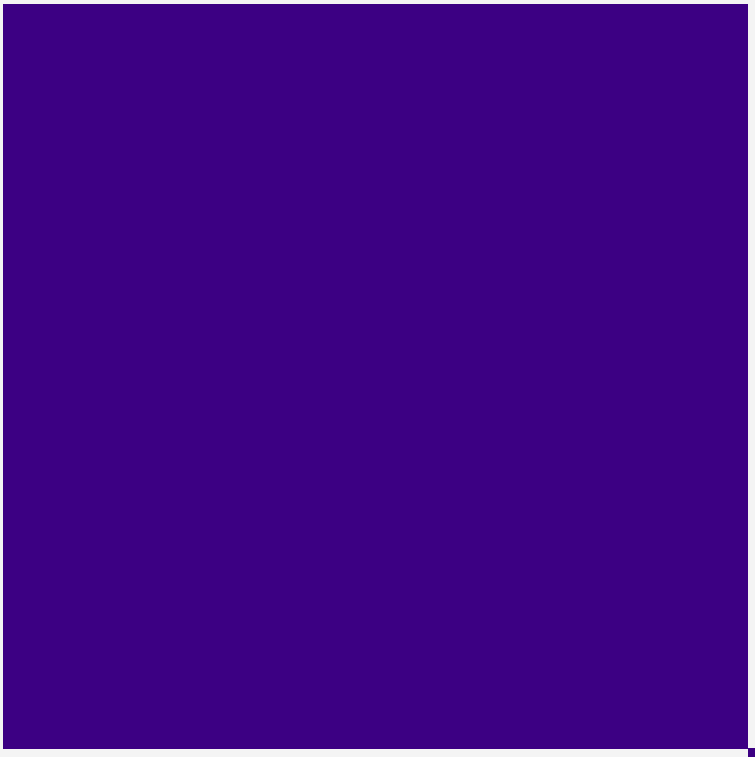
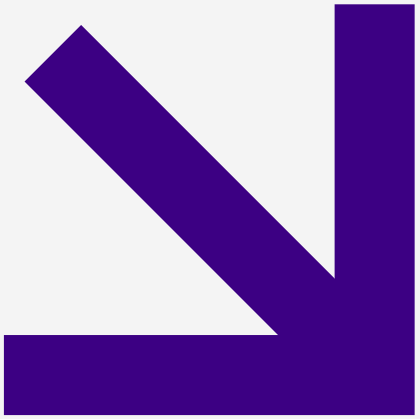
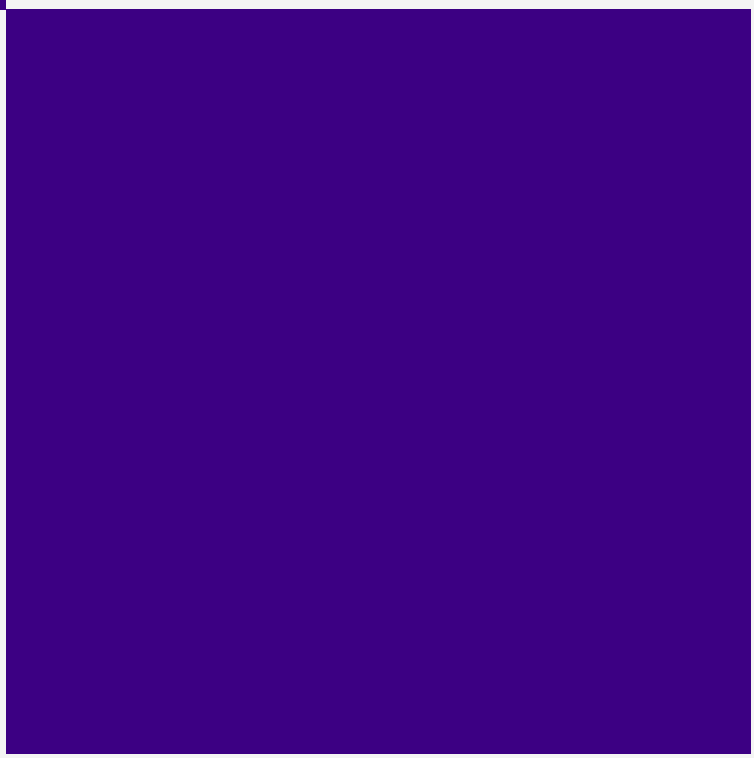
Open Banking no Brasil.

Uma visão sobre







Proposta de definições técnicas para construir um ecossistema seguro, aberto, eficiente, democrático, evolutivo e que beneficie o consumidor final.

nubank

Engenheiro líder do projeto:
Jonas de Abreu

Especialista:
Mariana Cunha e Melo

Contato:
openbanking@nu.bank

24 de Junho de 2020

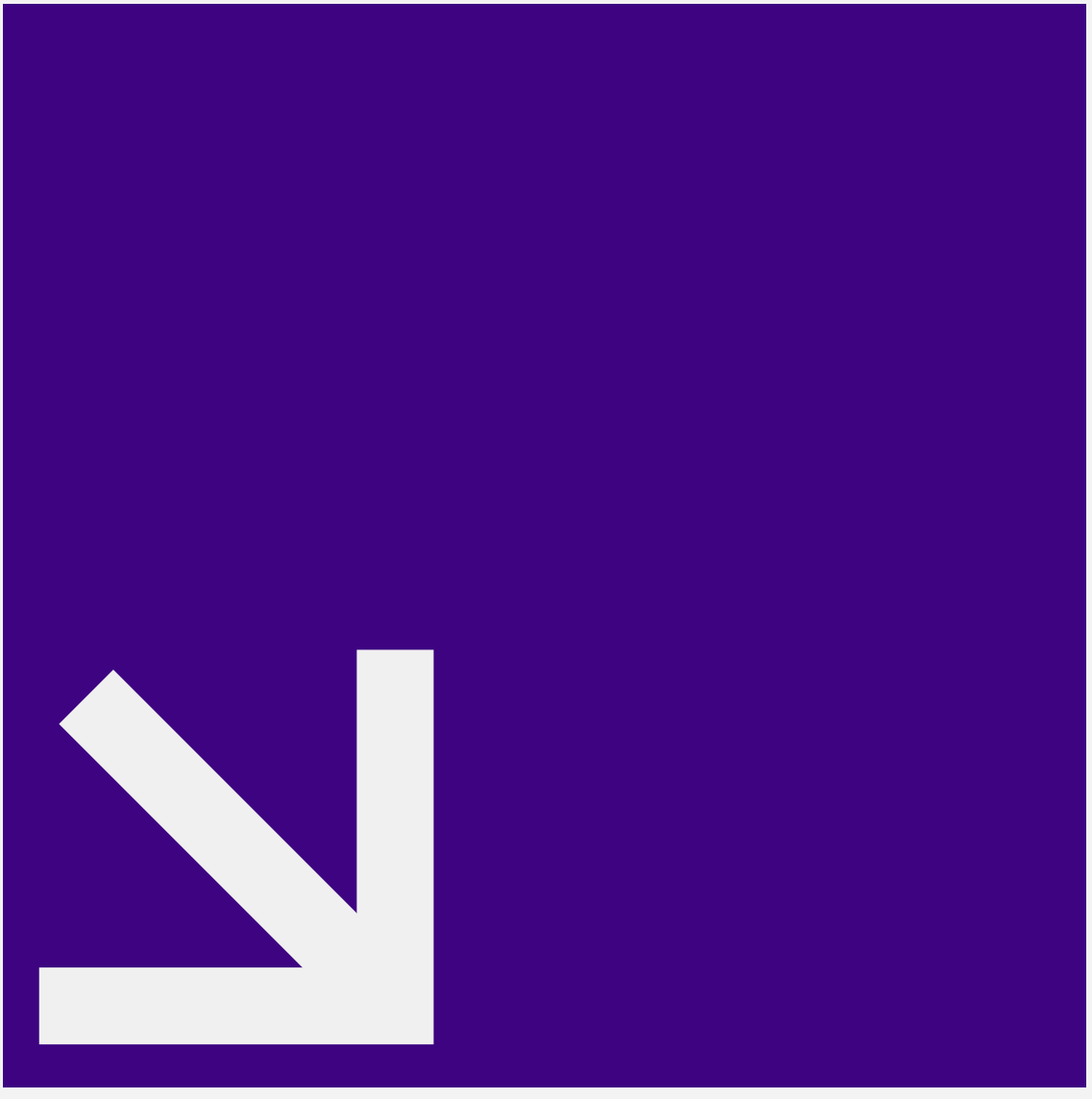
nu bank

índice

1

–

POR QUE UM DOCUMENTO ABERTO SOBRE OPEN BANKING?



