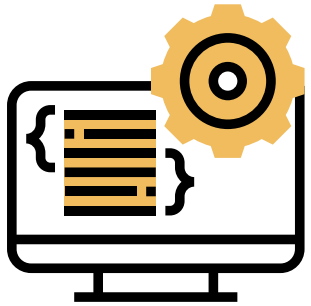
ANDROID iOS

- Implementar componentes que tornarão a experiência do usuário mais próxima da experiência convencional da sua plataforma.
- Viabilizar a implementação de funcionalidades que dependem de recursos nativos (biometria, câmera, GPS, giroscópio, dentre outros).
- Incorporar módulos com as funções do VR & Você (implementação nativa) e SDK de terceiros (geralmente nativos).
- Utilizar bibliotecas de segurança nativas por recomendação dos fabricantes.

Tecnologias cross plataforma

Considerações sobre evolução tecnológica futura

- A comunidade em torno do React Native é grande, madura e existe há mais tempo do que todas as outras alternativas.
- Em nossas pesquisas entendemos que a comunidade no React Native está mais envolvida e mais propensa a continuar o desenvolvimento se o Facebook decidir retirar o suporte oficial.
- Com relação ao compromisso do mantenedor, vemos o Flutter e o React Native com a mesma probabilidade de continuar sendo suportados a longo prazo.
- Embora o Flutter seja uma tecnologia mais recente, tem melhor documentação oficial e suporte do mantenedor e a comunidade está aderindo em ritmo acelerado.
- O Flutter foi um projeto de vários anos dentro da Google antes do lançamento oficial e está sendo usado atualmente em mais de 10 projetos no Google, por exemplo, o Google Adds que é público e se alcança milhões de usuários.
- Em 2021 nota-se um crescimento ainda maior do Flutter pela comunidade. Seu SDK se posiciona como um das plataformas mais desejadas pelos desenvolvedores para construção de aplicativos mobile.



SEE YOU SOON!

ANEXO

Cenário atual

- Aspectos funcionais
- Arranjo arquitetural
- Aspectos da implementação

Cenário atual

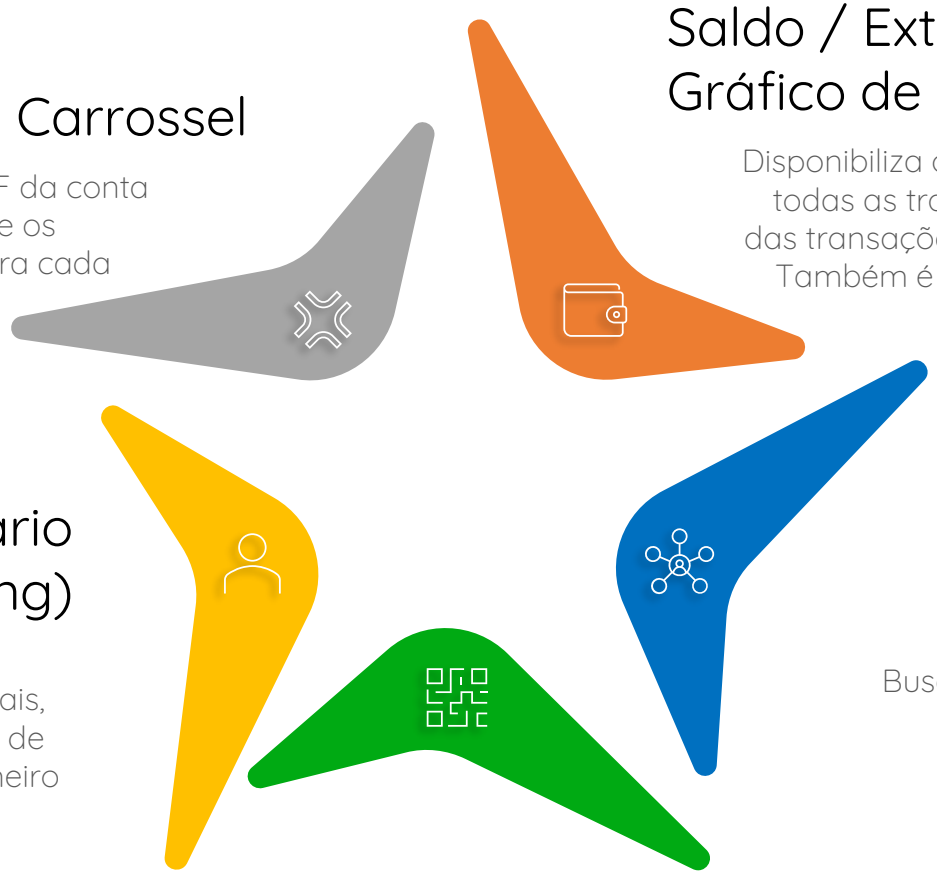
Funcionalidades atuais do aplicativo VR & VC

Carrossel

Exibe todos os cartões vinculados ao CPF da conta do usuário, onde é possível navegar entre os cartões e obter informações de saldo para cada um dos benefícios.

