


	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	


## Arquitetura Referência de Mobilidade



VITEC/DETEC/GEARQ

Minuta 6.3

#05


	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

## Histórico da Revisão


Data	Versão	Descrição	Autor
15/08/2017	1.0	Criação do documento	GEARQ01
12/09/2017	1.1	Revisão	GEARQ01
22/09/2017	1.2	Revisão	GEARQ01
19/10/2017	6.3	Revisão	GEARQ01

## SUMÁRIO

<b>1. GLOSSÁRIO</b>	4
<b>2. OBJETIVO DO DOCUMENTO</b>	5
<b>3. ARQUITETURA DE MOBILIDADE</b>	5
<b>3.1. FRONT-END</b>	7
<b>3.1.1. Tipos de Aplicação - APP</b>	8
<b>3.1.1.1. Aplicação Web Responsiva</b>	10
<b>3.1.1.2. Aplicação Web App</b>	10
<b>3.1.1.3. Aplicação Híbrida</b>	11
<b>3.1.1.4. Aplicação Nativo Híbrido</b>	14
<b>3.1.1.5. Aplicação Nativa</b>	15
<b>3.1.2. Critérios para Definições de Desenvolvimento</b>	18
<b>3.1.3. Plataformas e Ferramentas de Desenvolvimento</b>	21
<b>3.1.3.1. Android</b>	21
<b>3.1.3.2. iOS</b>	22
<b>3.1.3.3. Tecnologia Cross-plataform</b>	23
<b>3.1.3.3.1. Cordova</b>	23
<b>3.1.3.3.2. Xamarin</b>	24
<b>3.1.3.3.3. Appcelerator Titanium</b>	24
<b>3.1.4. Identificação do Dispositivo</b>	25
<b>3.1.4.1. Padrão de geração de ID de dispositivo</b>	25
<b>3.1.5. Criticidade dos Aplicativos</b>	26
<b>3.1.6. Diretrizes para o Front-end</b>	26
<b>3.2. MIDDLEWARE</b>	28
<b>3.2.1. Proxy Mobile</b>	29
<b>3.2.2. Servidor de Autenticação Corporativo</b>	30


	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

<b>3.2.2.1. RedHat Single Sign-on .....</b>	<b>31</b>
<b>3.2.3. Vinculação de Dispositivo .....</b>	<b>32</b>
<b>3.2.4. Middleware de Mobilidade .....</b>	<b>34</b>
<b>3.2.4.1. CA Mobile API Gateway (IDMS Orchestrator) .....</b>	<b>35</b>
<b>3.2.4.2. IBM Mobile First .....</b>	<b>36</b>
<b>3.2.5. API Manager .....</b>	<b>38</b>
<b>3.2.6. Analytics e Monitoring .....</b>	<b>38</b>
<b>3.2.7. Diretrizes para Middleware .....</b>	<b>39</b>
<b>3.3. BACK-END .....</b>	<b>40</b>
<b>3.3.1. Sistemas Auxiliares no Ecosistema de Mobilidade da CAIXA .....</b>	<b>41</b>
<b>3.3.2. Diretrizes para BACKEND .....</b>	<b>42</b>
<b>4. DECOMPOSIÇÃO ARQUITETURAL .....</b>	<b>43</b>
<b>4.1. CENARIO 1: SEM MIDDLEWARE DE MOBILIDADE .....</b>	<b>43</b>
<b>4.1.1. Vantagens .....</b>	<b>45</b>
<b>4.1.2. Desvantagens .....</b>	<b>45</b>
<b>4.1.3. Indicação de USO .....</b>	<b>45</b>
<b>4.2. CENARIO 2: COM MIDDLEWARE DE MOBILIDADE – IBM MOBILE FIRST .....</b>	<b>45</b>
<b>4.2.1. Vantagens .....</b>	<b>47</b>
<b>4.2.2. Desvantagens .....</b>	<b>48</b>
<b>4.2.3. Indicação de USO .....</b>	<b>48</b>
<b>4.3. CENARIO 3: COM MIDDLEWARE DE MOBILIDADE – CA MOBILE API GATEWAY ..</b>	<b>48</b>
<b>4.3.1. Vantagens .....</b>	<b>50</b>
<b>4.3.2. Desvantagens .....</b>	<b>50</b>
<b>4.3.3. Indicação de USO .....</b>	<b>51</b>

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

## 1. GLOSSÁRIO

<b>Termo</b>	<b>Descrição</b>
Back-End	É o nível de programação que encapsula os sistemas legados da Caixa e as respectivas bases de dados.
CSS	<i>Cascade Style Sheet</i> – Folha de estilos utilizada na estilização de páginas da internet;
Front-End	Interfaces que permitem abstração das camadas subjacentes da aplicação e interage diretamente com usuário.
GCM	Disciplina de Gerência de Configuração e Mudanças
IAM	Do inglês Identity and Access Management
JSON	<i>JavaScript Object Notation</i> é uma formatação de troca de dados, amigável para leitura e escrita, além de ser fácil para conversão e geração.
POC	Prova de Conceito – Do Inglês <i>Prove Of Concept</i>
REST	<i>Representational State Transfer</i> é a abstração da arquitetura WWW, que objetiva a definição de características fundamentais para construção de aplicações Web no contexto das melhores práticas.
SPA	<i>Single Page Application</i> – Modelo de desenvolvimento onde todo o conteúdo visual está definido em uma página única e partes dessa página são ocultadas e exibidas de acordo com o conjunto de dados que se deseja exibir. Neste modelo, todo o recurso estático como os arquivos HTML, JavaScript, CSS e imagens necessários são transferidos para o cliente no primeiro acesso e a partir daí, apenas dados são solicitados ao servidor.
Stateless	Protocolo sem estado - É um protocolo de comunicação que considera cada requisição como uma transação independente que não está relacionada a qualquer requisição anterior, de forma que a comunicação consista de pares de requisição e resposta independentes.
Wearable	É a palavra que resume o conceito das chamadas “tecnologias vestíveis”, que consistem em dispositivos tecnológicos que podem ser utilizados pelos usuários como peças do vestuário.
WLAPP	Pacote denominado “ <i>Work Light Application</i> ” onde ficam armazenados os recursos estáticos como HTM, CSS e JavaScript.
IAM	Identity Access Management (Gerenciador de Identidade e Acesso)

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

## 2. OBJETIVO DO DOCUMENTO

O objetivo deste documento é definir as arquiteturas referências de mobilidade para a CAIXA, padrões e tecnologias a serem utilizados. O documento também deve servir como um guia para direcionar tomada de decisões sobre quando utilizar cada arquétipo arquitetural e a necessidade de novo APP com base nas estruturas empresariais que suportam a atendimento a demandas de mobilidade.

## 3. ARQUITETURA DE MOBILIDADE

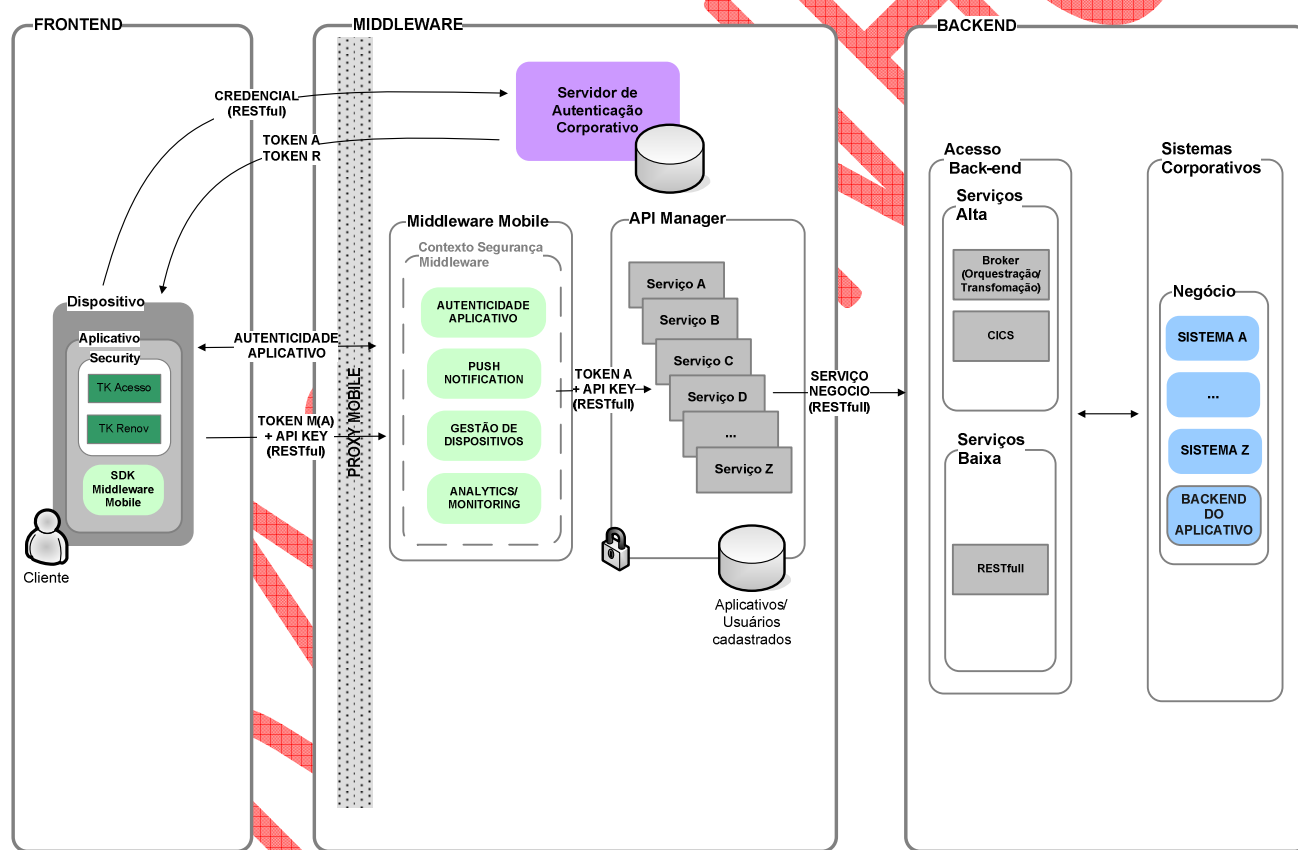



Figura 1: Representação da Arquitetura de Referência para Mobile

É importante entender que no universo da mobilidade, a estrutura de front-end (cliente) é em geral independente da estrutura do back-end (servidor). Vale ressaltar que genericamente, um APP mobile, nada mais é que uma "casca" que tem por finalidade


	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

consumir serviços que estão expostos na rede e de uma maneira geral estes serviços são consumidos através de um padrão Web onde os dados repassados são uma representação de estado, conhecido como “Transferência de Estado Representacional – REST” e estas respostas são retornadas no formato JSON (Java Script Object Notation).

Analizando pura e simplesmente o atendimento à regra de negócio (consumo do serviço), podemos fazer um paralelo de um APP com uma aplicação SPA (Single Page Application) em que todo o conteúdo visual encontra-se do lado cliente (Navegador ou Dispositivo) e os conteúdos são consumidos através das chamadas Web. Diferenciam-se essas abordagens pelo fato do navegador ser passivo (onde o usuário precisa informar o endereço para ativar a navegação) e o APP ter uma característica ativa (onde ao abrir o APP uma requisição inicial pode ser disparada automaticamente para o servidor em busca dos primeiros dados). Além disso, os recursos disponíveis do lado cliente propiciam uma experiência de uso diferente, pois o dispositivo possui recursos de hardware adicionais (Ex: GPS, Bussola, etc). Enquanto o navegador necessita utilizar de artifícios como localizar posicionamento através do IP de requisição, as APPs contam com alguns recursos de identificação providos pelo fornecedor do Sistema Operacional através das suas nuvens próprias.

A comunicação entre o cliente e o servidor, por padrão é sempre feita com um protocolo sem estado (do inglês stateless) que considera cada requisição como uma transação independente e que não está relacionada a qualquer requisição anterior, de forma que a comunicação consista de pares de requisição e resposta independentes.

Sendo assim, pode-se entender que mais importante do que definir uma estrutura de construção/funcionamento do APP, é identificar como e onde os serviços que serão consumidos estão disponibilizados. Nos itens a seguir detalharemos cada uma das camadas dispostas na representação arquitetural.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

### 3.1. FRONT-END

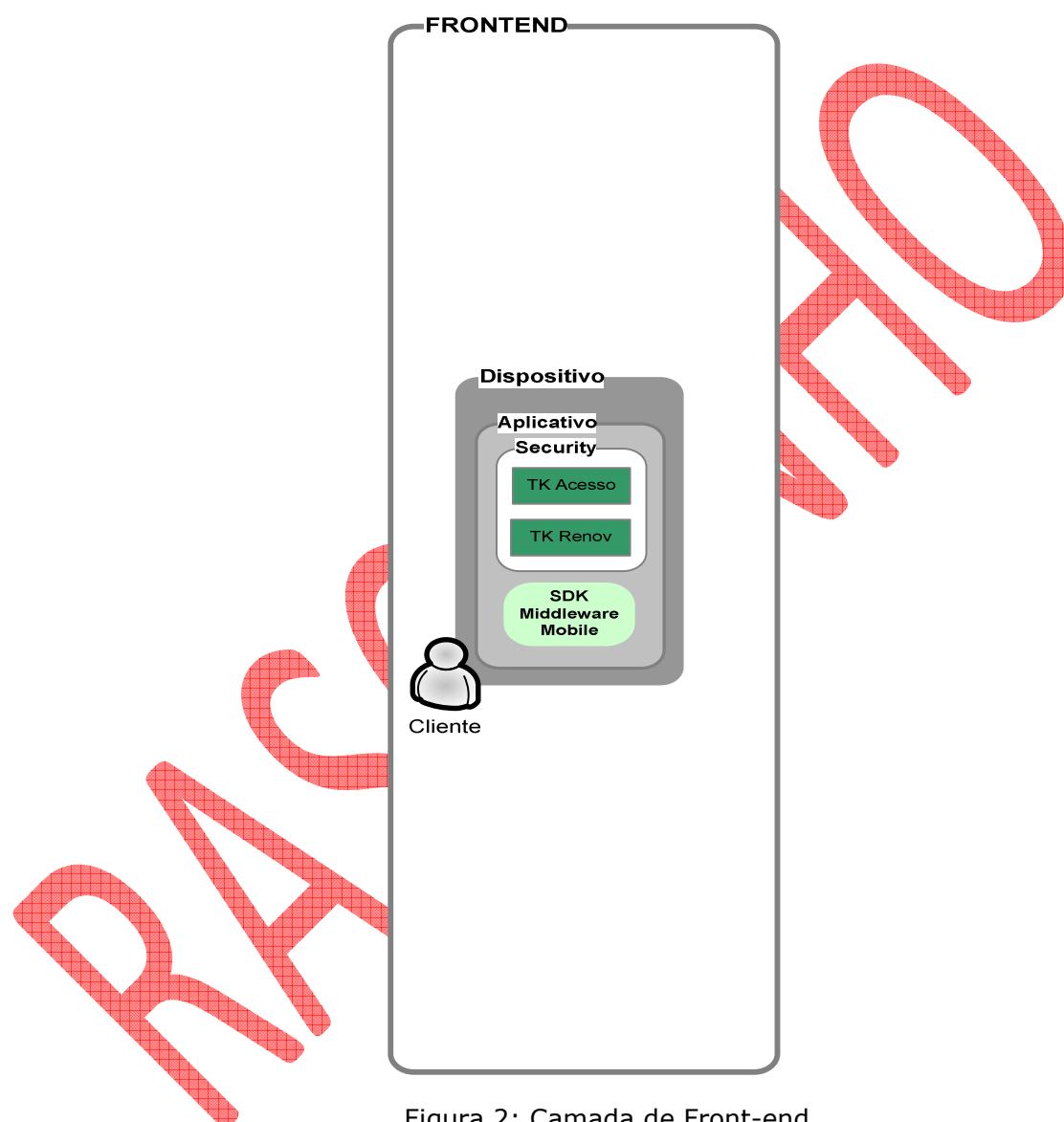



Figura 2: Camada de Front-end

Esta camada é basicamente constituída pelo aplicativo em si, o qual traz embarcado em seu código componentes de software (SDKs e/ou APIs), responsáveis por prover sua interação com os hardwares do dispositivo (câmera, GPS, acelerômetro, NFC, etc),