A missão do Nubank é combater burocracias e devolver para seus clientes o controle sobre suas finanças. E o Open Banking tem o potencial de ser uma enorme ferramenta neste sentido. Pode reduzir assimetrias históricas do mercado financeiro, fomentar a competição (o que pode levar a melhores taxas e serviços aos clientes) e dar mais controle ao consumidor final sobre seus dados financeiros.

Para que esse grande movimento provoque, de fato, uma revolução no sistema financeiro e torne a vida de todos os clientes mais fácil e barata, é muito importante que o processo de auto-regulação seja guiado por alguns princípios.

AS DEFINIÇÕES DE TECNOLOGIA SERÃO DETERMINANTES PARA O NÍVEL DE SUCESSO DO OPEN BANKING.

Elas é que vão estabelecer o parâmetro de segurança do ecossistema, a qualidade da experiência do usuário, a eficiência das comunicações, a facilidade de adesão de novos entrantes, e até se o ecossistema ficará obsoleto a médio prazo ou poderá evoluir de acordo com as necessidades dos participantes e seus clientes.

AS DECISÕES QUE FOREM TOMADAS AGORA PODEM IMPULSIONAR OU RESTRINGIR A TRANSFORMAÇÃO DO MERCADO E DAS VIDAS DOS BRASILEIROS.

Carregando a força da inovação no nosso DNA e orgulhosos da responsabilidade que temos não apenas com nossos clientes de hoje, mas também com todos os brasileiros que serão impactados pelas nossas decisões hoje e além, apresentamos este documento público com nossa visão de tecnologia para garantir um ecossistema seguro, aberto, eficiente, democrático, evolutivo e que beneficie o consumidor final.

2

–

ESCOPO E ESTRUTURA

Nas próximas seções, vamos tratar das definições comuns para todos os padrões técnicos.

Conforme definido pelo Banco Central, o trabalho de todo o setor financeiro em desenvolver os padrões tecnológicos e procedimentos operacionais das APIs de Open Banking foi dividido em diversas fases.

É importante destacar que o desenvolvimento de cada um desses blocos de APIs não se dá de forma isolada. Pelo contrário, há uma série de definições que serão comuns e que devem compor um todo coerente. O escopo a que se propõe o presente documento é cuidar justamente dessas definições comuns de arquitetura e design.
De forma específica, serão tratados os temas:

01 _ Comunicações entre participantes e usuários finais;
02 _ Comunicações entre participantes;
03 _ Padrões específicos para segurança; e
04 _ Mecanismos para redução do custo de implementação

Antes de se passar às definições técnicas propriamente, contudo, **vamos apresentar no detalhe as propriedades gerais que entendemos ser indispensáveis para o sucesso do Open Banking no Brasil.** Tratam-se de princípios básicos que permeiam todas as nossas propostas técnicas. É não apenas o fundamento de todas as definições desenvolvidas na sequência, mas também os pontos centrais em volta dos quais entendemos que todo o debate deveria circular.

3

–

PRINCÍPIOS DE DESIGN

CINCO PRINCÍPIOS GERAIS ORIENTARAM NOSSAS PROPOSTAS TÉCNICAS ESPECÍFICAS E DEVEM TAMBÉM GUIAR TODO O PROCESSO DECISÓRIO NO ÂMBITO DO ECOSSISTEMA DO OPEN BANKING.

Em primeiro lugar:
a garantia da segurança das comunicações no âmbito do Open Banking deve ser considerada desde a concepção dos primeiros traços de arquitetura e do desenho das interfaces dedicadas. A grande promessa do Open Banking passa pelo fato de ser essa a primeira vez em que será possível construir canais que garantam a segurança das comunicações entre quaisquer dois participantes do ecossistema.

O que isso significa?
São preferíveis mecanismos de segurança criados de forma integrada no fluxo de comunicação, em vez de requisitos anexos às definições básicas de arquitetura. Devem ser privilegiadas também soluções que reduzam a complexidade da adoção dos requisitos, reduzindo o risco de que erros na implementação abram espaço para vulnerabilidades acidentais.

Em segundo lugar:
Open Banking tem o potencial de operacionalizar o direito de autodeterminação informativa das pessoas, ao menos no que diz respeito aos seus dados financeiros. E a tecnologia deve trabalhar a serviço da concretização desse potencial.

O que isso significa?
São preferíveis definições que reduzam o atrito para o usuário final e potencializem sua liberdade de escolha. Assim como os mecanismos de garantia de segurança, as considerações sobre experiência do usuário devem ser levadas em consideração a cada decisão tomada na especificação técnica e nunca de forma apartada, para garantia de uma padronização coerente e consistente com os princípios que se busca preservar.

Em terceiro lugar:
as APIs devem ser eficientes. Os potenciais transformadores de Open Banking aumentam com cada pessoa ou empresa que pode fazer uso de Open Banking. A eficiência reduz o custo efetivo de participação, catalisando esse processo.

O que isso significa?
São preferíveis: (i) definições de arquitetura que reduzam a quantidade e o tamanho de requisições e de respostas, garantida a qualidade do resultado o cliente final; (ii) uso de protocolos abertos em vez de proprietários; (iii) redução do número de consultas a bancos de dado necessárias para completar a resposta a uma requisição.

Em quarto lugar:
a integração deve ser facilitada. O Open Banking tem um grande potencial de aumentar a competição pela qualidade e preço de serviços, bem como de abrir o mercado para novas formas de consumir esses serviços. E é a padronização técnica do Open Banking que definirá a complexidade de se participar desse ecossistema. Nesse contexto, a padronização deve conter especificações suficientes e adequadas para reduzir o custo de integração entre os participantes.

O que isso significa?
São preferíveis: (i) protocolos e desenhos que reduzam a complexidade técnica da integração; (ii) definições que deem menos espaço para divergências na implementação da especificação técnica; (iii) construção de mecanismos de mitigação do risco de implementações inconsistentes.

Em quinto lugar:
é importante também que o sistema seja evolutivo, extensível e, na medida do possível, flexível. Isso por duas razões. Quanto à primeira, já que a padronização técnica será responsável por absorver a complexidade de integração de quaisquer dois participantes do ecossistema, é importante que essa padronização não restrinja desnecessariamente a quantidade de casos de uso que podem ser construídos sobre essas conexões.

Ou seja: a especificação deve deixar espaço para que os participantes criem produtos que não foram sequer concebidos ao tempo da publicação da especificação. Já a segunda razão é que essas APIs devem sobreviver ao teste do tempo. Isto é: devem permitir sua atualização de forma menos abrupta. A propriedade da extensibilidade permite que o núcleo fundamental do que for definido hoje seja reaproveitado para prover mais dados e informações no futuro, sem necessariamente passar mais uma vez por longos processos de padronização.

O que isso significa?
São preferíveis protocolos e desenhos extensíveis e mais simples. Devem ser evitados desenhos que promovam a especificação excessiva de campos e funções, que amarram excessivamente a possibilidade de uso das especificações hoje e no futuro.