Saldo / Extrato / Gráfico de Utilização

Disponibiliza o saldo atual de cada cartão e o extrato de todas as transações realizadas. É possível ter detalhes das transações realizadas baseada em filtro de período. Também é disponibilizado um gráfico de utilização de cada benefício no período dos 30 dias.



Cadastro do usuário (Onboarding)

Formulário para criação do usuário com preenchimento de dados pessoais, definição da senha e aceite no termo de uso e politica de privacidade. No primeiro acesso é realizada a vinculação dos cartões ao CPF da conta criada.

Busca da Rede Credenciada

Busca da rede por endereço, nome ou CEP com possibilidade de favoritar no mapa os estabelecimentos de sua preferência.

Cartão Virtual / Pagamento via QR Code

Geração de cartões virtuais e pagamento via QR Code gerado a partir das maquininhas de cartões da Cielo ou via VR Pague.

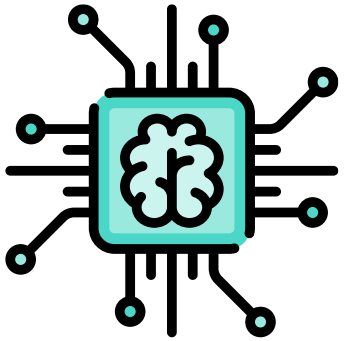
Arranjo arquitetural

Aspectos técnicos gerais

- Atualmente o **aplicativo mobile VR & VC** é desenvolvido em linguagem nativa (iOS e Android) e é baseado na arquitetura Model-View-Controller.

- A estrutura de serviços está dividida em dois ambientes,

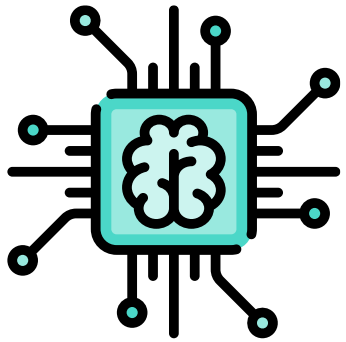
  - Um **ON-PREMISES na TIVIT**, que suporta o barramento IBM, 4Time, a aplicação SNCORE e o banco de dados Oracle.
  - E um ambiente na **CLOUD AWS,** onde encontram-se os serviços de notification center, gestão e promoção, MC busca, gestão de identidade e RHSSO, sendo que esses serviços são orquestrados pelo OPENSHIFT.
  - Há também o uso de serviços do Google (Firebase e Maps).



Arranjo arquitetural

Emprego das Tecnologias

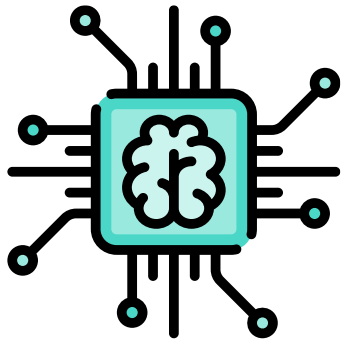
- Em termos tecnológicos, o aplicativo mobile é desenvolvido utilizando-se de **Objective C** e **Java** como linguagens principais. Contudo, para a versão iOS, as novas funcionalidades já estão sendo desenvolvidas em **Swift**.
- O Aplicativo mobile não adota a estratégia de armazenamento em bancos de dados local. As poucas informações que são salvas no dispositivo, utiliza-se de um arquivo que é persistido em sessões (**Preferences**).
- Os **Microservices** que são consumidos pelo APP são desenvolvidos na linguagem **Java** com SpringBoot. Estas aplicações são empacotadas em containers para execução em cluster **Kurbenetes** (gerido pelo Red Hat OpenShift).
- Os bancos de dados para os serviços de gestão de identidade e login do usuário utilizados são **MYSQL e PostgreSQL** que são contratados via serviços (RDS) no ambiente AWS.
- É utilizado o **ElasticSearch** alocado na AWS como plataforma para armazenamento das *push notifications* geradas pelo Notification Center.