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

recursos específicos da plataforma (armazenamento seguro de *tokens*), bem como acesso a funcionalidades providas pelos *middlewares* de mobilidade <sup>1</sup>adotados pela CAIXA.

No diagrama abaixo estão representados alguns dos conceitos e componentes chaves que constituem a camada de front-end de mobilidade, abstraindo-se neste momento questões relacionadas à natureza de constituição e construção do aplicativo (híbrido, nativo, nativo-híbrido, webview, etc).

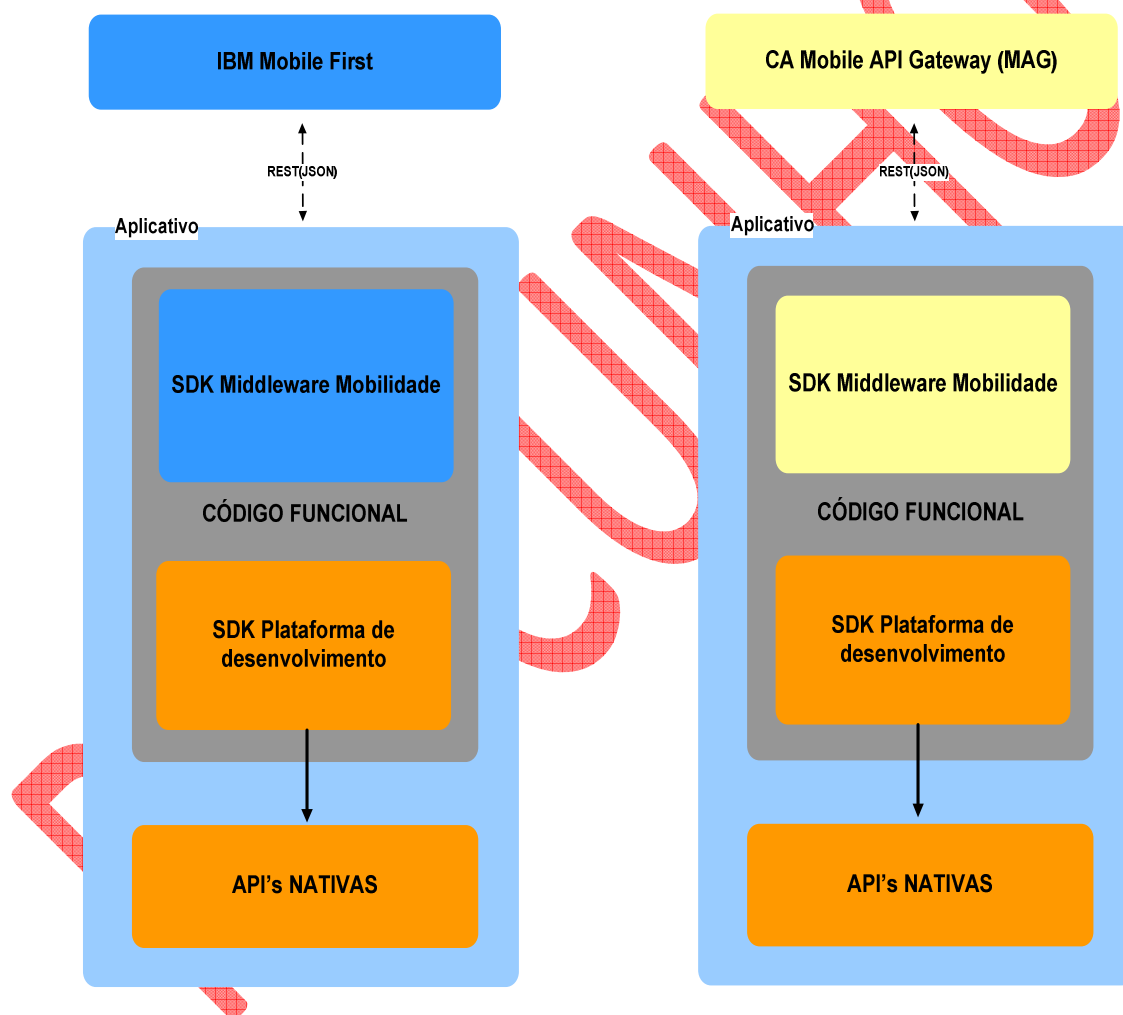



Figura 3: Camada *Front-end* – Elementos conceituais

### 3.1.1. Tipos de Aplicação - APP

<sup>1</sup> O *middleware* de mobilidade é o componente responsável por recepcionar as requisições dos aplicativos



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Há atualmente uma grande variedade de estratégias e tecnologias disponíveis para o desenvolvimento de aplicativos móveis. As opções vão desde o desenvolvimento em plataformas nativas viabilizadas por SDKs específicas de cada sistema operacional, até o desenvolvimento totalmente desvinculado de qualquer sistema operacional e que emprega o uso de tecnologias de desenvolvimento web (HTML, CSS e JavaScript) no seu core funcional e frameworks específicos para garantir a portabilidade entre várias plataformas móveis.

Para fins de convenção, consideraremos aplicativos móveis na CAIXA somente as aplicações instaláveis no dispositivo por meio de pacotes específicos a cada plataforma de mobilidade. Sendo assim, aplicações desenvolvidas totalmente com tecnologias web, acessíveis somente por meio do browser não são consideradas aplicativos móveis e sim webapps ou simplesmente sites responsivos.

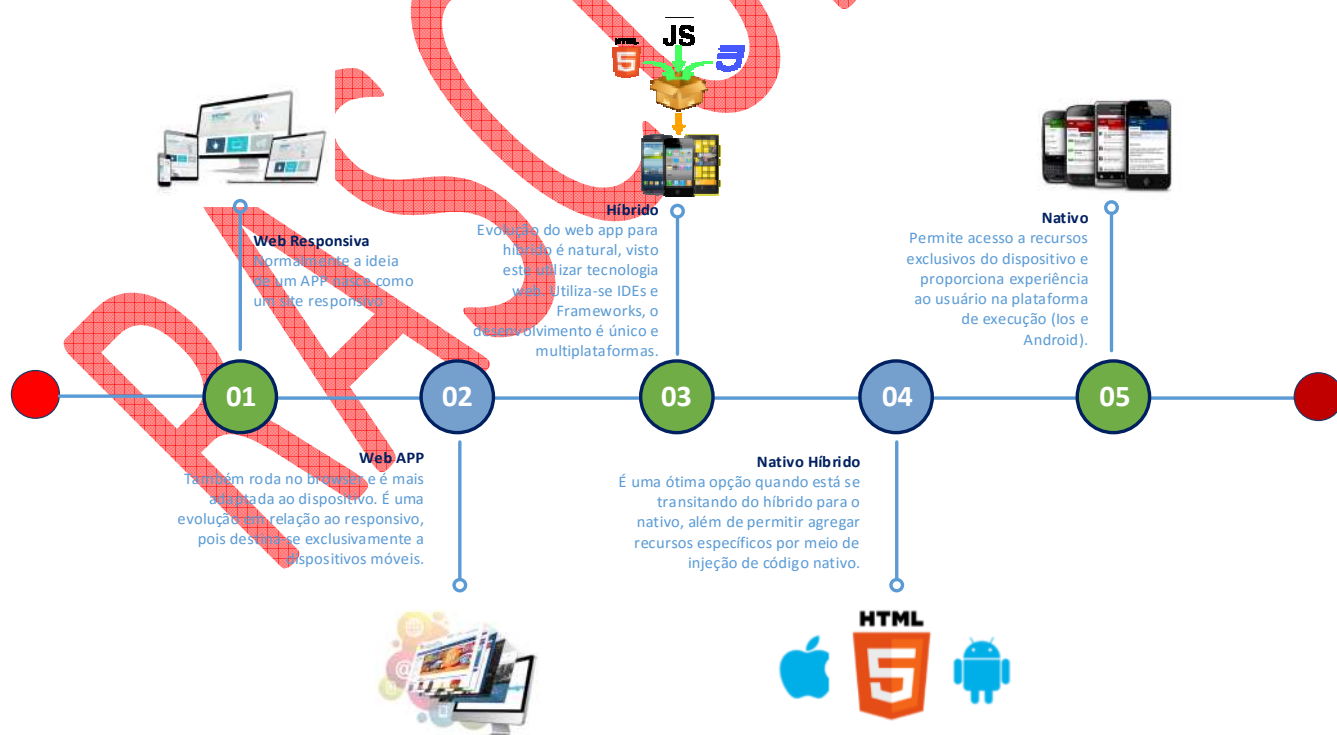



Figura 4: Tipos de Aplicações


	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

### 3.1.1.1. Aplicação Web Responsiva

Definição	É um site que tem o layout preparado para se adaptar ao formato de diversos dispositivos dentre os quais estão o tablet e/ou smartphone, porém não é um App. Quando o visitante visualiza o site no computador, o formato se expande e aproveita toda a tela. No dispositivo, as informações mudam de posição, mas preserva-se o conteúdo.
Vantagens	<ul style="list-style-type: none"> <li>• Renderização em qualquer dispositivo com acesso ao server;</li> <li>• Possibilidade de classificar em mecanismos de busca (SEO);</li> <li>• Reduz o custo de manutenção;</li> <li>• Reduz o esforço de atualização nos clientes pois o conteúdo passa a estar no server;</li> <li>• A atualização é centralizada;</li> </ul>
Desvantagens	<ul style="list-style-type: none"> <li>• O desenvolvimento do design é mais custoso, pois é necessário adaptar a diversos tamanhos de dispositivos;</li> <li>• Baixo acesso a recursos dos dispositivos;</li> <li>• Interface gráfica dos componentes divergente dos padrões da plataforma;</li> <li>• Experiência do usuário comprometida;</li> <li>• Tempo de implementação aumenta de acordo com a quantidade e complexidade dos elementos em tela;</li> <li>• Necessita de alta experiência de web design;</li> <li>• Desempenho dependente da largura de banda do usuário;</li> </ul>
Recomendações	<p>Recomenda-se a utilização de um cliente Web Responsivo se:</p> <ul style="list-style-type: none"> <li>• Não se necessita acessar recursos dos dispositivos;</li> <li>• A solução envolve a construção de portal web para exibir as mesmas informações através de um navegador na intranet/internet;</li> <li>• Utilizado para situações informativas;</li> <li>• A solução tratar-se efetivamente de um site;</li> <li>• Ex: <a href="https://sifec.caixa.gov.br/">https://sifec.caixa.gov.br/</a></li> </ul>

### 3.1.1.2. Aplicação Web App

Definição	É uma aplicação Internet acessada via web <i>browser</i> , sendo exclusivamente para dispositivos móveis. Programaticamente reconhece o acesso via smartphone, por exemplo, e se adapta ao dispositivo. Utilizada quando pretende-se apenas apresentar conteúdo, não consome memória do dispositivo, não armazena conteúdo off-line e não está disponível nas lojas.
-----------	--

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Vantagens	<ul style="list-style-type: none"> <li>• Não consome memória do dispositivo;</li> <li>• Costuma rodar satisfatoriamente em várias plataformas;</li> <li>• Menor custo de desenvolvimento;</li> <li>• Roda em vários navegadores.</li> </ul>
Desvantagens	<ul style="list-style-type: none"> <li>• Não é um aplicativo de fato, apenas a página inicial está no dispositivo;</li> <li>• Não permite controle/acesso aos recursos dispositivo;</li> <li>• A usabilidade não é a melhor possível;</li> <li>• Necessário conhecer a URL;</li> </ul>
Recomendações	<ul style="list-style-type: none"> <li>• É recomendada para usuário eventual, uma vez que não precisa de instalação.</li> <li>• Solução meramente informacional, sem a necessidade de utilizar recursos do dispositivo</li> <li>• Ex: <a href="https://m.facebook.com">https://m.facebook.com</a> e <a href="https://mobile.twitter.com">https://mobile.twitter.com</a></li> </ul>