Esses princípios gerais se refletem em algumas escolhas fundamentais que foram tomadas ao longo desta especificação técnica.

–

DECISÕES FUNDAMENTAIS

4

M-TLS

Segurança é o grande habilitador para a comunicação entre quaisquer dois participantes do ecossistema e todo o potencial do Open Banking. Como se sabe, a segurança da informação é um domínio complexo e demanda profissionais com conhecimentos extremamente específicos para garantir a eficácia dos controles e ferramentas empregadas.
A escolha dos protocolos básicos de comunicação no Open Banking, no entanto, pode ter o efeito fundamental de reduzir a exigência de conhecimento específico para os receptores de dados e elevar a segurança do sistema como um todo.

Essa é a grande vantagem de se adotar o TLS Mutuamente Autenticado (mTLS) para ser a primeira camada de proteção desse ecossistema. Além de dar garantias semelhantes a de uma Rede Virtual Privada - VPN a cada conexão estabelecida, sua própria estrutura reduz drasticamente o risco de implementações equivocadas. Além disso, ele ainda pode ser utilizado em conjunto com outros protocolos como OAuth 2.0, provendo defesas contra muitas dificuldades de implementação correta e efetivamente segura desses protocolos.

INTERFACE DE GESTÃO DO CONSENTIMENTO

Devolver o poder aos consumidores finais é fundamental, para que tomem as melhores decisões de acordo com seus interesses. Para esse fim, além de comunicação clara dos riscos de suas ações, entendemos como fundamental que ele possua total controle sobre quem pode acessar seus dados e como tal acesso pode ser feito.
Os Transmissores de dados devem oferecer aos seus clientes visibilidade e controles sobre todos as permissões concedidas para terceiros, incluindo opções para que seus clientes unilateralmente revoguem ou modifiquem as permissões concedidas a esses terceiros. Tais controles podem ser oferecidos como um dashboard de fácil acesso.
Notificação da concessão de permissão também é fundamental para que usos indevidos sejam detectados o quanto antes.

MECANISMOS DE OTIMIZAÇÃO DE REQUISIÇÕES E RESPOSTAS DE APIS

A eficiência de custos deve ser buscada a todo momento. Como se sabe, o custo operacional da comunicação via APIs funciona, em alguma medida, de forma espalhada. Ao custo de se receber e processar uma requisição e transmitir dados corresponde um custo de enviar a requisição, receber e processar a resposta. Deve haver, portanto, um alinhamento de interesses para que esses custos sejam os menores possível. Transmissores não devem ser obrigados a manter uma infraestrutura excessiva e Receptores não devem ser obrigados a reprocessar dados que já receberam no passado. Minimizar repetição de envio de dados e reduzir a variância de recursos utilizados por uma requisição em si facilitam o planejamento de capacidade dos Transmissores e efetivamente reduzem os custos operacionais internos. Como Transmissores só são autorizados a cobrar o custo marginal de cada requisições de API, repassar essas economias aos Receptores de dados alinha seus incentivos para que optem pelas formas mais eficientes.

PROTOCOLO HTTP, ARQUITETURA RESTFUL, FORMATO JSON

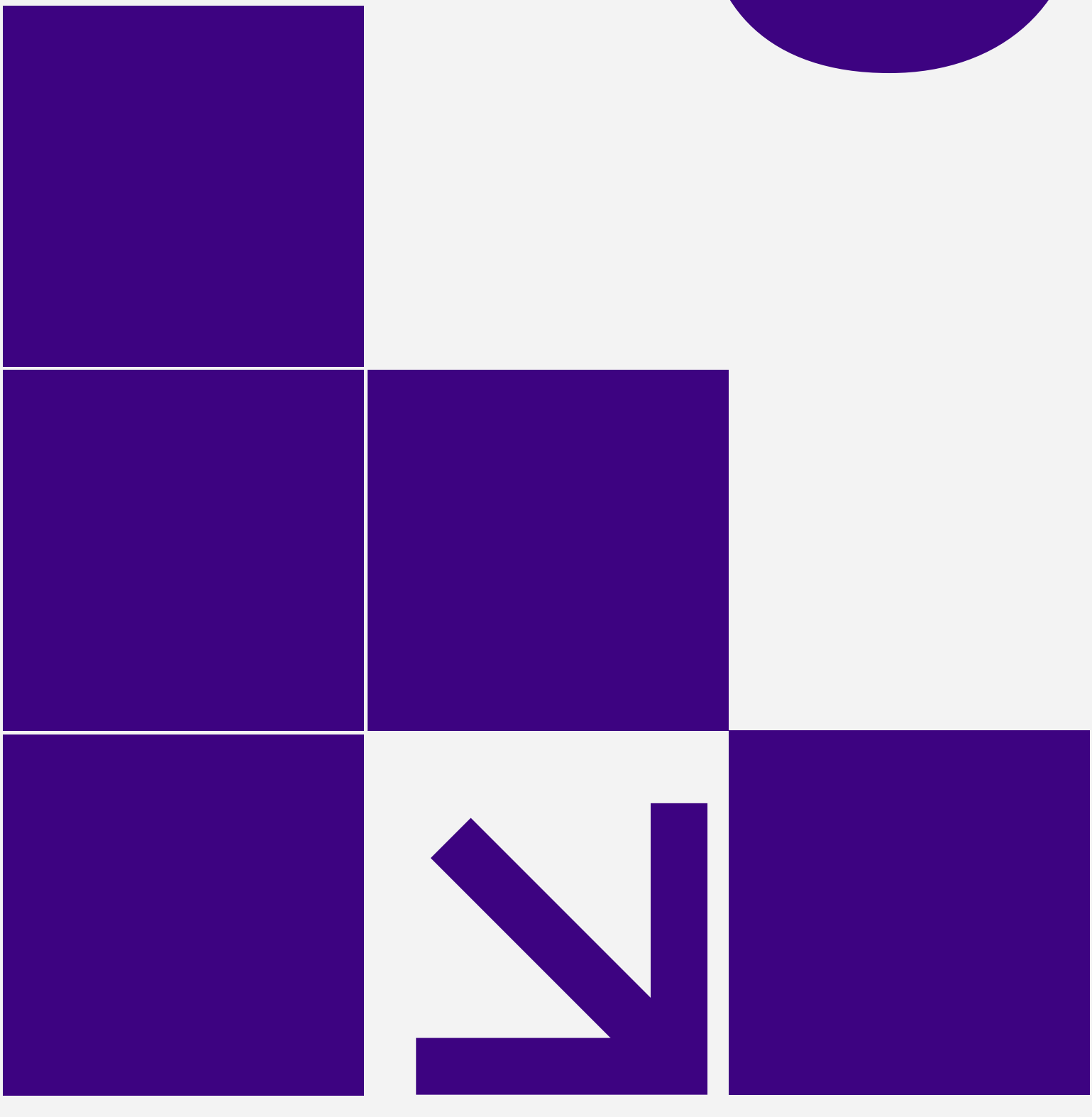
Facilitar a adoção do Open Banking, facilitando a criação de integrações é possível ao utilizar tecnologias de ampla difusão. HTTP, RESTful e JSON se consolidaram como a escolha preferida para o desenvolvimento de APIs - e não apenas na indústria financeira. Com o suporte de schemas de validação, exemplos de requisições e respostas e ambientes de homologação é possível acelerar de forma expressiva essas integrações. E, ao mesmo tempo, facilita-se que as implementações sejam consistentes, reduzindo necessidade de modificações especiais para acessar os dados em cada Transmissor.

MECANISMOS DE EXTENSÃO HTTP E LEITURA TOLERANTE.

Construir um sistema financeiro aberto para o futuro, ou seja, deixar aberto também o caminho para inovação futura. Nossa proposta se vale dos mecanismos de extensão próprias do protocolo HTTP e incorpora outros mecanismos, como a chamada Leitura Tolerante *(Tolerant Reader)*.