Arranjo arquitetural

Integrações

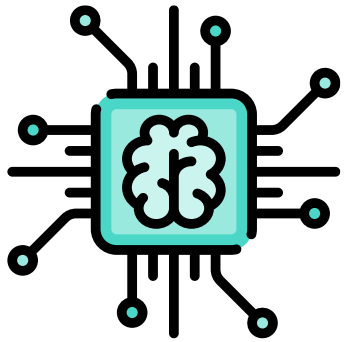
- A **integração com os serviços é feita via API** e conta com troca de arquivos JSON. Há uma camada de abstração, via SDK, para cada parceiro fornecedor integrado à plataforma.
- A integração com os serviços da VR é feito com **três provedores de serviços** diferentes:
  - API Gateway da Sensidia;
  - Barramento IBM Websphere;
  - 4Time WAS.
- Para comunicação via **notificações** é utilizado um broker de mensagens, através do **Amazon MQ for Rabbit MQ,** conectado ao serviço Notification Center, responsável pelo envio de mensagens transacionais e relacionais.
- Integração com **Plataforma Google Maps** para disponibilizar o recurso de adicionar marcadores, utilizado para facilitar a trajetória até os estabelecimentos da rede credenciada.



Arranjo arquitetural

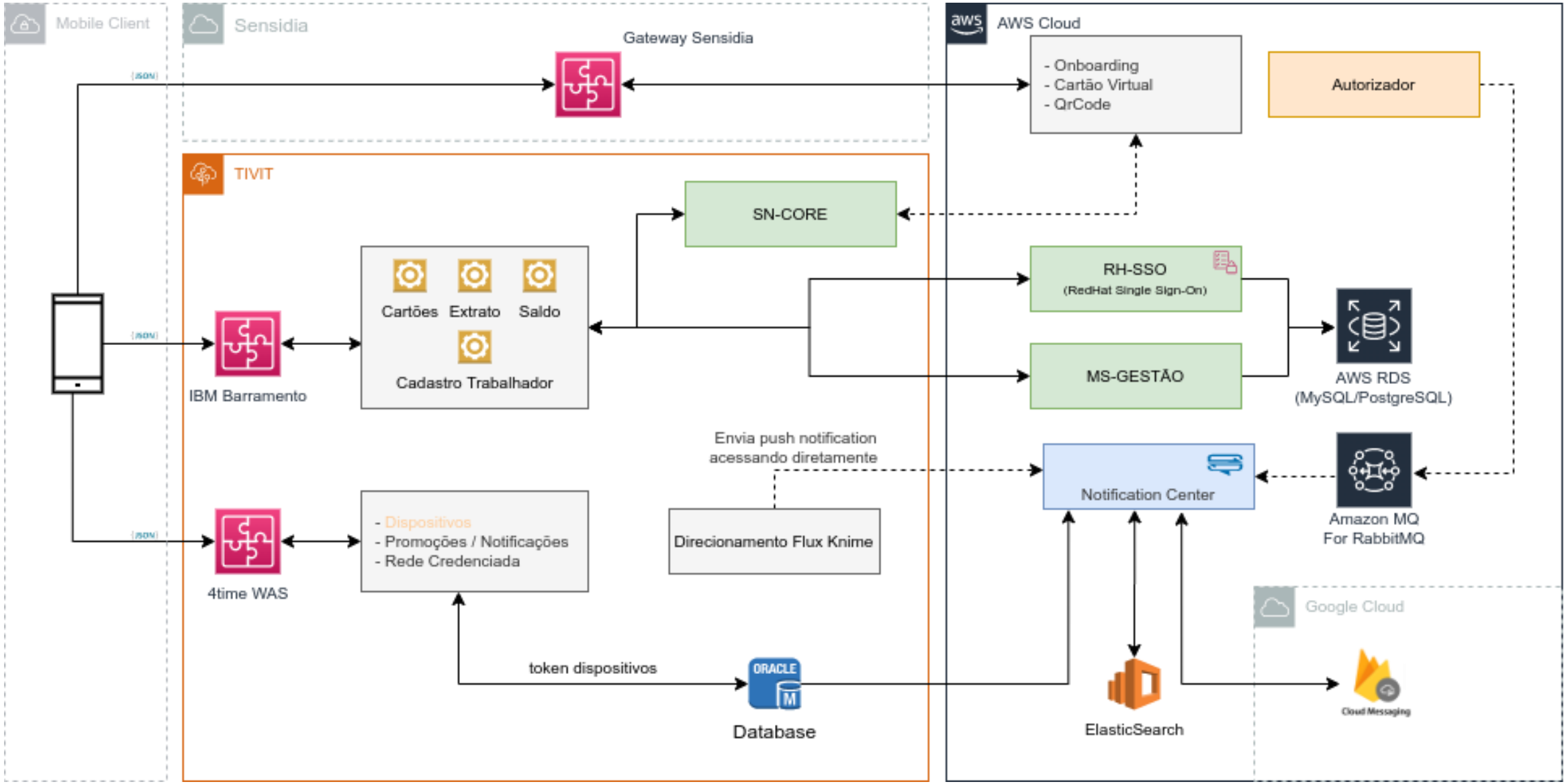
Segurança

- As informação de senha de acesso dos usuários são armazenadas de forma criptografada através de um **algoritmo simétrico**.
- O Aplicativo não utiliza-se de **recursos de ofuscação** para proporcionar redução do tamanho do APP e ajudar contra ataques de engenharia reversa, principalmente na versão Android.
- As informações de autenticação são trafegadas para o barramento IBM via JSON utilizando-se **protocolo https**.
- Não existe atualmente um **mecanismo de positivação** via SMS ou checagem do CPF a partir de um birô de crédito.
- Como mecanismo de autenticação do APP é utilizada a solução **Red Hat Single Sign-On (RH-SSO)**, baseado no padrão OAuth 2.0.
- A segurança da comunicação entre as redes Cloud AWS e TIVIT é garantida por uma ***VPN site-to-site***, estabelecida entre os ambientes. Este mecanismo garante que as duas redes possam se comunicar internamente através de acesso entre si.



CENÁRIO ATUAL

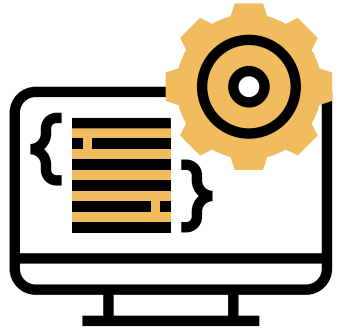
Blue Print – Desenho do arranjo arquitetural



Aspectos da Implementação

CI / CD / Build & Deploy

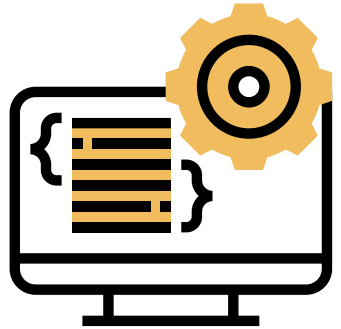
- Há uma iniciativa em fase de testes de **pipeline CI/CD** tanto para iOS quanto Android utilizando ferramenta **SaaS (Bitrise) na nuvem.**
- Em paralelo, também está sendo analisada opção de criação de ambiente na própria empresa para a execução de pipeline CI/CD.