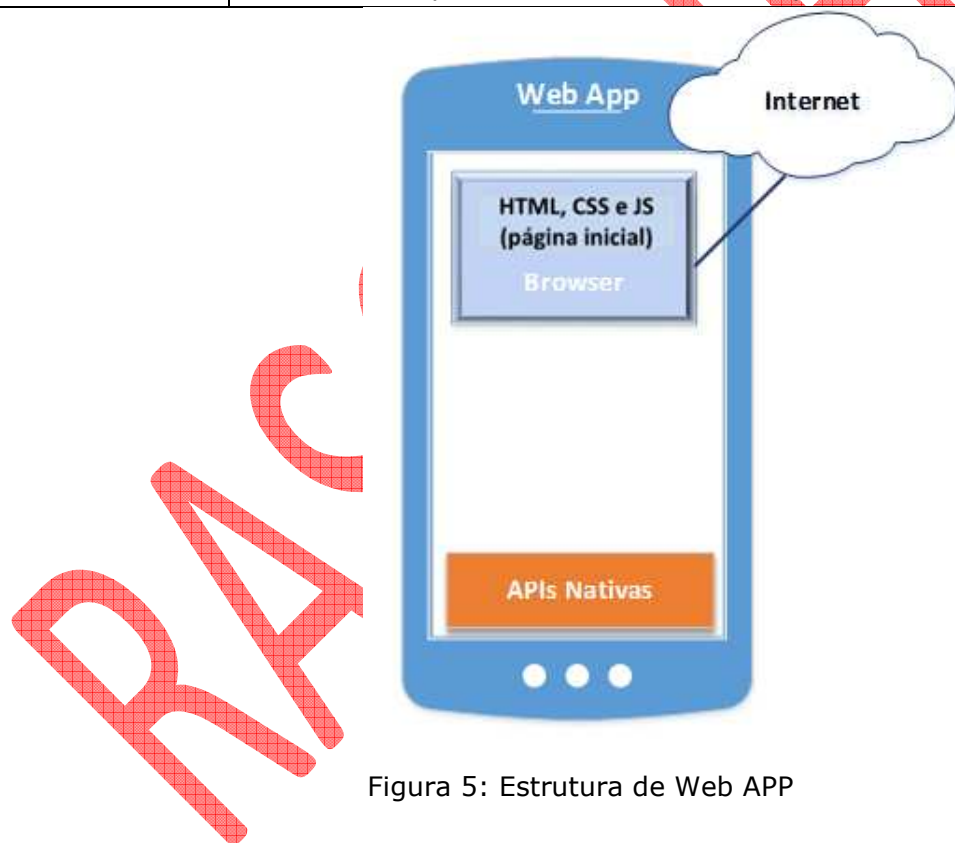




Figura 5: Estrutura de Web APP

### 3.1.1.3. Aplicação Híbrida

Definição	Um aplicativo híbrido e/ou cross-platform é aquele que possui parte considerável (ou todo ele) feita em tecnologia não específica da plataforma. O Apache Cordova é um framework de desenvolvimento móvel de código aberto. Ele permite que você use tecnologias web padrão - HTML5, CSS3
-----------	---

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

	e <i>JavaScript</i> para desenvolvimento Multiplataforma. Os aplicativos são executados em <i>wrappers</i> segmentados para cada plataforma e dependem de APIs compatíveis com padrões para acessar as capacidades de cada dispositivo, como sensores, dados, status da rede, etc.
Vantagens	<ul style="list-style-type: none"> <li>• Desenvolvimento único com compilação;</li> <li>• Tendência a gastar um tempo menor com desenvolvimento;</li> <li>• Desenvolvimento baseado em estrutura de WEB e compilação em um APP para diversas plataformas com acesso a recursos do dispositivo;</li> <li>• Aproveitamento do conhecimento da equipe em desenvolvimento WEB;</li> <li>• Grande diversidade de plug-ins disponíveis, incluindo acesso a recursos dos dispositivos;</li> <li>• Ferramental open-source para desenvolvimento;</li> </ul>
Desvantagens	<ul style="list-style-type: none"> <li>• Performance um pouco mais lenta por existir uma camada de tradução;</li> <li>• Compatibilidade depende do motor de renderização do navegador da plataforma;</li> <li>• Maior esforço de testes para validar a estrutura em todos os dispositivos e versões de sistemas operacionais desejados;</li> <li>• Acesso a recursos do dispositivo dependem de plug-ins e portanto alguns destes recursos e APIs nativas ainda não são suportados pelo Cordova e pode impactar a utilização de novos recursos a serem lançados;</li> <li>• Interface gráfica padronizada pelo CSS, geralmente divergindo dos modelos específicos das plataformas;</li> </ul>
Recomendações	<p>Recomenda-se a construção de um cliente híbrido se:</p> <ul style="list-style-type: none"> <li>• A solução não necessitar de características muito específicas da plataforma e/ou recursos de hardware muito específicos;</li> <li>• Todos os recursos de hardware necessários possuem plug-ins disponíveis e estiverem previamente validados através de POCs;</li> <li>• A equipe de desenvolvimento possuir domínio nas linguagens de programação para WEB (HTML5, CSS e JavaScript) de forma a reduzir a curva de aprendizado e tempo de desenvolvimento;</li> <li>• A padronização de interface gráfica entre as plataformas não for um problema;</li> <li>• Ex: Netflix</li> </ul>

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

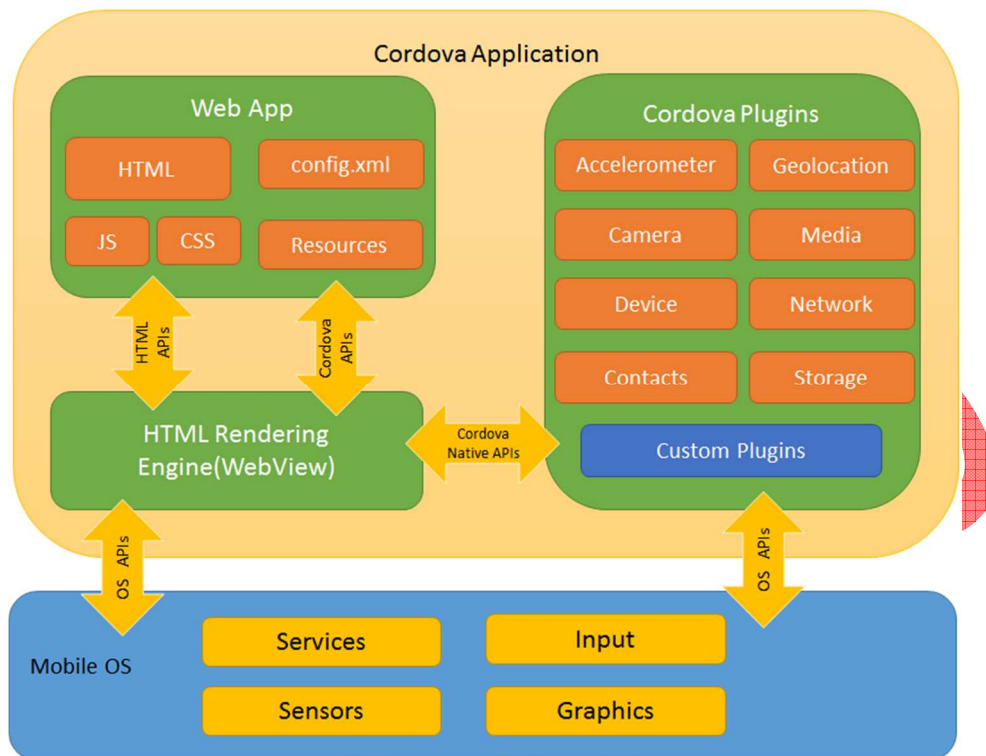


Figura 6: Arquitetura do Cordova

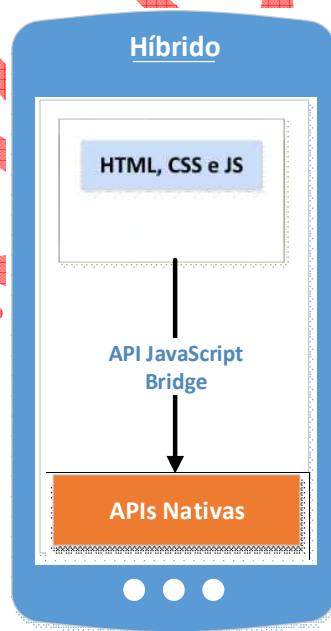




Figura 7: Estrutura de aplicativo híbrido

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

#### 3.1.1.4. Aplicação Nativo Híbrido

Definição	App Nativo Híbrido tem parte do código personalizado, compilada nativamente e executado diretamente no sistema operacional móvel. Adicionalmente tem parte do código personalizado executado dentro do controle nativo do WebView da plataforma móvel ou como um aplicativo habilitado para JavaScript que funciona no sistema ou no "navegador seguro". A parte nativa pode acessar recursos nativos diretamente através das APIs, enquanto o container Web executado dentro do controle WebView usa API de ponte JavaScript
Vantagens	<ul style="list-style-type: none"> <li>• Suporte a todas as funcionalidades oferecidas pela plataforma, tanto em software quanto em hardware;</li> <li>• Compatibilidade com componentes de interface gráfica;</li> <li>• Melhor desempenho da solução;</li> <li>• Possibilidade de desenvolvimento para <i>Wearables</i>;</li> <li>• Compartilhamento de partes da aplicação entre projetos de plataformas distintas;</li> </ul>
Desvantagens	<ul style="list-style-type: none"> <li>• Necessidade de maior experiência na GCM;</li> <li>• Projetos independentes para as plataformas que se desejar ofertar a solução;</li> <li>• Necessidade de conhecimento aprofundado nas linguagens de cada uma das plataformas;</li> <li>• Riscos de evoluir funcionalidades de forma desordenada nas plataformas;</li> <li>• Custo de desenvolvimento para construção e inclusão de novas funcionalidades multiplicado por tantas quantas plataformas forem suportadas, além do desenvolvimento com o uso do framework;</li> </ul>
Recomendações	<p>Recomenda-se a construção de um cliente nativo híbrido se:</p> <ul style="list-style-type: none"> <li>• Soluções com ciclos longos de evolução (pois a manutenção e sincronização dos códigos de cada parte considerada é mais complexa).</li> <li>• A solução necessite primordialmente de alto desempenho e com restrições orçamentárias de projeto;</li> <li>• Deseje que algumas partes essenciais da interface visual possa ser adequada a cada plataforma de forma mais fluída;</li> <li>• Necessidade de utilização de recursos não acessíveis na plataforma híbrida. Ex: Integração com o assistente pessoal.</li> <li>• Projetos que estão passando por uma transição da plataforma tecnológica utilizada no seu desenvolvimento (Ex: projeto originalmente híbrido, sendo migrado para Nativo e vice-versa)</li> <li>• Ex: Agência Virtual CAIXA</li> </ul>

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

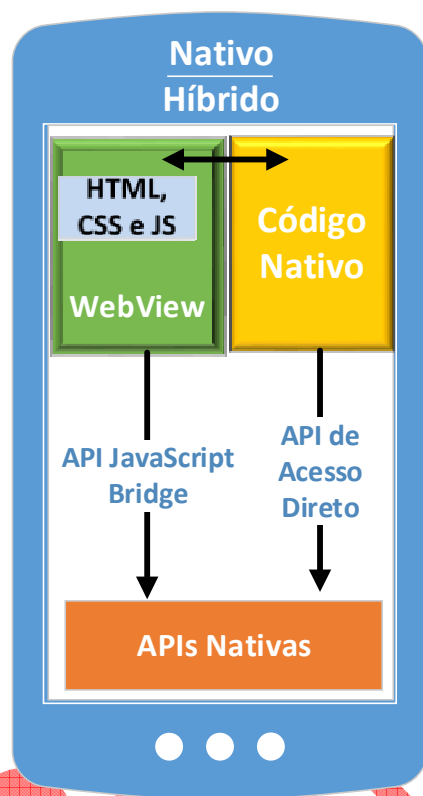



Figura 8: Estrutura de APP Nativo Híbrido

#### 3.1.1.5. Aplicação Nativa

Definição	É um aplicativo desenvolvido para plataforma específica, utilizando todo o potencial de funcionalidades nativas para a qual ele foi feito. É disponibilizado nas lojas da Google e Apple, sendo programado em Java para Android e Swift para iOS.
Vantagens	<ul style="list-style-type: none"> <li>• Suporte total a todas as funcionalidades oferecidas pela plataforma, tanto em software quanto em hardware;</li> <li>• Total compatibilidade com componentes de interface gráfica;</li> <li>• Melhor desempenho da solução;</li> <li>• Independência de manutenção e testes;</li> <li>• Maior facilidade para evoluir funcionalidades em consonância plataforma;</li> <li>• Possibilidade de desenvolvimento para <i>Wearables</i>;</li> </ul>
Desvantagens	<ul style="list-style-type: none"> <li>• Necessidade de maior experiência na Gestão de Configurações e Mudança;</li> </ul>



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

	<ul style="list-style-type: none"> <li>• Projetos independentes para todas as plataformas que se desejar ofertar a solução;</li> <li>• Necessidade de conhecimento aprofundado nas linguagens de cada uma das plataformas;</li> <li>• Riscos de evoluir funcionalidades de forma desordenada nas plataformas;</li> <li>• Custo de desenvolvimento para construção e inclusão de novas funcionalidades multiplicado por tantas quantas plataformas forem suportadas;</li> <li>• Ferramental específico de cada plataforma para desenvolvimento;</li> </ul>
Recomendações	<p>Recomenda-se a construção de um cliente nativo se:</p> <ul style="list-style-type: none"> <li>• A solução necessite primordialmente de alto desempenho;</li> <li>• Deseje garantir que seja possível utilizar os recursos mais novos dos dispositivos, logo que lançados (independente da característica existir em apenas uma das plataformas)</li> <li>• Deseje que a interface visual possa ser adequada a cada plataforma de forma mais fluída;</li> <li>• A aplicação utilize recursos muito específicos de cada plataforma. Ex: Integração com o assistente pessoal, etc.</li> <li>• Ex: WhatsApp, Instagram, Waze e Uber.</li> </ul>