5

PADRONIZAÇÃO DA COMUNICAÇÃO ENTRE AS INSTITUIÇÕES E O USUÁRIO FINAL



O processo de solicitação de compartilhamento é um ponto crítico tanto para a segurança quanto para a facilidade de uso de Open Banking pelos usuários finais. É necessário, portanto, equilibrar muito bem ambas características de forma a dar a melhor experiência segura ao usuário final.

Nossa proposta é que o fluxo seja preferencialmente feito entre aplicativos móveis possuindo um mecanismo de fallback automático para o uso de um browser caso o usuário final não possua o aplicativo do Transmissor instalado em seu dispositivo. Essa é uma circunstância comum para usuários com dispositivos mais simples, com menor capacidade de armazenamento.

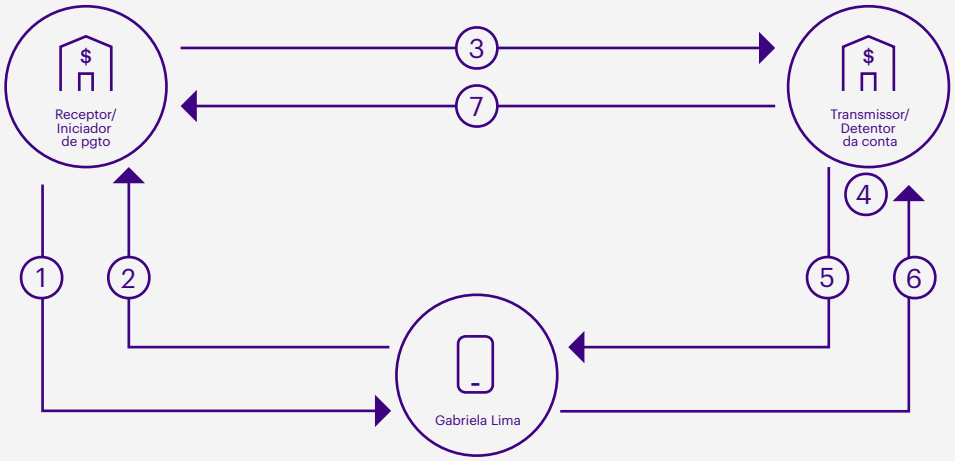
O fluxo preferencial via aplicativo permite que tanto Transmissores como Receptores façam uso de técnicas mais poderosas de autenticação, normalmente não disponíveis em um browser, melhorando a segurança do sistema. Na sequência, confira nossa proposta de fluxo padrão, seu fallback, bem como as definições específicas sobre as etapas de consentimento, autenticação e confirmação.

FLUXO DE SOLICITAÇÃO DE COMPARTILHAMENTO

Para podermos dar a experiência mais simples e segura para o usuário final, é importante podermos executar todo o fluxo de solicitação de compartilhamento a partir de aplicativos do Receptor e Transmissor de dados (o mesmo vale para o Iniciador de Pagamento e Detentor de Conta). É importante também que o fluxo determine que, da tela de consentimento na interface do Receptor, o usuário seja remetido ao aplicativo do Transmissor em uma tela já específica para a confirmação. Por fim, como não é possível garantir que ambos aplicativos estejam instalados no dispositivo do usuário final, é importante que exista um fallback para que o usuário conclua a operação pelo browser. As tecnologias que melhor se atendem a essa necessidade são App Links, para dispositivos Android, e Universal Links, para dispositivos iOS.

Confira abaixo a representação do fluxo completo de solicitação de compartilhamento:

Confira abaixo a representação do fluxo completo de solicitação de compartilhamento:



01	Apresentação das informaçoes previstas em regulação para consentimento do usuário final
02	Consentimento
03	Acesso ao universal link/app link
04	Autenticação
05	Apresentação dados na tela específica, seguindo link
06	Confirmação do consentimento
07	Entrega de credenciais

Nossa proposta para a solicitação de compartilhamento é que seja utilizado um fluxo tradicional de autorização tripartite com algumas pequenas adaptações para permitir o seu funcionamento para comunicação entre aplicativos com um fallback automático para browsers. Esse tipo de fluxo normalmente é associado ao protocolo OAuth, mas em nossa proposta ele é utilizado para a entrega de certificados digitais.

Na etapa 1, o Receptor apresenta a solicitação de consentimento ao cliente final, contendo os dados ou serviços que serão objeto de compartilhamento, bem como as demais informações exigidas pela regulação (como a o prazo de validade). Caso o cliente consinta (etapa 2), o aplicativo do Receptor pede, por meio de App Links ou Universal Links, que o dispositivo acesse um URL do Transmissor para (etapa 3) abrir a tela específica do aplicativo para iniciar o processo ininterrupto e contínuo de autenticação e confirmação.

Para isso, é importante que, ao chamar a URL do Transmissor, o Receptor inclua em seus parâmetros: (i) os dados ou serviços que serão objeto de compartilhamento e o prazo de validade do consentimento, bem como (ii) uma URL do Receptor de retorno pro aplicativo do Receptor e (iii) uma URL do Receptor para entrega das credenciais.

O Transmissor autentica o seu cliente (etapa 4) e apresenta os dados da solicitação para revisão (etapa 5). Caso o cliente confirme o consentimento (etapa 6), o Transmissor registra esse consentimento em seus sistemas e pede que o dispositivo acesse a URL de retorno para o aplicativo do Receptor e envia as credenciais para a URL de entrega de credenciais (etapa 7), ambas recebidas na etapa 3 (itens ii e iii).

As etapas 2, 4 e 6 do fluxo receberam disciplina específica da regulação e merecem ser detalhadas em maior profundidade. É o que se passa a expor.

CONSENTIMENTO

Na etapa do consentimento, o Receptor deve apresentar o seu prazo de validade, o Transmissor/Detentor de Conta, o cliente final, os dados objeto de compartilhamento, bem como as finalidades do uso dos dados. Entendemos que a forma de estruturar essas informações deva ficar a critério das instituições Receptoras, observados os requisitos legais e regulatórios. O único ponto que merece atenção na padronização técnica, por determinação da Resolução Conjunta nº 01/2020: os critérios de agrupamento dos dados objeto de compartilhamento e, portanto, a granularidade que pode ser empregada no processo de consentimento.

A granularidade do consentimento é um ponto fundamental da experiência que os clientes terão com Open Banking. Granularidade excessiva pode confundir o usuário final e levá-lo a ignorar os itens listados para consentimento. A falta de granularidade, por outro lado, também não é adequada, pois usuários finais não teriam alternativa senão compartilhar conjuntos mais abrangentes de dados, o que poderia levar a uma exposição desnecessária do usuário.

A definição da granularidade do agrupamento de dados que serão objeto de compartilhamento depende de dois elementos: (i) a identificação dos dados que o Receptor pleiteia acesso; e (ii) a aceitação dos termos pelo usuário final. E, aqui, vale voltar ao ponto sobre o objetivo básico da padronização técnica do Open Banking, que é a garantia da comunicação entre dois quaisquer participantes. Nesse sentido, o foco deve ser em permitir que as requisições de acesso a dados e serviços, enviadas pelos Receptores aos Transmissores, possam ser processados no grau adequado de granularidade.

Quanto ao agrupamento dos dados e serviços na apresentação ao consumidor, entendemos que a padronização técnica deve definir um agrupamento máximo, ficando aberta ao Receptor a opção de esmiuçar mais detalhadamente os dados que serão objeto de compartilhamento. Naturalmente, esse espaço de conformação é limitado pelos deveres impostos pela legislação específica. Mas entendemos que aqui os controles devem ser feitos em cima das políticas impostas aos participantes e não da arquitetura técnica, a bem da evolutibilidade do sistema.