- Apesar dessa iniciativa, atualmente, os processos relacionados a *Integration* e *Delivery* dos aplicativos (tanto iOS quanto Android) estão sendo disparados manualmente nas próprias máquinas dos desenvolvedores.
- Há um pipeline consolidado e específico para **testes de integração**. Esse pipeline é executado diariamente em horário específico (ou também sob demanda) e executa uma suíte de testes de sanidade escrita em Ruby.
    - **Testes de regressão** não fazem parte dessa pipeline executada diariamente, mas podem vir a ser executados sob demanda



Aspectos da Implementação

Testes

- A cobertura de **testes unitários** é ainda incipiente para os apps (tanto iOS quanto Android).
- Para os **testes de integração**, já há um conjunto significativo de testes automatizados executados diariamente. Os scripts desses testes são codificados em Ruby e estão em um repositório apartado do repositório do código fonte dos apps.
- Em relação aos **testes sistêmicos**, os critérios de aceite das histórias são especificados em linguagem Gherkin. Em um exemplo visto, os termos utilizados estão mais voltados para a parte tecnológica do que para o negócio em si.
- **BDD (Behaviour Driven Development)**, apesar do uso de especificação dos critérios de aceite e testes de aceitação em Gherkin, as práticas e técnicas relacionadas ao BDD ainda estão sendo consideradas e não há consenso se devem ser adotadas na empresa
- Os **critérios de aceite** são a base para os testes funcionais de aceitação, porém a especificação dos cenários (com exemplos concretos de uso) só é feita quando há requisição do time de desenvolvimento.
- Observamos que há iniciativa de **automação de testes** sistêmicos funcionais com utilização de stack com Appium, mas essa iniciativa está atualmente pausada.
- Atualmente, os dispositivos onde são testados os apps se resumem aos dispositivos físicos do próprio time de testers. Há em curso um estudo de uso de **serviços de teste** com "device farms" utilizando providers como AWS e Google Firebase.
- É utilizado o plugin Zephyr Scale no JIRA que **gerencia os ciclos e suítes de testes** e, para testes automatizados (com execução via Jenkins), atualiza também status dos casos de testes definidos.
- Para casos específicos (sob demanda), é construída suíte de **testes de performance**, utilizando jMeter, executados em ambiente de desenvolvimento e homologação.



SEE YOU SOON!