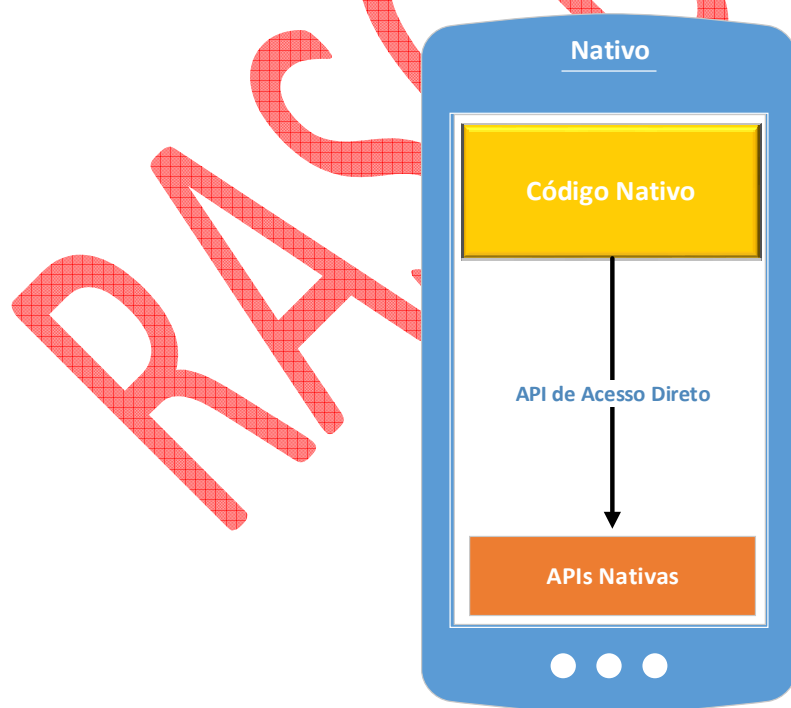


Figura 9: Estrutura de APP Nativo




	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	


Tabela comparativa de arquitetura de APP			
Perfil Arquitetural	Nativo	Nativo Híbrido	Híbrido
Linguagem	Nativo (Swift e Java)	JavaScript e Nativo (Swift e Java)	JavaScript, HTML e CSS
Acesso a API Nativa	API de Acesso Direto	API JS Bridge e API de Acesso Direto	API JS Bridge
Portabilidade	Não portátil	Código JS e Híbrido: Portável Código Nativo: Não	Portável
Complexidade	Média	Alta	Média
Interface de Usuário Nativa	Sim	Código JS e Híbrido: Não Código Nativo: Sim	Não
Características Nativas	Sim, todas acessíveis	JS e Híbrido: Sim, várias acessíveis Código Nativo: Sim, todas acessíveis	JS e Híbrido: Sim, várias acessíveis
Opções de Deploy	Publicável nas lojas ou estático	Publicável nas lojas ou estático	Publicável nas lojas ou estático

Tabela 1: Comparação de perfis de arquitetura de aplicativos

Conforme apresentado na figura acima o framework e as ferramentas a serem utilizadas na construção de um App podem variar de acordo com a arquitetura adotada.

**Na Arquitetura de App Nativo** utiliza-se SDKs nativos para a construção, teste, compilação e publicação. Adicionalmente pode-se utilizar um cross-compiler (Xamarin) para se construir o código nativo, porém o SDK ainda será necessário para testar e publicar a aplicação.

**Na Arquitetura de App Nativo Híbrido** deve-se utilizar o cross-compiler ou SDKs nativos para construir as funcionalidades nativas do app, porém será necessária a utilização de framework híbrido baseado em HTML5, CSS e JavaScript para geração do código Web que rodará no Webview gerenciado pelo aplicativo. Vale ressaltar que a parte nativa do App pode lançar mão do SDK nativo para realização dos testes necessários.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	


**Na Arquitetura de App Híbrido** recomenda-se a utilização de framework híbrido para gerar o wrap do App, como Cordova, visto este prover a JavaScript API bridge para acesso aos recursos nativos por parte do código Web (HTML, CSS e JavaScript) gerado. Nessa arquitetura não se lança mão dos SDKs nativos para fase de construção, sendo que não se fará uso de APIs nativas. Essa arquitetura é utilizada quando se pensa em um modelo de negócio que necessita de desenvolvimento otimizado do aplicativo, pretende-se entregar ao cliente boa experiência de usuário (UX), os Apps demandam celeridade na construção, além de rodarem em várias plataformas. Outra relevância é a padronização da arquitetura, ferramentas e frameworks de desenvolvimento dos Apps, dessa forma pretende-se consolidar as melhores práticas na criação destes.

### 3.1.2. Critérios para Definições de Desenvolvimento

Existe uma grande discussão a respeito de qual estratégia é a mais adequada para se desenvolver a camada de *front-end* de um aplicativo móvel. Ter ciência das características, vantagens e desvantagens inerentes de cada uma delas é essencial na escolha da opção mais adequada a um determinado contexto.

O processo decisório envolverá basicamente o confronto destas características com o seu nível de relevância para cada um dos seguintes fatores, essenciais na concepção do aplicativo a ser construído:

- Interface Visual;
- Abordagem Orientada a Plataforma;
- Desempenho;
- Recursos utilizados do dispositivo;
- Características dos usuários alvo;
- Complexidade das funcionalidades;
- Tempo disponível para desenvolvimento;
- Custo para construção/manutenção da solução.


	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Para auxiliar nesta definição foi desenvolvida a matriz de decisão que orienta os idealizadores/desenvolvedores do aplicativo na identificação e avaliação de cada um dos requisitos, conceitos e estratégias de uso acima discutidos, buscando promover desta forma uma convergência para a opção mais indicada a ser aplicada em um dado contexto.

Estão mapeados resultados que vão desde a não viabilidade de construção do aplicativo, até as arquiteturas de implementação mais indicadas: Nativo, nativo-híbrido ou híbrido.

É importante ressaltar que a matriz proposta não tem o objetivo de contemplar todas as possibilidades a serem avaliadas no momento de se decidir qual a estratégia a ser adotada no desenvolvimento do aplicativo. Trata-se de um guia simplificado e de alto nível, construído com as principais questões a serem respondidas durante esta análise. Não há a intenção de excluir quaisquer outros questionamentos, haja vista a complexidade e as inúmeras situações que podem estar envolvidas na concepção e construção de aplicativos móveis.

Esta matriz deve servir de ponto de partida para análise a ser feita, ou até mesmo a base de evolução para uma análise mais detalhada, caso necessário.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

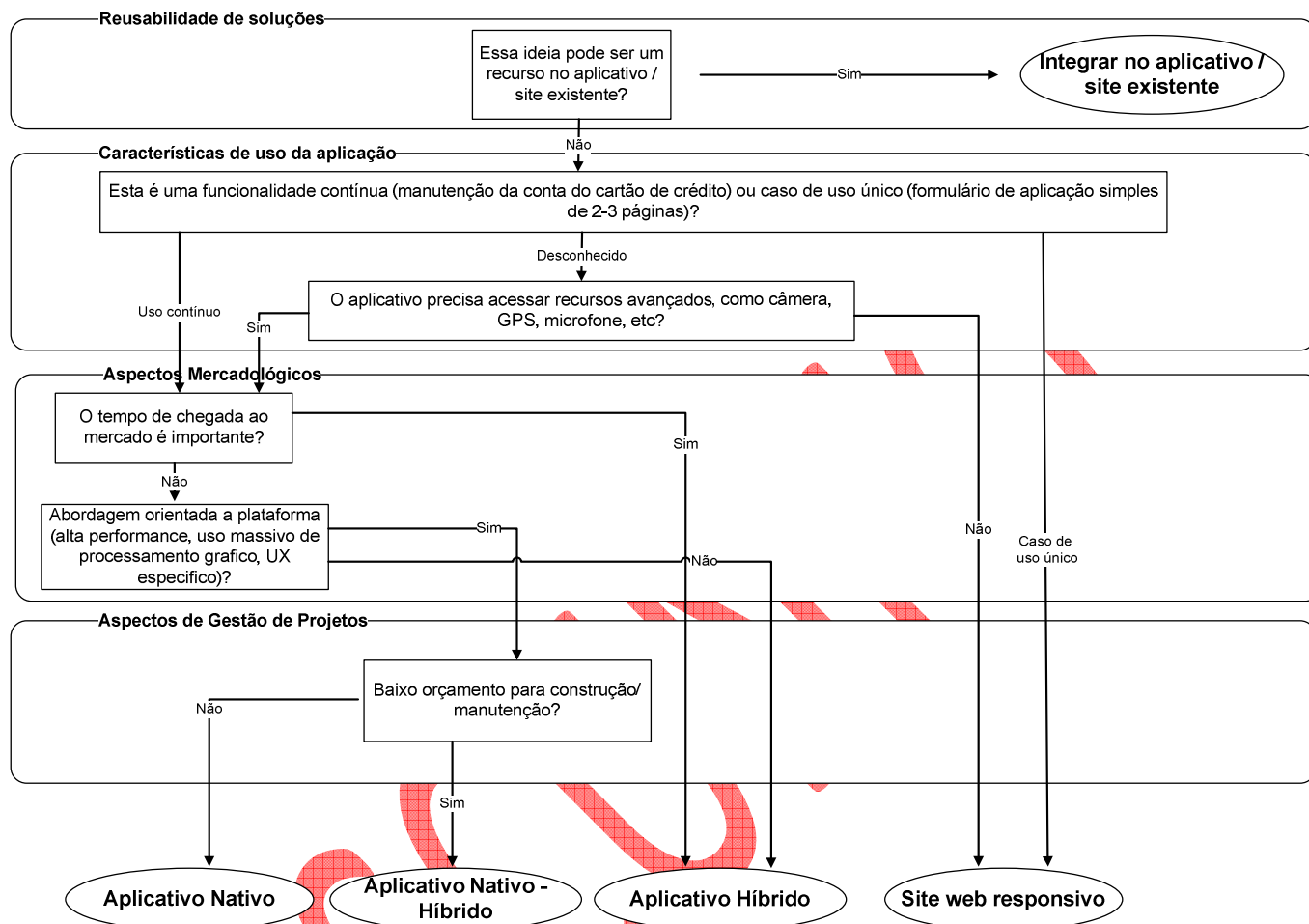



Figura 10: Matriz de Decisão (Fonte Consultoria BCG)

Seguem abaixo alguns cases da aplicação da matriz na definição da melhor estratégia de desenvolvimento de alguns aplicativos da CAIXA:

**Cartão Virtual do Cidadão:** Proposta inicial de um aplicativo específico para consulta de saldos e pagamentos de benefícios. Após análise do critério de reusabilidade, optou-se pela unificação das funcionalidades da consulta de saldos (já implementada em um outro aplicativo) e a funcionalidade de pagamento em um único aplicativo.

**Cartões Caixa:** Atende ao critério de reuso, de recorrência de uso e uso de recursos avançados típicos de um dispositivo móvel, mas devido ao curto *time to market* e

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

moderada utilização de recursos do dispositivo em seu ciclo inicial de vida, optou-se pelo desenvolvimento do aplicativo em uma plataforma híbrida inicialmente.

Outra abordagem a ser adotada para construção de aplicativos é a evolutiva. O aplicativo em um contexto inicial de testes e maturação é desenvolvido de forma híbrida, e à medida que a sua interatividade evolui, novas funcionalidades dos dispositivos são agregadas à experiência do usuário. Tecnologias de desenvolvimento nativas passam a compor uma parte cada vez mais significativa do aplicativo até que finalmente o aplicativo atinja o ápice de importância estratégica e exigência de uma experiência de uso específica da plataforma que justifique os custos e complexidades inerentes a um desenvolvimento puramente nativo.

### **3.1.3. Plataformas e Ferramentas de Desenvolvimento**


Cada estratégia de desenvolvimento discutida anteriormente é suportada por um conjunto de ferramentas (IDEs, SDKs, APIs, linguagens, etc) as quais terão os seus principais conceitos aqui relacionados.

O objetivo não é delimitar o uso de uma tecnologia ou outra dentro CAIXA, mas sim estabelecer uma linha de entendimento acerca dos princípios de uso vinculados a cada uma delas, mesmo porque todas seguem evoluindo e novas surgem a todo momento, as vezes como uma extensão das existentes, como é o caso do framework Apache Cordova, base para o Ionic.

#### **3.1.3.1. Android**

O sistema operacional criado em 2005 compartilha o mesmo kernel do Linux e é atualmente propriedade da Open Handset Alliance, um consorcio de empresas criado pelo Google e com participação das principais fabricantes de celulares do mundo como a Motorola, Samsung e LG.

O desenvolvimento nesta plataforma prevê a instalação do JDK (Java Development Kit) e o Android SDK (Software Development Kit), disponíveis publicamente aos

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

desenvolvedores desde setembro de 2008. O SDK contém todas as bibliotecas e APIs para manipular os aplicativos nativos da plataforma e os recursos de hardware do dispositivo, como GPS, acelerômetros, tela sensível ao toque, rede de dados, etc.

Como ambiente de desenvolvimento não há uma limitação. Pode se utilizar o Android Studio, Eclipse, Netbeans ou IntelliJ entre outras.

Cabe por fim salientar que a máquina virtual Java (JVM) que roda nos dispositivos Android é uma versão da tradicional que roda em desktops e contém o seu próprio conjunto de instruções. Não há portanto compatibilidade entre os binários de ambas plataformas e mesmo através de recompilação, nem todas as bibliotecas Java tradicionais funcionam no Android. A plataforma suporta o desenvolvimento com as linguagens Java, C, C++, HTML+CSS+JSS entre outras, com diferente níveis de desempenho, compatibilidade e conjunto de recursos.


Mais informações sobre a plataforma podem ser obtidas em <https://developer.android.com/index.html>.

### **3.1.3.2. iOS**

O sistema operacional da Apple lançado em 2007 (inicialmente com o nome de iPhone OS) é baseado no OS X, sistema operacional da Apple para desktops e notebooks. Para instalar o SDK e programar para o iOS é necessário um computador que rode o OS X.

Um ponto a se destacar aqui é que ao contrário de outros sistemas mobile como o Android, detalhes no iOS são mais padronizados, existindo uma quantidade muito menor de dispositivos e resolução de tela, o que simplifica muito a vida na hora de criar aplicativos.

A Apple disponibiliza gratuitamente todas as ferramentas e documentação para desenvolver para iOS, incluindo simuladores iPhone e iPad, além da sua IDE de desenvolvimento, o XCode.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

As linguagens utilizadas para o desenvolvimento nesta plataforma são o Objective-C e o Swift. Mais informações podem ser obtidas em: <https://developer.apple.com/documentation/>.

### **3.1.3.3. Tecnologia Cross-plataform**

#### **3.1.3.3.1. Cordova**


O projeto Apache Cordova é o principal representante desta estratégia de desenvolvimento. Trata-se da versão *open source* mantida pela Apache Foundation. A versão original que serviu de base para o Cordova é o Phone Gap, de propriedade da Adobe.

A partir deste framework vários outros foram implementados melhorando ou otimizando algumas características/funcionalidades pouco exploradas nele, tal como o Ionic (desenvolvido com o objetivo de agregar uma melhor experiência com o usuário por meio do provimento de componentes de interface customizados e com bastante similaridade de aparência /comportamento das suas respectivas versões nativas), entre outras.