Nos próximos tópicos será detalhada nossa proposta. Em resumo, o usuário final poderá consentir com o compartilhamento de três classes de dados, bem como os serviços específicos a que desejar conceder acesso. As três classes de dados são: (i) dados cadastrais, cujo consentimento pode ser concedido em um único bloco, (ii) dados personalizados do produto; e (iii) dados circunstanciais de uso do produto. Quanto à iniciação de pagamento, deve ser destacado cada produto objeto de consentimento: débito em conta, transferências entre contas na própria instituição, TED, PIX, DOC, boletos e outros que venham a ser incluídos no futuro. Passa-se a explicar cada um deles.

DADOS CADASTRAIS

Para facilitar a leitura dos dados objeto de compartilhamento, entendemos que os dados cadastrais - tanto os de identificação quanto os de qualificação - devem poder ser todos agrupados sob a mesma denominação de dados de cadastro. A comunicação do Receptor com o Transmissor, no entanto, deve poder contemplar pedidos de dados específicos dentro dessa categoria geral.

Assim, caso um Receptor deseje acessar o nome, email e telefone do cliente, deve poder indicar essa restrição na requisição ao Transmissor. Mas é importante que o ato de consentimento do cliente possa recair sobre o conjunto dessas informações e não individualmente.
Essa modelagem permite que Receptores minimizem o acesso aos dados cadastrais, e também parece ser uma granularidade razoável, pois não deixa excessivamente opacos os dados a serem compartilhados nem onera o usuário final com excesso de informação.

Para refletir esse design, a especificação técnica deverá contemplar uma modelagem do próprio payload enviado na comunicação entre o Receptor e o Transmissor para permitir essa customização dos dados solicitados. O ponto será retomado adiante. Por ora, basta o registro de que os campos desse payload deverão ser opcionais na primeira comunicação do Receptor ao Transmissor (para que esse tenha a liberdade de especificar os dados a que quer ter acesso), mas, após a confirmação pelo usuário final, os campos que foram selecionados passam a ser de preenchimento obrigatório pelo Transmissor na resposta às requisições do Receptor.

DADOS DE PRODUTOS

Para acesso aos dados transacionais de cada produto, entendemos que é necessário que sejam destacados ao menos dois conjuntos de dados: (i) dados personalizados do produto; (ii) dados circunstanciais do uso do produto. Da mesma forma, o Receptor poderá optar por enviar pedidos de acesso a uma ou ambas as classes de dados, conforme a necessidade de seu caso de uso.

Os dados personalizados do produto se diferenciam dos dados abertos sobre o produto pois se tratam de informações para um cliente específico. São dados como saldo, limite de crédito, encargos, data do vencimento de fatura, etc. Já os dados circunstanciais do uso do produto são aqueles que permitem inferências sobre o comportamento da pessoa. Ou seja: informações sobre os seus hábitos de consumo e os estabelecimentos que frequenta, que poderiam vir a revelar informações sensíveis sobre o indivíduo. Essa classe de dados pode ser mais facilmente identificada, caso haja a presença de um campo para descrição do destinatário ou do originário da transações (pagador e recebedor).

No escopo atual do Open Banking, esses seriam as “transações de crédito e de débito realizadas”, os “débitos e pagamentos autorizados” e as “transações de pagamento realizadas”.

Essa distinção é importante porque nem todos os casos de uso demandarão acesso aos dois conjuntos de dados. Parece relevante, portanto, que seja possível aos Receptores pedirem acesso a apenas uma ou a ambas classes de dados, conforme o caso. Aqui vale destacar que o propósito não é dificultar a transmissão dos dados circunstanciais - é direito do usuário final compartilhá-los de forma conveniente. O objetivo desse agrupamento é facilitar a adequação do pedido de acesso a dados à necessidade de cada caso de uso. Essa distinção, vale lembrar, só se reflete no agrupamento dos dados objeto de compartilhamento na apresentação ao usuário final para obtenção do consentimento.

Sob o ponto de vista do design da padronização técnica, faz sentido que haja APIs específicas para acesso a cada uma dessas classes de dados, para facilitar o controle e a operação da obtenção desses dados pelos Receptores (o modelo, portanto, é de acesso a tudo ou nada em cada uma dessas APIs).

INICIAÇÃO DE PAGAMENTO

Para iniciação de pagamento, entendemos que o consentimento deve ser concedido de acordo com a modalidade de pagamento. Isso permitirá que o cliente final defina que um Receptor pode, por exemplo, iniciar TEDs e PIX, mas não iniciar pagamento de boletos.

Mais uma vez, é importante ressaltar que deve ser possível pedir múltiplos consentimentos e receber autorizações em um única operação, não sendo necessário repetir o fluxo múltiplas vezes.

Vale notar que aqui também uma modelagem de APIs específicas para cada meio de pagamento parece ser a mais adequada.

AUTENTICAÇÃO

Quanto ao processo de autenticação dos clientes por parte do Transmissor de dados, entendemos que esse é um dos casos em que a padronização restringiria excessivamente a liberdade dos participantes definirem seus processos. A Regulação do Banco Central já impõe o ponto fundamental de que os Transmissores devem fazer uso de mecanismos de autenticação que sejam compatíveis com os já utilizados pelos Transmissores para operações em seus ambientes, inclusive sob o ponto de vista dos fatores de autenticação, quantidade de etapas e duração. Entendemos ser importante o controle para que não se crie barreiras indevidas ao compartilhamento dos dados pelos usuários finais. A tecnologia a ser empregada, no entanto, não deve ser padronizada, para deixar espaço para a evolução da técnica e ampliação da segurança e da usabilidade do sistema.

Caso o cliente não esteja ainda autenticado e seja necessário efetuar o login, deve ser obrigatório que após o login o cliente seja imediatamente levado para a tela de confirmação, evitando que ele tenha que reiniciar o processo de solicitação de compartilhamento e que ele seja obrigado a navegar pelo ambiente do Transmissor até encontrar o local adequado. Como já descrito na apresentação do fluxo completo de solicitação de compartilhamento, essa experiência é garantida por meio do uso de App Links/Universal Links.

CONFIRMAÇÃO

Para o processo de confirmação, é fundamental que se reapresente quais acessos a dados serão concedidos e a validade desses acessos utilizando a mesma linguagem e categorização utilizada na etapa de consentimento, para evitar confusões.

Embora seja interessante estabelecer recomendações genéricas sobre a forma de apresentação das informações na tela de confirmação, entendemos que não deve ser estabelecida uma padronização do fluxo dentro do aplicativo do Transmissor. Afinal, haverá aqueles que farão a autenticação do usuário antes de se apresentar a tela de confirmação e haverá aqueles que farão a autenticação após a tela de confirmação. O que é essencial neste ponto - e já estabelecido na regulação - é que as etapas necessárias para solicitação de compartilhamento sejam feitas de forma sucessiva, ininterrupta, com segurança, agilidade, precisão e conveniência e exclusivamente por canais eletrônicos.

ENTREGA DO CERTIFICADO COMO ÚLTIMA ETAPA DA CONFIRMAÇÃO

Ao final do processo de confirmação deve ser entregue ao Receptor uma credencial para que ele possa fazer requisições as APIs de Open Banking. A nossa proposta é que nesse momento seja entregue um certificado digital assinado e gerido pelo Transmissor. Tais certificados, embora utilizem a mesma tecnologia base que os certificados digitais emitidos nas cadeias de certificado do ICP-Brasil não exigem as mesmas restrições exigidas para uma Autoridade Certificadora aprovada pelo ICP-Brasil. A forma específica de implementação e uso desses certificados é discutida com detalhes na seção sobre TLS Mutuamente Autenticado.

GESTÃO DO CONSENTIMENTO

Dado que Open Banking define diversos mecanismos de comunicação entre máquinas do Receptor e do Transmissor é bastante fácil que com o passar do tempo o usuário final não consiga ter clareza de quais Receptores possuem quais consentimentos.

Para dar transparência e garantir que o usuário possa, a qualquer momento, modificar ou revogar consentimentos, é fundamental que o Transmissor ofereça aos seus clientes uma interface onde conste necessariamente todas as autorizações presentes e passadas, contendo opções de revogação.

Também deve ser facultado ao Transmissor oferecer informações adicionais em tais interfaces, que possam auxiliar na tomada de decisão sobre manter ou revogar essa autorização.

6

PADRONIZAÇÃO DA COMUNICAÇÃO ENTRE AS INSTITUIÇÕES

Além da padronização de todas as requisições, formatos de requisição e resposta, é importante que a padronização também leve em consideração o custo típico de uma chamada de interface. Naturalmente, esse custo varia entre as diversas arquiteturas adotadas por Transmissores de dados e idealmente tal análise seria capaz de demonstrar impacto em cada uma dessas arquiteturas. No entanto, tal análise seria inviável, sendo necessário um proxy para tal informação.

Com base em latência típica de operações, optamos por utilizar a **busca por dados em um banco de dados** como o proxy para o custo típico de uma requisição. Independente da arquitetura adotada pelo Transmissor, existe um mecanismo de armazenamento de dados de longo prazo e ele tende a ser uma das operações mais caras executadas em requisições para APIs de acesso a dados.

Para demonstrar de forma mais simples o impacto de possíveis alternativas de padronização, vamos assumir como unidade de medida o número de Unidades de Dados Financeiros (UDF) que forem acessadas pelo Transmissor e entregues ao Receptor de dados.

Uma Unidade de Dados Financeiros pode ser composta por uma operação transacional, um conjunto de dados cadastrais, os dados personalizados de um produto ou qualquer outro conjunto que possua uma relação contextual entre si, mas não tenha como característica inerente a sua modificação constante de acordo com o uso do produto financeiro. Ou seja: informações sobre o limite de crédito total de um produto, com a discriminação de seus valores utilizado e disponível configura uma UDF. O extrato transacional do produto, contudo, por sua característica de lista que é atualizada conforme o uso do produto, tem em cada item dessa lista uma UDF.

Essa distinção é importante para se aplicar os mecanismos mais adequados de otimização para cada caso. **Com isso, o princípio de eficiência das APIs pode ser descrito como (i) minimizar o número de UDFs transmitidos na relação entre Transmissor e Receptor e (ii) reduzir a variância na quantidade de UDFs transmitidos em uma única requisição.**

DADOS TRANSACIONAIS

Os dados do histórico transacional representam possivelmente a maior parte dos dados que serão transmitidos aos Receptores. Eles também apresentam a característica de aumentar substancialmente em volume ao longo da relação com os clientes, o que exige maiores cuidados de eficiência no desenho das APIs referentes a eles.

Nos itens abaixo apresentamos três formas de deixar a transmissão de tais dados mais eficiente.

PAGINAÇÃO

Paginação é fundamental para reduzir a variância da quantidade de UDFs transmitidas em uma requisição. Transmitir 500 UDFs em 40 vezes tende a ser mais eficiente que transmitir 20000 UDFs de uma única vez porque exige menos recursos por requisição, o que distribui melhor a carga nos servidores, sendo até possível responder via múltiplos servidores independentes.

Vale ressaltar que embora 20000 UDFs pareça grande, contas de Pessoas Jurídicas podem ter muito mais UDFs em apenas um dia.

Considerando que é de interesse do consumidor que todos os seus dados transacionais possam ser enviados ao Receptor, seria restritivo demais estabelecer que cada requisição paginada corresponda a uma requisição cobrada, porque isso inviabilizaria as gratuidades estabelecidas pela resolução de Open Banking.

Levando isso em consideração, nossa proposta de paginação para dados transacionais é construída sobre dois pilares: Capability Urls e o Cabeçalho Link.

O fluxo geral pode ser descrito como:

01 _ O Receptor faz uma requisição pedindo o histórico transacional do cliente do Transmissor. A URL dessa requisição é preditiva, definida na especificação e passível de cobrança caso realizadas em volume superior ao estabelecido nas gratuidades.

02 _ O Transmissor devolve na resposta os primeiros 200 UDFs daquele histórico específico no cabeçalho Link uma referência com a URL que contém os próximos 200 UDFs. Essa URL precisa ser uma Capability URL.

03 _ O Receptor que está interessado nos dados além dos primeiros 200 UDFs, faz a requisição para essa Capability URL. Essa URL não é predizível, então não existe uma forma do Receptor descobrir a URL. Dado que essas requisições são apenas para receber a continuação dos dados do passo 1, essa URL não seria passível de cobrança.

04 _ Da mesma forma que no passo 2, o Transmissor devolve os próximos 200 UDFs e novamente um Cabeçalho Link contendo a Capability URL do próximo conjunto.

05 _ Esse fluxo é repetido até acabarem os UDFs associados àquele histórico transacional ou o Receptor não ter mais interesse em receber os dados e parar de acessar as Capability URLs recebidas.

Um ponto importante a se definir é que, embora Capability URLs normalmente não sejam autenticadas, considerando a sensitividade dos dados transmitidos, convém também autenticar e autorizar tais URLs.

Outro ponto relevante: é possível implementar essa estratégia tanto de forma a manter o estado dentro do servidor (stateful) quanto sem manter estado no servidor (stateless), que tende a ser mais eficiente.

Embora não seja a única forma de se implementar tal estratégia, uma abordagem simples é encriptar de forma simétrica os parâmetros tradicionais de paginação (como página e tamanho da página), encodar esse conteúdo criptografado em base64 e colocar ele como um parâmetro da requisição. É também importante tomar cuidado para evitar a repetição de tais requisições, que poderia ser utilizada para burlar os mecanismos de cobrança.

MECANISMO DE PUSH DE INFORMAÇÕES

Umas das maiores possibilidades de aumento de eficiência do sistema é o envio proativo de UDFs transacionais. Devido à natureza desses dados, embora o histórico acumulado seja normalmente grande, as novas adições tendem a ser apenas um pequeno percentual do total de UDFs.

Para melhor entender o impacto dessa mudança, vamos analisar o caso de uma aplicação simples, que se vale das quatro requisições gratuitas por dia para acessar e manter atualizado os dados de um cliente. Esse cliente possui um perfil de em média fazer uma compra por dia (1 UDF por dia) e possui quase um ano de histórico (300 UDFs).

Como o objetivo do Receptor é manter os dados atualizado, ele vai requisitar o histórico a cada 6 horas, descartar todos os UDFS que já tiver (799) e manter apenas os UDFs novos (1). Podemos dizer que a eficiência em transmissão de UDFs é de apenas 0.125% (1/800).

Caso exista um mecanismo que proativamente envie para o Receptor os novos dados conforme chegam, a eficiência pode chegar bem mais próximo de 100% (não é possível ser 100% devido a possíveis re-tentativas e outras falhas de comunicação), garantindo que o Receptor receba quase que somente os novos UDFs.

Para aumentar ainda mais a eficiência desse processo, deixando de olhar apenas para os UDFs e passando a também observar o número de requisições, com o empacotamento de diversos UDFs nesses envios pró-ativos, o volume de requisições de envio também é substancialmente reduzido, reduzindo efetivamente necessidade de recursos para tais envios.

Vale lembrar que é possível empacotar até mesmo requisições de clientes diferentes que tenham autorizado o mesmo Receptor a receber seus dados, deixando ainda mais eficiente o processo.

Esse ganho de eficiência é ainda mais importante para contas de Pessoas Jurídicas, que tendem a apresentar um volume substancialmente maior de UDFs em seus fluxos diários.

Uma vantagem além do ganho de eficiência no processo de envio de UDFs é que a característica da entrega dos UDFs logo após a sua criação permite novas aplicações impossíveis anteriormente.

Uma dessas possibilidades é a criação de aplicações que monitorem e detectam fraudes transacionais, avisando o consumidor assim que ela acontecer, para que ele possa reagir e impedir danos maiores.