O ponto forte do projeto Apache Cordova é intermediar o acesso a funcionalidades nativas dos dispositivos, como: acelerômetro, câmera, conexões, contatos, eventos diversos, arquivos, geolocalização, entre outras. Para isso mantém um conjunto de plug-ins chamados "core plug-ins". Este framework precisa de outra ferramenta ou APIs especializadas na criação visual do aplicativo, que se responsabilize pela interface com o usuário. Para auxiliar nesta tarefa são utilizados vários outros frameworks como por exemplo: Ionic, Sencha Touch, Dojo Mobile, jQuery Mobile entre outras.

Mais informações acerca dos plug-ins Cordova assim como das especificidades do framework podem ser encontradas em: <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>.



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

### 3.1.3.3.2. Xamarin

Trata-se do principal representante de uma classe de ferramentas cuja filosofia é preservar a experiência interativa pura do usuário dentro de sua plataforma mobile de uso e ao mesmo tempo prover as facilidades tão almejadas pelos desenvolvedores mobile, tais como a gestão de configuração e mudança mais simples do código, reuso e o domínio de uma única linguagem de programação. O aplicativo gerado portanto por esta ferramenta é nativo, mas a partir de uma linguagem única: o C#.

Trata-se de uma ferramenta adquirida pela Microsoft e incorporada à sua suíte de desenvolvimento (Visual Studio) e, como já dito, utiliza a linguagem C# (e outras *features* do C# e do .Net suportadas pelo Visual Studio) no desenvolvimento dos aplicativos. No projeto, o desenvolvedor tem a possibilidade de separar o código comum a todas as plataformas alvo, do código específico (envolvendo aspectos inerentes à plataforma tais como acesso ao hardware entre outros).


Tal façanha é implementada pelo Visual Studio (única ferramenta que suporta o desenvolvimento com Xamarin) por meio da importação e utilização no código do projeto escrito em C# das bibliotecas e APIs providas pela SDK nativa de cada plataforma considerada, bem como recursos específicos da ferramenta como o Xamarin.Forms.

No caso de utilização do Xamarin.Forms, os componentes visuais são convertidos em componentes nativos de acordo com a plataforma alvo. A plataforma promete agilidade na atualização para garantir o sincronismo com as novidades (componentes de interface, etc) inseridas em cada plataforma (em até dois dias após a atualização do SO).

### 3.1.3.3.3. Appcelerator Titanium

Appcelerator Titanium era utilizado para desenvolvimento de aplicações desktop e comparava-se ao Adobe Air. Em 2012 APPAcclerator já suportava desenvolvimento para



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

iPhone e Android, em 2016 foi adicionado Appcelerator Hiperloop, que é utilizado para acessar APIs nativas para iOS, Android e Windows Phone, neste mesmo ano Appcelerator foi adquirida pela empresa Axway que manteve os planos opensource para aplicação e suas APIs de acesso nativo.

O open source formawork permite criar aplicativos móveis nativos, híbridos ou cross-platforms. Dentre os principais benefícios destacam-se: SDK JavaScripts com suporte a APIs para iOS e Android; Desenvolvimento baseado em tecnologia web que diminui a curva de aprendizado; Desenvolvimento cross-plataforma com possibilidade de reuso de até 90% do código gerado.

#### **3.1.4. Identificação do Dispositivo**


O ID do dispositivo é uma identificação única associada ao smartphone ou qualquer outro dispositivo portátil. A identificação do dispositivo não faz parte do serial number.

Todo dispositivo Apple tem único número ID (UDID – unique device ID number). O ID Apple é composto de uma sequência de 40 dígitos que incluem letras e números, que podem ser acessados via iTunes ou via aplicativo.

Os dispositivos Android são determinados durante a primeira ativação (boot). Esse ID deve permanecer o mesmo durante o tempo de vida do dispositivo, a menos que seja realizada reposição de fábrica.

##### **3.1.4.1. Padrão de geração de ID de dispositivo**

Para uma gestão efetiva dos dispositivos dos usuários é mandatório a geração de um código identificador único para cada dispositivo, com regras de cálculo padronizada para toda a empresa.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Há algumas especificações de geração de identificadores disponíveis no mercado e a grande maioria utiliza algoritmos baseados em componentes físicos do dispositivo, como IMEI entre outros dados.

Para os aplicativos da CAIXA, o padrão a ser utilizado na identificação única de dispositivos é o UUID, um número de 128 bits padronizado pela Open Source Foundation. A implementação desta especificação está disponível para todas as plataformas utilizadas por meio de APIs e SDKs a serem importadas nos projetos feitos em desenvolvimento nativo e híbrido.

Os detalhes de como o UUID é gerado são determinados pelo fornecedor do dispositivo e são específicos para cada modelo e plataforma.

### **3.1.5.Criticidade dos Aplicativos**

Os Aplicativos deverão passar, anteriormente ao início do desenvolvimento, por uma consultoria das áreas de segurança (GESET e GEFEM) para verificação dos níveis de segurança exigidos para cada produto.

De acordo com critérios de risco definidos, os Aplicativos serão enquadrados em uma classificação própria das áreas de segurança.


As definições se diferem para o público de clientes CAIXA (externo) e para os empregados (interno).

### **3.1.6.Diretrizes para o Front-end**

As plataformas móveis abarcadas pelos projetos de mobilidade da CAIXA são:

- iOS;
- Android.

As IDEs de desenvolvimento utilizadas pela CAIXA nos projetos de mobilidade são:

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

- XCode;
- Android Studio;
- Android Studio;
- Eclipse;
- Netbeans.

As linguagens de programação utilizada para desenvolvimento nativo são:

- JAVA (Android);
- SWIFT(IOS).

Os FRAMEWORKS de desenvolvimento híbrido adotados são:


- Cordova;
- IONIC.

O padrão a ser utilizado na identificação única de dispositivos é o UUID.

A estratégia de tombamento dos Ids de dispositivos atualmente registrados no SIPER - gerados pela ferramenta da GAS - para IDs gerados conforme a especificação UUID deverá ser definida pela GESET.

A definição da criticidade do aplicativo deve ser feita com a consultoria das áreas de segurança (GESET e GEFEM).

**Quaisquer outras tecnologias devem ser validadas pela GEARQ antes de ser utilizada.**

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

### 3.2. MIDDLEWARE

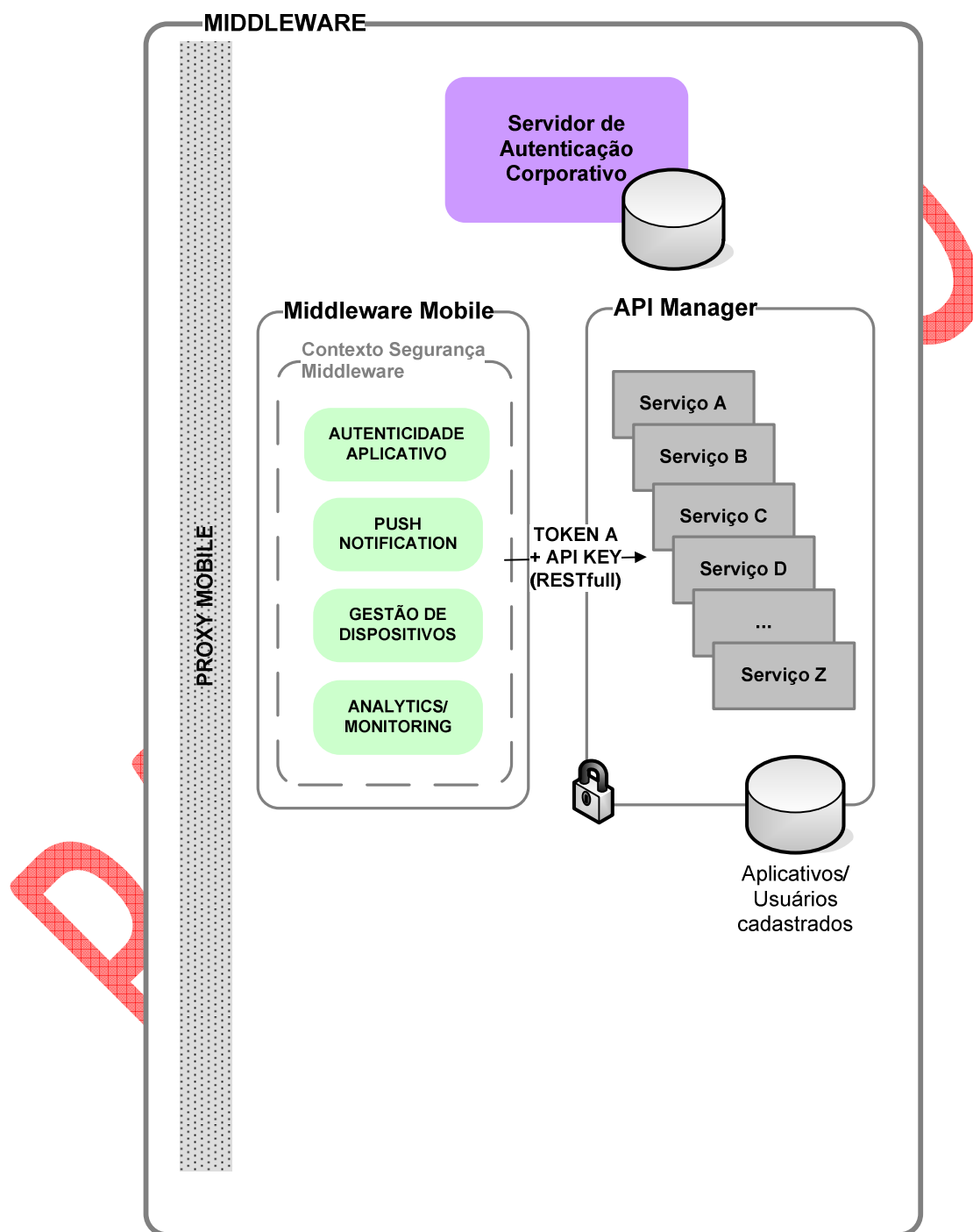



Figura 11: Middleware da Arquitetura de Referência

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Em linhas gerais esta camada detém as seguintes responsabilidades

- Recepção de forma segura das conexões oriundas do front-end;
- Execução de atividades relacionadas à segurança de acesso às funcionalidades disponibilizadas (tais como a identificação/validação do dispositivo, aplicativo, usuário);
- Registros das requisições/interações do usuário com o front-end (analytics e monitoring);
- Direcionamento destas solicitações à camada de back-end, responsável pela sua execução de fato.

As ferramentas responsáveis por tais atividades e que compõem esta camada na arquitetura de mobilidade da CAIXA estão detalhados na sequência.

### 3.2.1.Proxy Mobile

O proxy mobile corresponde à borda de contato da CAIXA com o aplicativo na internet e tem por finalidade recepcionar a requisição baseada em tunelamento SSL em porta padronizada do protocolo e redirecionar este pacote para os componentes internos da arquitetura (middleware de mobilidade ou API manager) através de IP interno e porta específica exercendo a função de gateway dentro de uma DMZ.

Esses componentes internos são protegidos por firewall de forma a garantir que todo o acesso realizado possa ser registrado e identificado origem de rede, porta e serviço desejado.

Existem opções a fazer esse papel como o IHS - IBM HTTP Server, Apache HTTP Server ou NGINX. Nestes servidores são instalados os certificados SSL para permitir a comunicação com os dispositivos. A Apple necessita que o certificado seja emitido por entidade certificadora internacional como a COMODO.

O acesso aos recursos disponibilizados pelos back-ends via middleware não devem ser expostos diretamente, sendo que o proxy mobile tem essa função. O Domain Name System (DNS) deve ser resolvido para um pool de servidores de apresentação que redireciona a requisição para o *resource owner*.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

No cenário de APIs externas o redirecionamento é para o API Manager externo, sendo estas APIs acessadas normalmente por parceiros e desenvolvedores.

No cenário de APIs internas o redirecionamento é para o API Manager interno, sendo os recursos Caixa protegidos por esta infraestrutura.


A função de proxy mobile pode ser, em alguns casos, assumida pelo API Manager externo, sendo a requisição direcionada diretamente para ele que protegerá o acesso aos recursos internos. Essa configuração é possível na ferramenta, mas será avaliada pela GEARQ quando a necessidade se apresentar.

Quando o APP utilizar o Mobile First a função de proxy mobile é desempenhada pelo IHS - IBM HTTP Server, que redireciona a requisição para o IBM Mobile First acessar o back-end da aplicação.

### **3.2.2. Servidor de Autenticação Corporativo**

O Sistema Corporativo Caixa para autenticação de usuários e autorização de acesso a recursos é o Siset – Sistema de Segurança Tecnológica - internamente denominado Login.caixa. Este sistema está baseado na solução KeyCloak da Red Hat, a qual alterou o nome, recentemente para RED HAT SSO. Portanto registre-se, todas essas nomenclaturas fazem referência à mesma solução, sendo esta gerida pela Gerência Nacional de Segurança e Continuidade de Negócios em TI - GESET.

O Siset é uma solução de mercado para single sign-on e federação de identidades baseada em especificações de mercado (SAML 2.0, OpenID Connect e OAuth 2.0) adquirida pela CAIXA para centralizar todo o processo de autenticação de usuários na organização. Atualmente cada nicho de negócio da empresa praticamente possui a sua própria base autenticadora, o que traz sérias dificuldades e limitações ao projeto, desenvolvimento e disponibilização aos clientes de uma plataforma de aplicativos CAIXA ágil e de fácil utilização. Em linhas gerais esta ferramenta fornece serviços de autenticação, fornecimento de tokens, sementes geradoras de senhas OTPs e validação de senhas OTPs entre outras funcionalidades. Todas as funcionalidades implementadas pelo Siset estão disponíveis para uso das aplicações por meio de serviços REST.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	


O Siset (Login.caixa) é a solução que proverá acesso único aos aplicativos Caixa, não havendo necessidade de se autenticar novamente ao se navegar entre os APPs implantados com este. Diante dos benefícios, os aplicativos devem implementar o acesso via Siset, o qual já fornece tela de login padrão via serviço, porém, alternativamente, o APP poderá desenvolver a própria tela de login, mas obrigatoriamente utilizar o serviço da solução corporativa.

Sistemas alternativos de provimento de autenticação não devem ser utilizados.