REQUISIÇÃO DAS DIFERENÇAS

Uma outra forma de aumentar substancialmente a eficiência do cenário descrito no item anterior é a padronização de requisições que transmitem apenas os UDFs gerados após um momento especificado em um parâmetro da requisição.

Uma forma simples de modelar isso seria definir um parâmetro opcional para a URL de acesso ao histórico transacional chamado, por exemplo, de "a-partir-de" que recebe uma data e horário (em formato ISO 8601) como valor.

Quando esse parâmetro é informado pelo Receptor, o Transmissor deve colocar no corpo da resposta apenas UDFs criados após a data e horário especificados.

Para se evitar erros no tratamento das datas é recomendado que data e horário sejam sempre especificados em UTC.

DADOS CADASTRAIS, DADOS PERSONALIZADOS DE PRODUTOS E OUTROS

Dados cadastrais e dados personalizados de produtos possuem características substancialmente diferentes dos dados transacionais. Eles tendem a mudar com bem pouca frequência, caso mudem.

Dada essa diferença de características, embora seja possível usar as mesmas técnicas de aumento de eficiência descritas para dados transacionais, convém utilizar técnicas mais simples para semelhante ganho de eficiência.

Nossa proposta é que esse ganho de eficiência seja atingido por meio do uso de requisições condicionais.

Dado que o principal benefício de requisições condicionais é para o Transmissor de dados, reduzir o custo ou não cobrar por requisições que não transmitam UDFs incentiva fortemente a adoção por parte dos Receptores.

REQUISIÇÃO CONDICIONAL COM CABEÇALHO IF-MODIFIED-SINCE

O Cabeçalho If-Modified-Since, quando utilizado na requisição Http, definirá que ela apenas deve devolver UDFs caso eles tenham sido modificados após a data especificada no cabeçalho.

Nos servidores do Transmissor é bastante direto uma implementação que controle, por meio de mecanismo de invalidação de cache, quando que os dados devem ser devolvidos e quando que não devem.

Essa mudança tem o potencial de até mesmo eliminar a necessidade do Transmissor de acessar os dados caso não tenha acontecido uma modificação, podendo ser construída por meio de notificações de invalidação de cache nos fluxos que efetivamente alteram tais dados.

REQUISIÇÃO CONDICIONAL COM CABEÇALHO IF-NONE-MATCH

Embora o Cabeçalho If-Modified-Since seja o que melhor se encaixa nesse uso específico, convém também mencionar o uso do Cabeçalho If-None-Match.

O Cabeçalho If-None-Match também faz com que a requisição seja tratada de forma condicional, mas ao invés de condicionar à uma data específica, ele condiciona a um valor anteriormente obtido em um Cabeçalho ETag.

Ele também é capaz de gerar os mesmos efeitos de eficiência descritos para o cabeçalho If-Modified-Since, mas calcular um Cabeçalho ETag tende a ser um pouco mais trabalhoso que controlar a data da última modificação de um conjunto de dados.

OPERAÇÕES DE INICIAÇÃO DE PAGAMENTO E OUTRAS OPERAÇÕES DE ESCRITA

Embora cada tipo de pagamento possua parâmetros e peculiaridades sobre sua forma de iniciação, é fundamental que todos eles possam ser iniciados de forma idempotente, prevenindo assim que falhas de comunicação deixem o ecossistema em estado inconsistente indefinidamente, com Receptores e Transmissores tendo visões diferentes sobre o estado do pagamento.

Para tanto, é fundamental que cada API de iniciação de pagamento seja (i) definida utilizando o verbo Http PUT e (ii) possua um identificador para verificações de idempotência. Normalmente, esse identificador será o identificador do recurso sendo criado por essa requisição e será tratado como parte da URL em uma API Rest.

Embora a única propriedade fundamental para que tal identificador funcione para verificações de idempotência é que ele seja único, é bastante recomendável que sejam utilizados Identificadores Universalmente Únicos (UUIDs) gerados aleatoriamente. Dentre as vantagens do uso de UUIDs está a impossibilidade se descobrir o identificador de um recurso, o que dificulta ataques cibernéticos contra o Transmissor. Também, dada a sua estrutura, é extremamente improvável que seja gerado o mesmo identificador para duas transações (necessário gerar 2127 UUIDs para que a possibilidade de encontrar dois iguais seja de 50%).

Tendo ambas as propriedades descritas acima, é possível ao Receptor, em qualquer caso de falha, re-executar a requisição de iniciação de pagamento sem receio de que o mesmo pagamento seja efetuado múltiplas vezes, garantindo a consistência eventual do sistema.

Além de modelar de forma idempotente, para iniciação de pagamento, devido aos diversos tipos possíveis de pagamento, é importante que o fluxo seja estruturado em duas etapas. A primeira etapa serve para o Iniciador de Pagamento recuperar informações sobre o pagamento para apresentar ao cliente final. A segunda, para efetivar tal pagamento.

Usando o pagamento de um boleto, como exemplo, o fluxo seria:

01 _ [Iniciador] De alguma forma obtêm as informações para iniciar o pagamento. No caso de um boleto seria o código de barras.

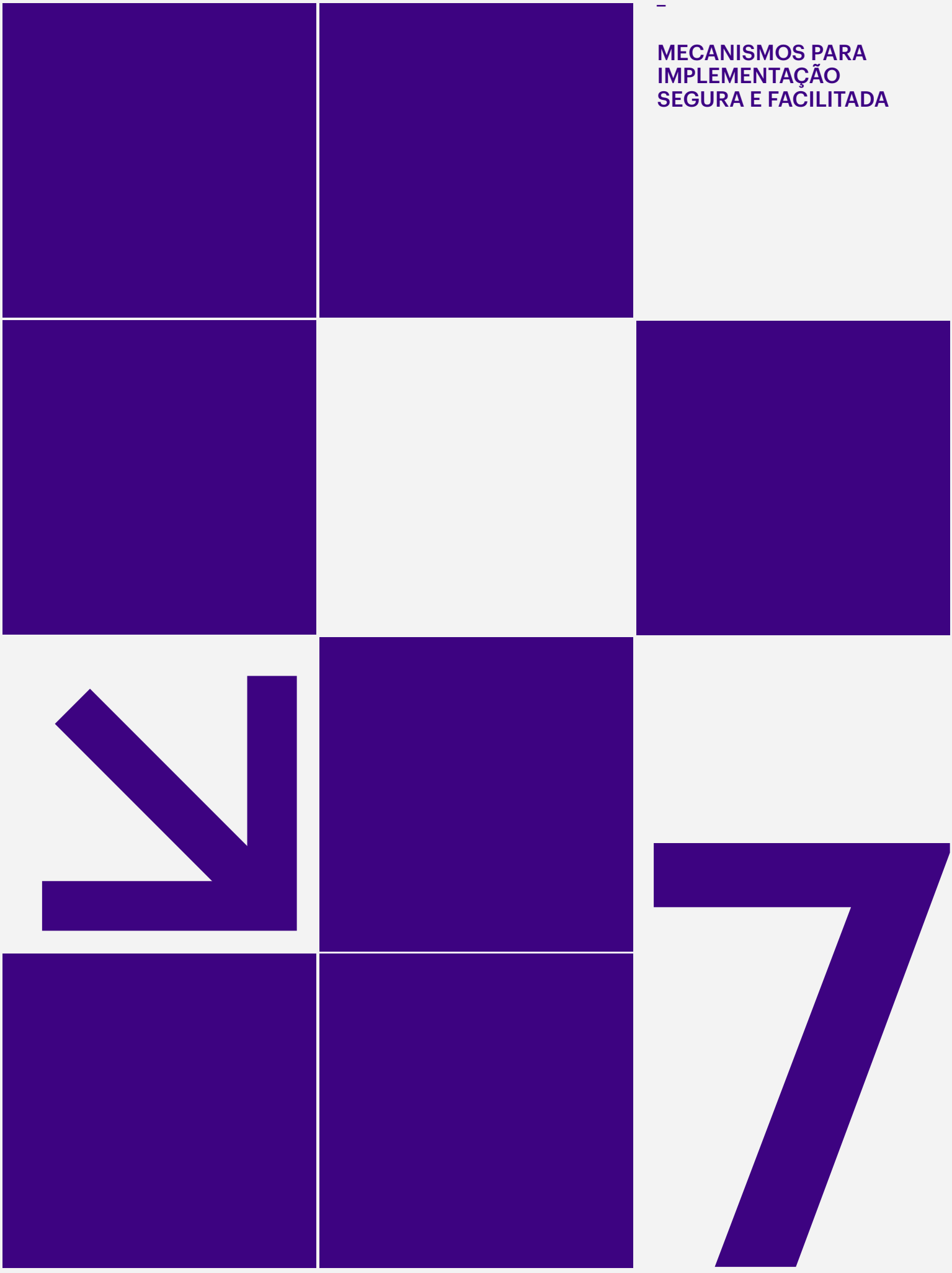
02 _ [Iniciador] Faz uma requisição para o Detentor da conta, para obter as informações relativas àquele pagamento.