### 3.2.2.1. RedHat Single Sign-on

Descrição	Red Hat Single Sign-On (RH-SSO) baseado no projeto Keycloak permite proteger as aplicações por meio de Web SSO utilizando-se padrões como: SAML 2.0, OpenID Connect e OAuth 2.0. O servidor pode interagir SAML ou OpenID Connect based identity provider (Idp), mediando acessos dos usuários corporativos através de informações de credenciais do usuário e de aplicação baseado em padrões de tokens de acesso. Objetiva modernizar a arquitetura de provisionamento e autenticação, proporcionar a gestão de acessos com múltiplos fatores de autenticação.
Características	<ul style="list-style-type: none"> <li>• <b>Authentication Server</b> <ul style="list-style-type: none"> <li>◦ Trabalha autonomamente como Sercurity Assertion Markup Language (SAML) ou OpenID Connect baseado em provedor de identidade;</li> </ul> </li> <li>• <b>Identity Brokering</b> <ul style="list-style-type: none"> <li>◦ Integração com provedores de identificação de terceiros, incluindo redes sociais como fonte identidade, ou seja, login social;</li> </ul> </li> <li>• <b>User Federation</b> <ul style="list-style-type: none"> <li>◦ Certificação via servidor LDAP e Microsoft Active Directory com fonte de informações de usuários;</li> </ul> </li> <li>• <b>REST APIs and Administration GUI</b> <ul style="list-style-type: none"> <li>◦ Especificação de federação de usuário, mapeamento de perfil de acesso, e aplicação clientes com administração via interface de usuário e REST APIs;</li> </ul> </li> </ul>



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

A figura abaixo representa o processo básico de realização de login do usuário junto ao Login.caixa para aplicações/aplicativos da CAIXA, bem como os atores e ações envolvidas:

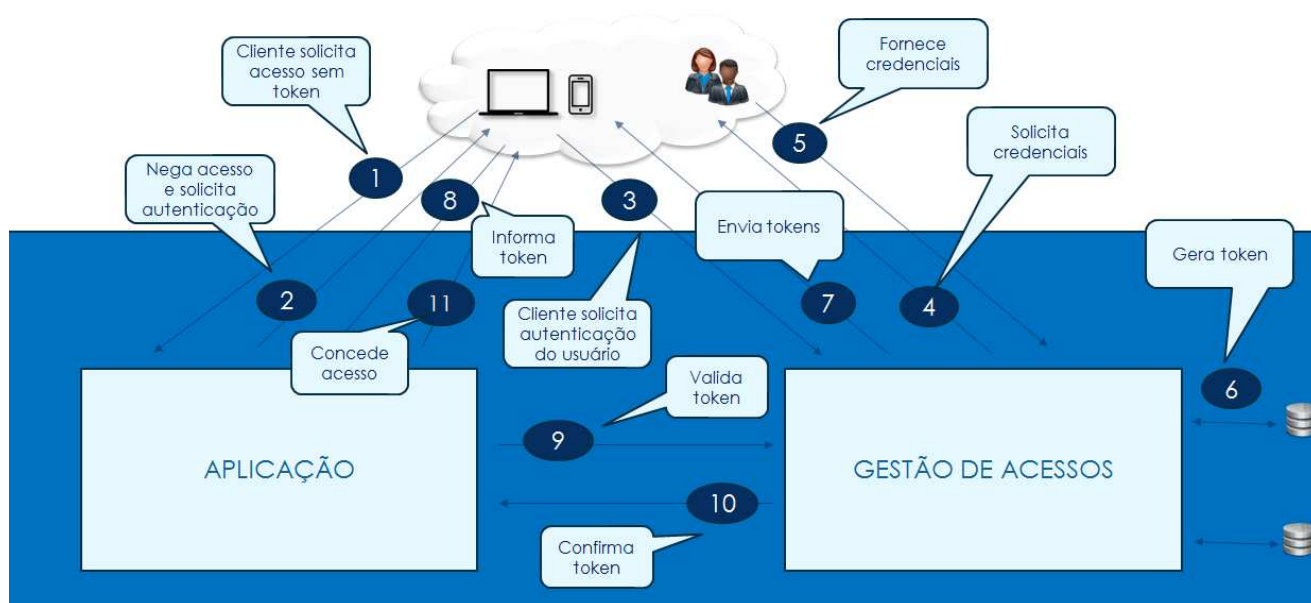



Figura 12: Processo do login.CAIXA (Gestão de Acesso Corporativo)

### 3.2.3.Vinculação de Dispositivo

Este processo é de fundamental importância para o atendimento de exigências de segurança tais como rastreabilidade e monitoramento de dispositivos móveis e usuários a eles vinculados.



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Para isso é necessário um componente que faça o registro da identificação dos clientes e dispositivos móveis a eles vinculados, bem como os aplicativos utilizados entre outras informações para a formação e controle do perfil do cliente e monitoramento de fraudes.


O componente deve manter o registro do nível de segurança atribuído ao conjunto usuário/dispositivo conforme a classificação de criticidade do aplicativo em uso.

Os níveis de segurança definidos pela GESET para os aplicativos móveis variam de 0 a 3 – do mais baixo para o mais alto - e estão diretamente vinculados ao nível de segurança do canal utilizado para realizar a vinculação do dispositivo ao usuário.

- **Nível 0:** Auto provisionamento – não há nenhum fluxo de validação de informações previsto para que o dispositivo seja vinculado ao usuário.
- **Nível 1:** Própria aplicação - a vinculação do dispositivo está correlacionada à execução de um fluxo de validação de informações internas do próprio sistema demandante, sem a necessidade de se utilizar um segundo canal confiável;
- **Nível 2:** Canal digital - fluxo de confirmação de informações executado por meio de um segundo canal digital com um relativo nível de confiabilidade: push notification, e-mail, sms, etc...;
- **Nível 3:** Canal físico - fluxo de confirmação de informações executado por meio de um segundo canal físico: ATM, certificado digital, agência, etc... É atualmente o canal com maior grau de confiabilidade;

Um dispositivo ativado para um determinado nível de segurança está automaticamente ativado para os níveis inferiores, de forma que a instalação de um aplicativo vinculado a um nível de segurança inferior não demandará a realização do processo de vinculação do dispositivo novamente.

Salientamos que mesmo para aplicativos vinculados ao nível 0 de segurança, alguns dados relacionados às interações processadas deverão ser coletados pelo middleware de

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

mobilidade e registrados na base do sistema de gestão de dispositivos (aplicativo utilizado, data e horário de conexão, entre outras).

Por fim ressalta-se que a vinculação dos dispositivos junto ao middleware de mobilidade é realizada por meio da execução de dois processos distintos: Um responsável pelo registro e governança sobre os usuário/dispositivos e o seus respectivos níveis de criticidades atrelados e um outro responsável pela recepção e validação de informações críticas fornecidas pelo usuário por meio de um segundo canal confiável, essencial à validação e registro do conjunto dispositivo/usuário em questão.


### 3.2.4. Middleware de Mobilidade

O *middleware* de mobilidade é o componente responsável por recepcionar as requisições dos aplicativos, realizar alguns processamentos específicos desta camada (em sua maior parte relacionadas à segurança) e direcioná-las para as estruturas internas da organização que efetivamente proverão os serviços. Ela é projetada para executar funções relacionadas:

- à garantia de autenticidade do aplicativo,
- proteção de acesso aos recursos corporativos,
- coletar informações de *analytics* e *monitoring* relacionadas ao consumo destes recursos,
- gerenciamento de versões dos recursos,
- gestão de dispositivos,
- entre outras.

Para abstrair as chamadas a tais serviços a partir dos aplicativos a ele conectados, o *middleware* disponibiliza SDKs e APIs específicas para as principais plataformas de desenvolvimento do mercado (iOS, Android e frameworks híbridos).

Toda a comunicação do aplicativo com o *middleware* de mobilidade deve ser baseada no padrão REST, seguindo as recomendações de uso dos verbos HTTP para identificação da ação. E a troca de informações deve ser baseada em JSON de forma a melhorar a

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

performance evitando operações custosas de “parser” para analisar o corpo da requisição/retorno da informação.

Os *middlewares* atualmente disponíveis na CAIXA são o IBM Mobile First e o CA Mobile Application Gateway, componente integrante da suíte IDMS Orquestrator, os quais serão melhor detalhados abaixo.

#### **3.2.4.1. CA Mobile API Gateway (IDMS Orquestrator)**


A suíte CA IDMS Orquestrator adotada pela CAIXA para realizar a gestão de APIs, provê um conjunto de ferramentas relacionadas à governança na publicação e consumo de API corporativas, bem como um middleware de mobilidade (Mobile API Gateway).

Basicamente o Mobile API Gateway (MAG) fornece um gerenciamento centralizado de APIs a serem consumidas pelos aplicativos móveis. A ferramenta disponibiliza SDKs específicas (iOS, Android, Cordova) para que os desenvolvedores possam implementar de forma transparente complexas funcionalidades de segurança (autenticação de aplicativos e usuários entre outras).

Esta ferramenta integra de forma segura o *front-end* com o *backend* disponibilizando aos desenvolvedores SDKs e APIs que permitem basicamente o controle sobre quais usuários, dispositivos e aplicações podem acessar os recursos protegidos da organização.

O Mobile API Gateway viabiliza também a implementação de uma gestão de dispositivo, disponibilizando de forma integrada serviços para cadastro, bloqueio, desbloqueio de dispositivos bem como serviços adicionais de autorização baseada em regras de validação do aplicativo/dispositivo providos de forma integrada pelo módulo CA Advanced Authentication.

Há algumas outras funcionalidades disponibilizadas pela ferramenta não utilizadas em um primeiro momento mas com potencial de uso futuro, dentre os quais podemos mencionar a autenticação por login social, controle de acesso por geolocalização, autenticação de dispositivo para dispositivo, Apple and Android push notification e Samsung Knox Integration.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Maiores informações acerca da CA Mobile API Gateway em <https://docops.ca.com/ca-mobile-api-gateway/4-0/en>;

### 3.2.4.2. IBM Mobile First


Plataforma da IBM que provê um framework para o desenvolvimento, otimização, integração, e gestão da segurança de aplicativos móveis sem a introdução de uma linguagem de programação ou modelos proprietários de desenvolvimento.

Aplicativos podem ser desenvolvidos nesta plataforma com o uso de HTML5, CSS3 e JavaScript (Cordova) bem como com código nativo das plataformas suportadas: Java para android e swift para iOS.

A ferramenta fornece SDK que incluem bibliotecas e chamadas acessíveis pelas tecnologias de desenvolvimento suportadas para a criação de aplicativos, bem como componentes programáveis no lado Server (JavaScript e Java) usados para conectar com segurança os aplicativos clientes ao back-end corporativo.


A seguir uma breve descrição das principais funcionalidades embarcadas na ferramenta:

- **Implementação de OAuth:** Possibilita que os adapters disponíveis realizem autenticação utilizando padrão de mercado OAuth e com isso permitir a confiança em outros mecanismos de autenticação como SSO ou Login Social (Google+, Facebook, Twitter, etc);
- **Autenticidade de APP:** Confirmação de "checksum" da APP no handshake garantindo que as requisições sejam atendidas apenas a clientes construídos pelo Mobile First. Estas informações vão no pacote gerado para o cliente e ficam no servidor para conferência;
- **Offline Authentication:** Caso a comunicação da plataforma com a solução de IAM esteja indisponível, a plataforma provê uma estrutura para armazenar os tokens previamente gerados e permitir o acesso a clientes que estejam com o mesmo ainda

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

dentro da validade, mesmo que não possa trocar informações momentaneamente com o IAM;

- **Live Update:** Para aplicações híbridas, é possível colocar pacotes de conteúdo diferentes e indicar alguns dispositivos previamente para que possam acessar as URL (Serviços disponibilizados pelos adapters) para solicitar as funcionalidades e ir ampliando o conjunto de dispositivos/clientes até que a versão seja liberada totalmente ao público;
- **Gerenciamento de versões:** A plataforma permite que uma mesma aplicação possa atender a duas ou mais versões diferentes de forma simultânea, dessa forma os clientes podem levar um tempo maior para atualizar suas aplicações junto a loja de aplicativos antes de que uma versão específica deixe de ser atendida. Também é possível desabilitar remotamente essas versões mantendo o código ainda na plataforma;
- **Gerenciamento de Dispositivos:** A plataforma gera uma identificação do dispositivo que gerou requisições de forma que o mesmo possa ser “desabilitado” para uma solução específica ou para toda a plataforma de forma a que futuras requisições deste dispositivo não sejam mais atendidas;
- **Direct Update:** Para aplicações não nativas, é possível distribuir os pacotes de wlap (Pacote de recursos estáticos como HTML, JS, CSS) permitindo que a aplicação seja atualizada sem a necessidade de encaminhar a atualização para a loja atendendo a correções de forma imediata;
- **Monitoração de APPs e Diagnóstico de problemas:** Estrutura de logs baseado em ElasticSearch e customizações de relatórios baseado nos eventos de logs, captura de logs por dispositivos e informações de crash. Possibilidade de integração com ferramenta Kibana para navegação de dados do ElasticSearch;
- **Estrutura de Analytics Operacional:** A plataforma conta com uma estrutura de Analytics vinculada aos seus adapters e capturados de forma automática de forma a permitir uma análise mais detalhada sobre roundtrip de requisições ou informações de crash por exemplo;

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

### 3.2.5.API Manager

Todos os recursos disponibilizados para consumo dos aplicativos mobile Caixa devem necessariamente lançar mão de padrões de programação para acesso a aplicativo ou plataforma baseado na Web, ou seja, deve-se construir uma API que proveja serviços Web REST, sendo esta necessariamente publicada no API Manager, que é responsável por toda governança das APIs, tais como: políticas de utilização, acesso e *rate limit*.

### 3.2.6.Analytics e Monitoring


Estas funcionalidades visam a captura e geração de insumos tanto sobre o comportamento do usuário no dispositivo quanto sobre o consumo do serviço disponibilizado.

Nesta camada da arquitetura recomenda-se que sejam armazenadas as informações de forma a permitir análises pontuais e momentâneas de forma mais ágil e direta, porém diariamente os dados gerados devem ser repassados a uma estrutura de BigData para que possam ser analisadas informações comportamentais históricas utilizadas na geração e oferta de novos negócios aos clientes.

As informações sobre o comportamento do cliente no dispositivo podem ser obtidas através do "track" de eventos realizados no dispositivo por parte do usuário como por exemplo, ao clicar num determinado item de menu, o aplicativo dispara uma requisição Web assíncrona (feita em batch pelo IBM Mobile First) que consome um serviço disponibilizado pela solução de *analytics* para capturar essa informação que o item de menu X foi selecionado. Neste envio de informação, pode ser identificado por exemplo:

- Plataforma do usuário;
- Identificador do dispositivo utilizado;
- Geolocalização (Caso habilitado e autorizado);
- Evento ocorrido;
- IP de acesso;



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

- Data/Hora.

### 3.2.7.Diretrizes para Middleware

O Sistema Corporativo Caixa para autenticação e autorização de acesso é o Siset.

Sistemas alternativos de provimento de autenticação não devem ser utilizados.

A vinculação de dispositivo deve ser feita por meio do Mobile First.

As APIs, necessariamente, devem ser publicadas no API Manager.

Nenhuma API de mobilidade deve ser acessada diretamente.

Todos os aplicativos, sejam Nativos, Nativos Híbridos ou Híbridos, devem consumir APIs disponibilizadas no API Manager, no qual devem ser configuradas as políticas de acesso e consumo.


As APIs disponibilizadas no *resource owner* denominado API de mobilidade devem migrar gradativamente para os respectivos backends, barramento de serviços e disponibilizadas no API Manager. Não estão autorizadas a construção de novas APIs em *resource owner*.

O API Manager deve autorizar as requisições com validação da chave de acesso do aplicativo e, quando necessário, o *access token* do usuário.

Quanto a construção de aplicativos Nativo Híbrido ou Híbrido utilizar a solução IBM Mobile First em sua versão mais recente -atualmente V8- sendo vedado o desenvolvimento em versões anteriores. Deve ser habilitada a *feature* de autenticidade de aplicativos, a qual garante as requisições por um APP Caixa válido, deve-se evitar atualizações do aplicativos via recursos *Direct Update*, em detrimento do versionamento do APP, salvo em casos de necessidade extrema.

As requisições não devem ser direcionadas diretamente ao back-end da aplicação, sendo que os serviços devem estar publicados no API Manager.

Dentro do contexto de mobilidade, é importante salientar que os serviços publicados pelo API manager devem ser síncronos, *stateless* e implementados sobre o protocolo RESTful.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

### 3.3. BACK-END

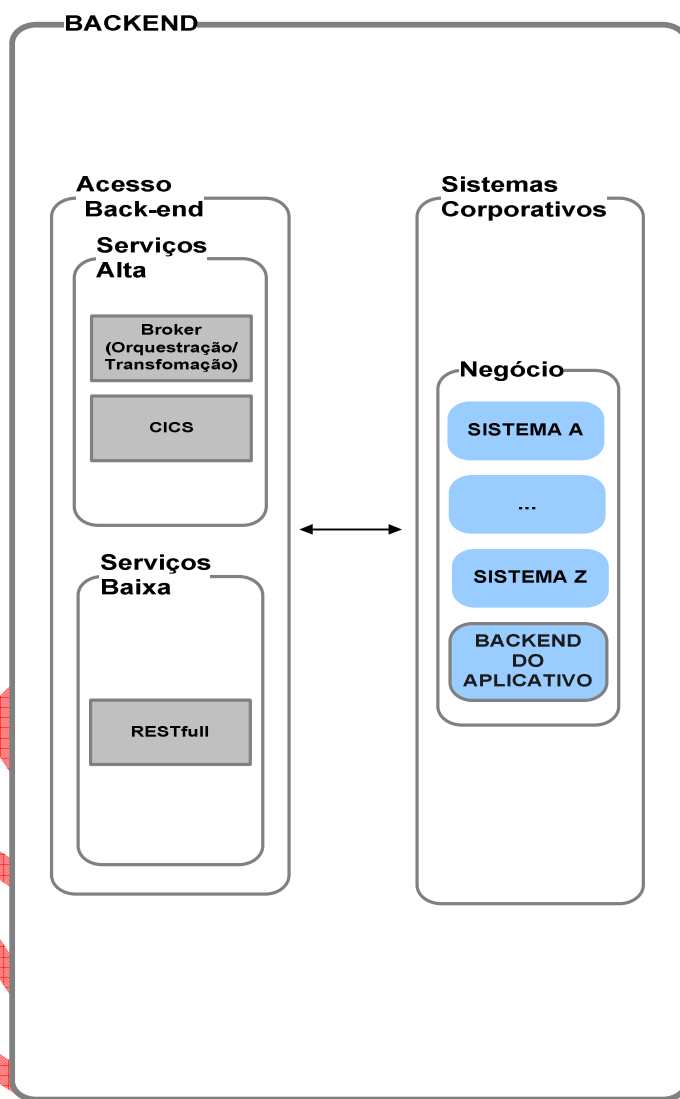



Figura 13: Back-end – Arquitetura de Referência

Esta camada é a responsável por recepcionar as requisições oriundas do front-end - já tratadas e validadas pela camada de middleware - e efetivamente prover a execução dos



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	


serviços requisitados (regras de negócios). Ela é composta basicamente por serviços disponibilizados na plataforma alta por meio do broker (IBM Integration BUS) e rotinas CICS bem como por serviços disponibilizados na plataforma baixa exclusivamente por meio do protocolo HTTP (RESTFul). O consumo de serviços através de filas, corresponde ao pior cenário possível para o mundo da mobilidade pois todo fluxo ocorre em requisições síncronas, e quando chega nesta etapa o consumo necessariamente se torna assíncrono mantendo uma sessão aberta durante a espera podendo ocasionar represamento dos atendimentos ao serviço com possibilidade inclusive de degradação e queda do ambiente devido ao estouro de memória por exemplo.

### **3.3.1. Sistemas Auxiliares no Ecossistema de Mobilidade da CAIXA**

Mesmo com uma gama bem ampla de funcionalidades de infraestrutura, apoio ao desenvolvimento e produção de aplicativos móveis, os *middlewares* de mobilidade não são capazes de suprir integralmente todas necessidades postas a este canal de interação. Algumas ferramentas até disponibilizam funções de infraestrutura, tais como envio de *push notification* ou gestão de dispositivos de forma nativa, mas as complexidades e especificidades inerentes aos sistemas e processos da CAIXA inviabilizam a utilização de tais recursos em sua forma original, ou seja, sem que seja necessário implementar customizações em tais ferramentas.

Portanto, seguindo as boas práticas de desenvolvimento e utilização de ferramentas de mercado no atendimento de suas necessidades de negócio, a CAIXA desenvolveu internamente dois sistemas para suprir tais necessidades. Salienta-se que embora estejam na camada de backend os serviços providos por tais sistemas são característicos da camada de *middleware*: serviços de infraestrutura de segurança e comunicação.

**SICPU** – Sistema desenvolvido pela CEDES/RJ responsável por centralizar toda a gestão de envio de *push notification* na CAIXA. Este sistema implementa entre outras funcionalidades o cadastro de aplicações usuárias, envio de *push notification* individual e em lote. Todas as informações necessárias ao envio de *push*, entre elas o identificador do usuário/aplicativo junto às nuvens da Apple e Android são persistidas neste sistema.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

Todas as funcionalidades implementadas pelo SICPU estão disponíveis para uso das aplicações por meio de serviços REST.

**SISIT** – Sistema desenvolvido pela CEDES/RJ responsável pela gestão dos dispositivos utilizados pelos clientes. Trata-se de um importante componente de segurança da arquitetura de mobilidade na CAIXA. O SISIT é responsável pelo cadastro do dispositivo utilizado, sua vinculação ao aplicativo e ao usuário (após autenticação por um segundo canal seguro, conforme a criticidade do aplicativo), verificação da validade do dispositivo utilizado durante as transações, bloqueio e desbloqueio de dispositivos entre outras funcionalidades. Um front-end administrativo é disponibilizado para viabilizar a visão de todas as informações necessárias a realização deste gerenciamento pelas áreas de segurança da CAIXA. Todas as funcionalidades implementadas pelo SISIT estão disponíveis para uso das aplicações por meio de serviços REST.


### **3.3.2.Diretrizes para BACKEND**

Para as APIs que suportam o canal mobilidade, o padrão de publicação a ser utilizado é RESTful.

Está proibido desta forma a exposição de serviços síncronos do mainframe por meio de filas MQ ou JCICS DIRECT.

Não devem ser construídas soluções de integração com sistemas da plataforma alta baseadas em filas MQ ou do componente JCICS Direct. As APIs existentes nesse modelo, relacionadas com a camada de mobilidade, devem ser migradas para o barramento ou WebService CICS.

Os aplicativos em ambiente produtivo que fazem acesso a serviços na alta plataforma, encapsulam as chamadas via API de mobilidade, porém essa pratica não deve ser utilizada em novas soluções. Alternativamente à chamada MQ, deve-se criar a REST API do serviço, sendo esta funcionalidade disponível no IBM Integration Bus 10, ou implementar um servidor de aplicação do IBM CICS e prover serviço web REST.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

As requisições não devem ser direcionadas diretamente ao back-end da aplicação, sendo que os serviços devem estar publicados no API Manager


#### 4. DECOMPOSIÇÃO ARQUITETURAL

Seguem abaixo os cenários arquiteturais previstos para a construção de aplicativos móveis na CAIXA. Em cada cenário, basicamente é levado em consideração a utilização ou não da ferramenta de middleware de mobilidade, suas consequências, vantagens, desvantagens e critérios de uso.

Salientamos que o cenário 3 (com o middleware de mobilidade da CA) ainda está em avaliação técnica pela GEARQ/CEDESBR e sua presença de forma preliminar neste documento visa apenas dar visibilidade à provável solução de integração desta ferramenta com o RedHat SSO elaborada pela CA.

Salientamos também que as soluções auxiliares SISIT (Gestão de dispositivos) e SICPU (*Push Notifications*) serão avaliadas pela GEARQ/GESET acerca da manutenção ou não dos seus escopos iniciais, ou da sua descontinuidade mediante a absorção de suas funções pela ferramenta de middleware de mobilidade.

##### 4.1. CENARIO 1: SEM MIDDLEWARE DE MOBILIDADE

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

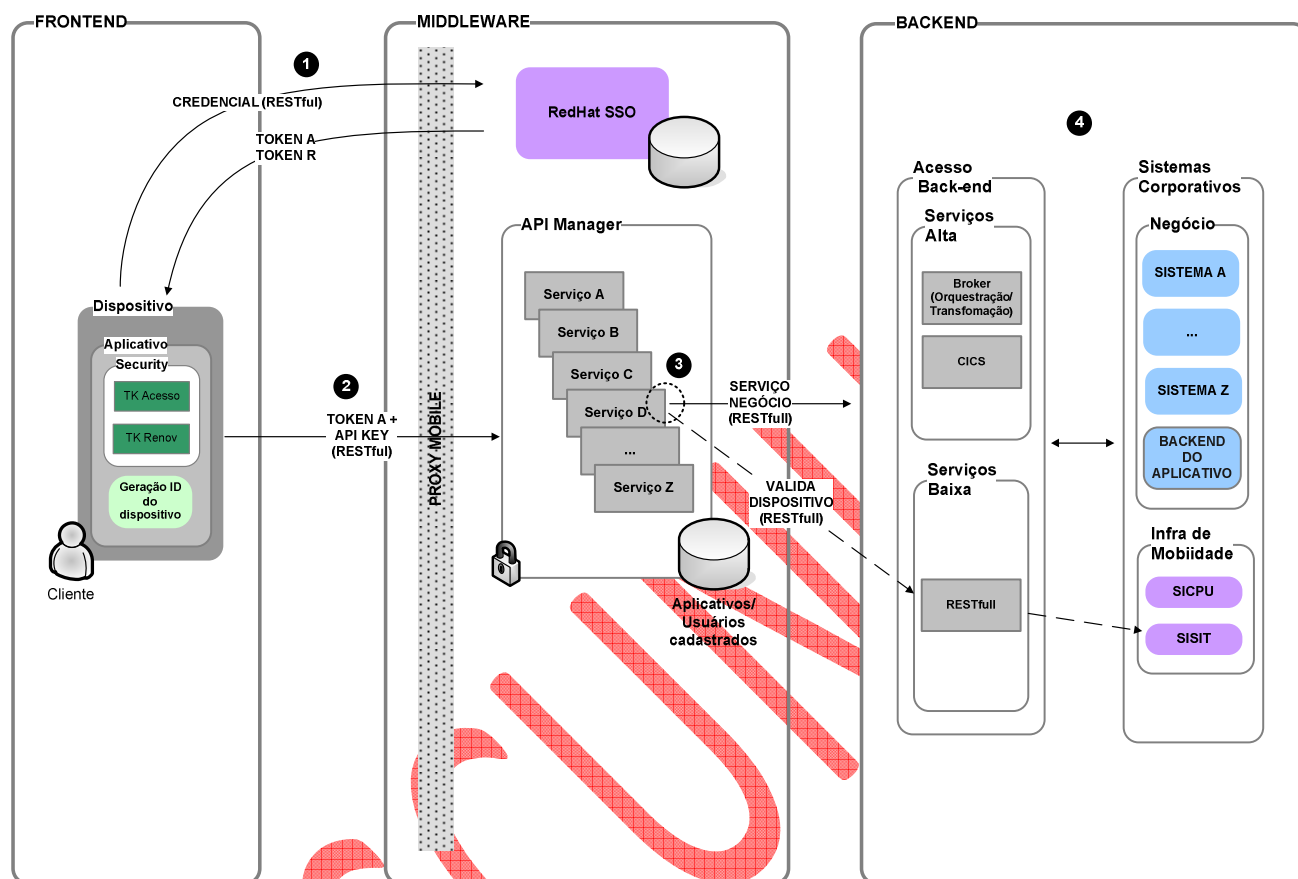



Figura 14: Arquitetura sem Mobile First

[1] – Aplicativo realiza login do usuário junto ao autenticador corporativo, o RedHat SSO. Ao final deste processo um access token e um refresh token (opcional) é gerado pelo RedHat SSO e encaminhado ao aplicativo para armazenamento.

[2] – Uma vez logado junto ao RedHat SSO e de posse dos tokens, o aplicativo está apto a realizar chamadas de serviços (RESTful) junto ao API Manager. O access token deve ser enviado na solicitação.

[3] – O API manager recebe a requisição e valida o token repassado. A vinculação do dispositivo deve ser checada junto ao SISIT antes da requisição ser atendida de fato pelo backend.

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

[4] – O serviço é enfim executado pelo *backend* e a resposta retornada ao aplicativo.

#### 4.1.1.Vantagens

- Independência de plataformas de middleware;
- Número menor de camadas envolvidas na arquitetura – arquitetura com uma latência menor de processamento;
- Maior fluidez na construção dos aplicativos tendo em vista a ausência do uso de SDKs específicas da plataforma de middleware.


#### 4.1.2.Desvantagens

- Aplicativos com um menor grau de segurança embarcado;
- Impactos na performance da requisição de negócios: Cada requisição de serviço de negócio junto ao API Manager envolve uma consulta ao SISIT para validação do dispositivo usado na chamada. Esta orquestração em princípio é feita pelo API;

#### 4.1.3.Indicação de USO

- Aplicativos com requisitos de segurança mais flexíveis (ausência de comprovação de autenticidade para os aplicativos);
- Aplicativos com baixa criticidade negocial para a CAIXA e baixo retorno financeiro;

#### 4.2. CENÁRIO 2: COM MIDDLEWARE DE MOBILIDADE – IBM MOBILE FIRST

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

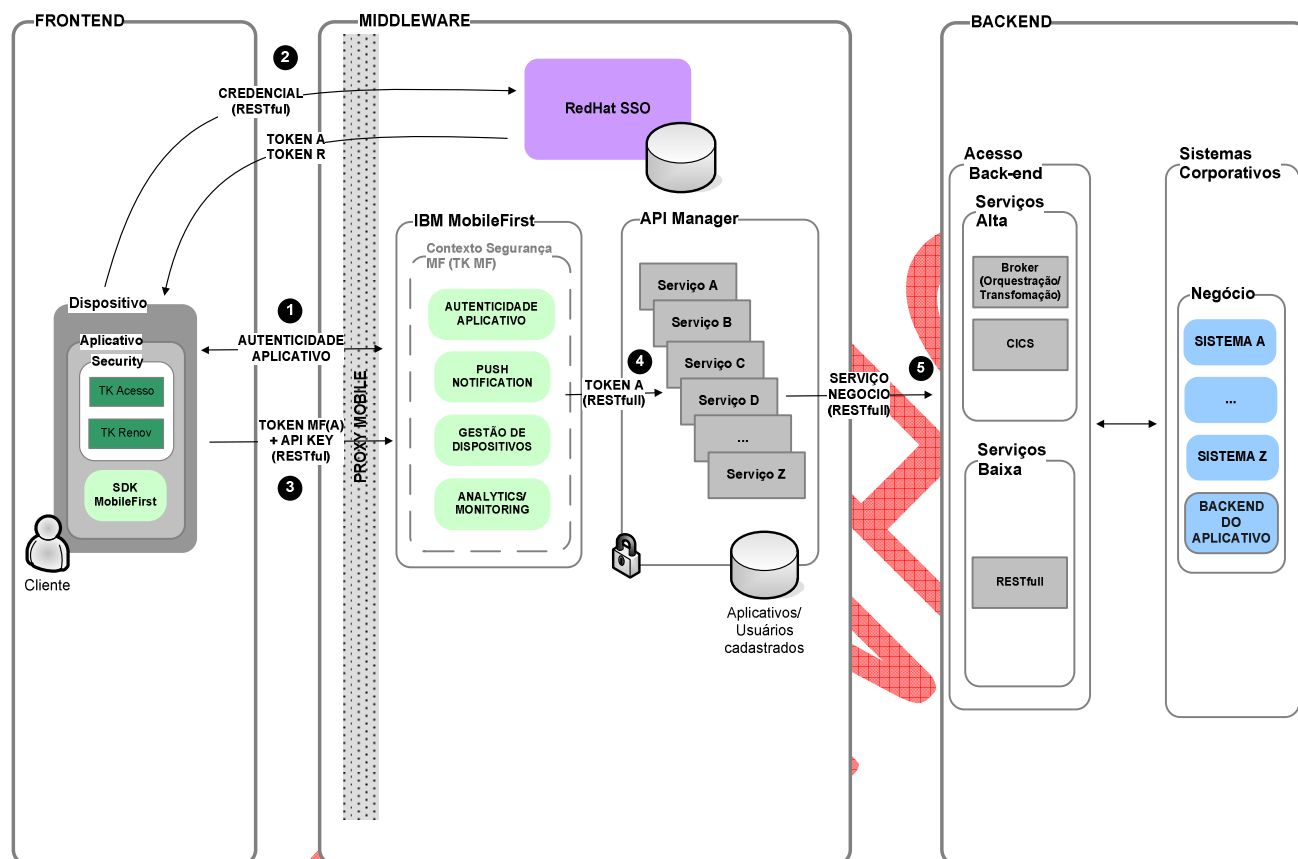



Figura 15: Arquitetura com IBM Mobile First

- [1] – Autenticidade do Aplicativo é realizada junto ao *Mobile First* em sua primeira requisição. Esta funcionalidade é nativa do *Mobile First* e sua implementação é transparente para o desenvolvedor do aplicativo;
- [2] – Uma vez autenticado, o aplicativo realiza então o *login* junto ao RedHat SSO, para obtenção do *access token* e *refresh token* (opcional). Neste momento a tela de *login* padrão CAIXA gerada pelo RedHat SSO é exibida para o usuário no aplicativo;
- [3] – RedHat SSO recebe a credencial do usuário informada na tela de login e a valida. Em caso de sucesso o RedHat SSO gera o *access token* e o *refresh token* (opcional) e os encaminha para o aplicativo. Uma vez logado junto ao RedHat SSO e de posse dos tokens, o aplicativo está apto a realizar chamadas de serviços (RESTful) disponibilizadas pelos

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

*adapters* do *Mobile First*. Em um processo intermediário de autenticação do usuário junto ao *Mobile First*, o *access token* gerado pelo RedHat SSO é encaminhado ao *Mobile First* que por sua vez faz sua validação, gera tokens próprios do seu contexto de segurança, encapsula dentro do seu token o *token* gerado pelo RedHat SSO e o envia de volta para o aplicativo. Ao enviar uma requisição junto ao adapter funcional do *Mobile First*, o aplicativo encaminha o token do *Mobile First* já vinculado ao contexto de segurança do aplicativo criado pelo middleware anteriormente. Tal contexto será necessário às funcionalidades de gestão de dispositivo e analytics disponibilizadas pelo middleware. O adapter funcional, retira o token do RedHat SSO de dentro do token *Mobile First* e o repassa ao API Manager juntamente com a requisição ao serviço de negócio;


[4] – O API manager recebe a requisição e valida a token repassado. A vinculação do dispositivo deve ser checada junto ao SISIT antes da requisição ser atendida de fato pelo backend.

[5] – O serviço é enfim executado pelo backend e a resposta retornada ao aplicativo.

#### 4.2.1.Vantagens

- Aderência às diretrizes definidas pela GESET em relação à disponibilização de uma tela única padronizada de login ao usuário, fornecida pelo RedHat SSO;
- Abstrai do desenvolvedor o armazenamento e uso do token *Mobile First* (armazenado no dispositivo) no momento das chamadas aos *adapters* de negócios providos pela SDK da ferramenta;
- Integração do *Mobile First* com o RedHat SSO feito de forma programática e com o uso das flexibilidades disponibilizadas pela ferramenta;
- Funções de *push notification* e gestão de dispositivos disponibilizados de forma nativa na ferramenta por meio de SDK/APIs;
- Envio de dados de analytics/monitoring gerenciada pela SDK embarcada no aplicativo. Pode ser efetivada em background e postergada para ocasiões de alta disponibilidade de rede.



	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	


#### 4.2.2.Desvantagens

- Moderado acoplamento à plataforma de *middleware* por conta do uso de suas funcionalidades (autenticidade de aplicativos, push, gestão de dispositivo) para atender aos requisitos do aplicativo;
- Esforço adicional de aprendizado necessário às equipes de desenvolvimento para dominar a utilização dos recursos providos pela plataforma de middleware embarcados nos aplicativos por meio das SDKs

#### 4.2.3.Indicação de USO

- Aplicativos com altos requisitos de segurança;
- Aplicativos com alta criticidade negocial e alta rentabilidade financeira.

#### 4.3. CENARIO 3: COM MIDDLEWARE DE MOBILIDADE – CA MOBILE API GATEWAY

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

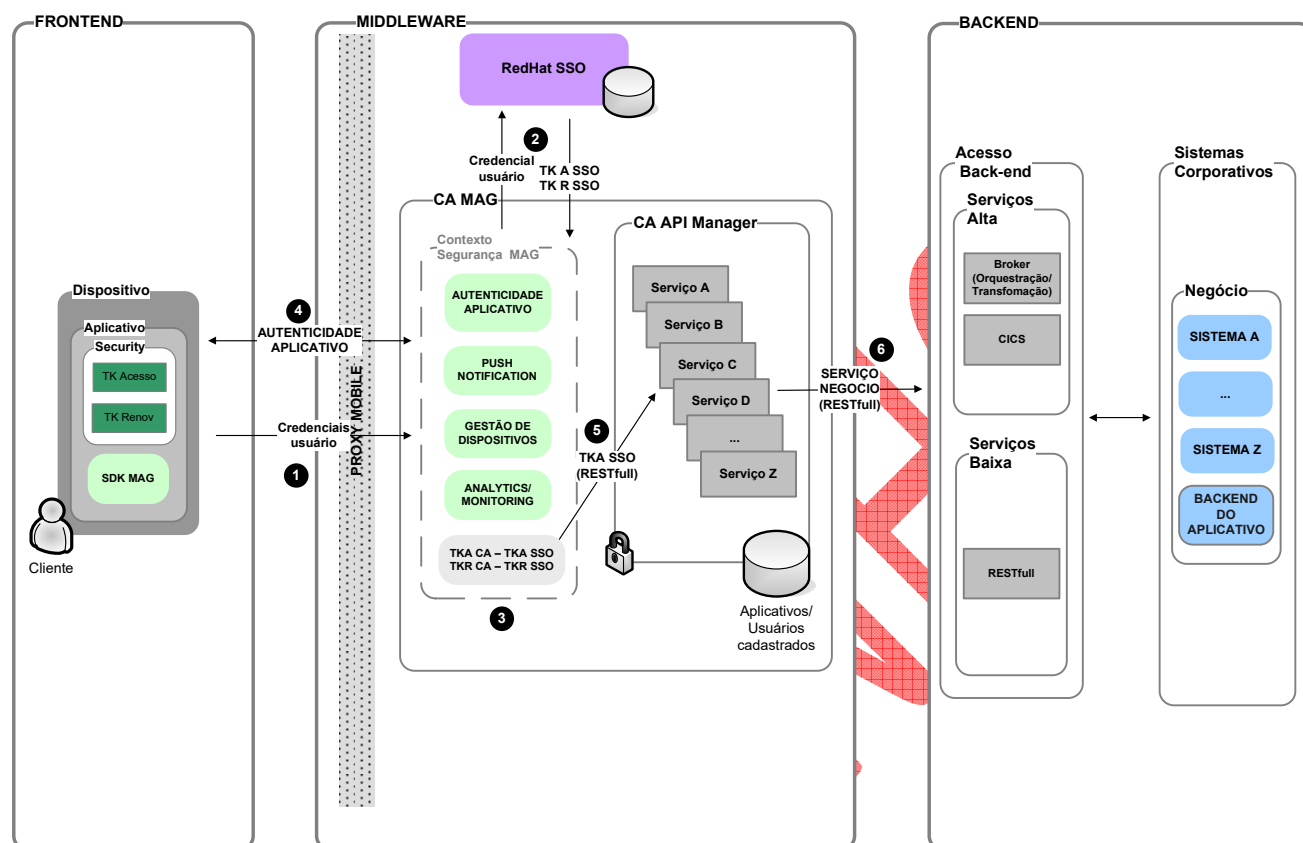



Figura 16: Arquitetura com CA Mobile API Gateway

- [1] – A credencial do usuário é obtida pelo aplicativo junto ao usuário e fornecidas ao MAG para início do processo de autenticação. A tela de login deve ser implementada no aplicativo;
- [2] – MAG realiza a autenticação da credencial fornecida junto ao RedHat SSO para obtenção dos tokens de acesso e de renovação desta ferramenta.
- [3] – Uma vez logado junto ao RedHat SSO e de posse dos tokens, o MAG cria um contexto de segurança para o usuário logado, cria os seus próprios tokens de acesso e renovação, correlaciona-os aos tokens fornecidos pelo SSO e os armazena em cache para recuperação em requisições de negócios futuras;

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

[4] – Uma vez criado o contexto de segurança do MAG, a troca de certificados entre o MAG server e o aplicativo é iniciada, viabilizando desta forma a autenticação e provisionamento do dispositivo, bem como o registro do usuário logado para fins de gestão do dispositivo.

[5] – Quando uma requisição a um serviço de negócio é recepcionada pelo MAG, ele recupera o *token* de acesso SSO correspondente ao token MAG do usuário dentro do contexto de segurança ativo e o propaga na chamada ao API Manager;


[6] – O token repassado é validado pelo API Manager, o serviço é enfim executado pelo backend e a resposta retornada ao aplicativo.

#### 4.3.1.Vantagens

- Abstrai do desenvolvedor o armazenamento e manipulação dos tokens do MAG (armazenados no dispositivo) no momento das chamadas aos serviços de negócios providos pela SDK da ferramenta;
- Integração do MAG com o RedHat SSO feito de forma simples (configurações) e com o uso das flexibilidades disponibilizadas pela ferramenta;
- Funções de push notification e gestão de dispositivos disponibilizados de forma nativa na ferramenta por meio de SDK/APIs;

#### 4.3.2.Desvantagens

- A necessidade de se implementar a tela de login para obtenção das credenciais do usuário no aplicativo não está aderente às diretrizes de segurança da GESET (Tela única de login gerada pelo RedHat SSO);
- Esforço adicional de aprendizado necessário às equipes de desenvolvimento para dominar a utilização dos recursos providos pela plataforma de middleware embarcados nos aplicativos por meio das SDKs;

	<b>Arquitetura de Mobilidade</b>	
<b>Documento</b> Arquitetura Referência de Mobilidade	<b>Data</b> 15/08/2017	

- Possibilidade de alto consumo de infra-estrutura por conta da necessidade de se armazenar em memória de todos os correlacionamentos existentes entre os tokens gerados pelo Redhat SSO e o MAG de todos os usuário logados;

#### **4.3.3.Indicação de USO**

- Aplicativos com altos requisitos de segurança;
- Aplicativos com alta criticidade negocial e alta rentabilidade financeira.

**RASCUNHO**