03 _ [Detentor] Devolve as informações do pagamento. É possível modelar os dados de forma semelhante ao PIX e possuir apenas informações básicas especificadas (como o valor do pagamento e valor total cobrado do cliente) e ter todas as outras informações em uma lista de campos livres, que devem ser apresentado aos clientes. Isso reduz o esforço de se especificar cada tipo de pagamento.

04 _ [Iniciador] Recebe os dados e apresenta ao cliente final, podendo este cancelar ou efetuar o pagamento.

05 _ [Iniciador] Executa a operação escolhida pelo cliente final.

MECANISMOS PARA IMPLEMENTAÇÃO SEGURA E FACILITADA



PARÂMETROS NÃO SEQUENCIAIS EM URLS

Falhas em controle de acesso é uma das categorias de falhas mais comuns em aplicações web, o que pode ocasionar acesso indevido a dados de usuários.

Uma forma de adicionar uma camada de segurança extra é não utilizar parâmetros sequenciais em URLs, preferencialmente utilizando-se UUIDs, o que reduz a probabilidade de sucesso de ataques chamados de "controle quebrado de acesso", atualmente a quinta vulnerabilidade mais comuns em aplicações. Isso porque, ainda que o servidor do recurso falhe em verificar adequadamente a autorização de acesso, atacantes não conseguirão acessar o recurso, porque não saberão o identificador do recurso.

A nossa recomendação é que toda a padronização estabeleça URLs cujos parâmetros não sejam sequenciais ou preditivos de alguma forma. Para tanto, recomendamos o uso de UUIDs.

TLS MUTUAMENTE AUTENTICADO

TLS Mutuamente Autenticado, ou TLS Mútuo (mTLS) é uma forma alternativa do uso de TLS disponível em todas as versões de TLS.

Em uma conexão TLS normalmente utilizada, apenas o servidor apresenta um certificado que deve ser validado antes de completar a conexão. Podemos dizer que essa conexão é anônima no lado do cliente. O servidor não sabe quem abriu a conexão e não tem alternativa exceto aceitar a conexão.

Em uma conexão TLS Mutuamente Autenticada, além do servidor apresentar um certificado digital, o cliente também apresenta um certificado. Esse segundo certificado deve ser validado pelo servidor. O servidor apenas estabelece uma conexão caso esse certificado, que identifica o cliente, for válido. Isso impede conexões anônimas, que são a causa raiz para diversas vulnerabilidades, como ataques Man in the Middle ou de "downgrade" do protocolo.

Uma distinção importante a ser feita com relação aos certificados mencionados nesse documento é que eles não são certificados digitais emitidos nas cadeias de certificados do ICP-Brasil. São certificados válidos apenas no PSP emissor deles. A razão para isso é evitar a complexidade da gestão de certificados digitais do ICP-Brasil (uma complexidade necessária dado objetivo do ICP-Brasil) e evitar a necessidade de credenciamento de todos os Transmissores. É tecnicamente possível adequar a proposta desse documento aos certificados do ICP-Brasil, embora tal possibilidade não pareça de grande vantagem.

PROPRIEDADES

Para orientar a comparação entre alguns possíveis métodos de autorização, convém elencar as propriedades que o uso de TLS Mútuo trás.

01 _ Ataques "Man in the Middle" são substancialmente mais difíceis. Ao contrário do TLS anônimo, é necessário o comprometimento da credencial do servidor ou do cliente para que a comunicação seja interceptada.

02 _ Resistência a "downgrade" do protocolo. Tanto o servidor como o cliente verificam a consistência das mensagens trocadas durante o processo de handshake, incluindo os certificados apresentados. Mudanças nessas mensagens são detectadas pelo servidor ou pelo cliente, rejeitando a conexão.

03 _ Possibilidade de remover rapidamente conexões ofensoras. Dado que um ataque necessita de um certificado válido para iniciar, a simples revogação desse certificado impede que novos ataques sejam lançados, dando tempo às equipes de resposta a incidente corrigir a falha.

04 _ Separação de tráfego autenticado do não autenticado. Hoje, milhares de robôs varrem a internet procurando por servidores vulneráveis para iniciar ataques. De forma similar a uma VPN que impede que tais robôs consigam encontrar os serviços, o TLS Mútuo impede que se obtenha informações sobre o servidor sem uma credencial válida.

05 _ Credencial não é enviada ao servidor. Isso evita que bugs ou falhas no código do servidor exponha a credencial. Também não é possível acidentalmente se autenticar no servidor errado.

06 _ Capaz de funcionar para autorização. Embora certificados digitais sejam muito utilizados para identificação, também é possível atribuir permissões específicas a um certificado, seja utilizando alguma extensão com esse propósito ou mapeando no servidor quais operações um certificado pode executar.

07 _ Não repúdio. Uma propriedade interessante possível de se obter com o uso de certificados para se estabelecer uma conexão é a inforjabilidade da mesma. Do ponto de vista criptográfico, isso atribui a mesma confiança de uma assinatura digital à conexão.

COMPARAÇÃO COM O USO DE OAUTH

Embora OAuth seja um dos protocolos de autorização mais utilizados no mundo, a complexidade de se ter uma implementação segura é substancialmente maior do que modificar a conexão para se utilizar TLS Mútuo.

Das propriedades mencionadas de TLS Mútuo, dependendo de quais protocolos específicos são utilizados em uma implementação de OAuth, é possível apenas suportar as propriedades 5, 6 e 7. Dessas, a propriedade 5 exige que seja utilizado um mecanismo de assinatura no cliente e 7 exige que tal mecanismo de assinatura se valha de chaves assimétricas.

Vale mencionar que a maior parte das implementações de OAuth fazem uso de "token portador" (bearer token, em inglês), ou seja, o cliente envia em cada requisição a mesma credencial estática. Nesse modo de uso, OAuth é capaz de prover apenas a propriedade 6.

Dado que OAuth atua apenas na camada 7 (aplicação) do modelo OSI de comunicação de redes, OAuth não é capaz de fornecer nenhuma propriedade relativa às camadas mais baixas de comunicação.

COMPARAÇÃO COM O USO DE VPN

Embora o uso de VPN em conjunto com OAuth possa prover propriedades semelhantes ao uso de TLS Mútuo, o custo operacional seria substancialmente maior por 3 razões.

Primeiro, a VPN precisa estar em constante funcionamento tanto no Receptor quanto no Transmissor. Os requerimentos de monitoramento e capacidade técnica para tal execução tanto nos Receptores quanto Transmissores seria bastante alto, aumentando o custo efetivo dos serviços ao usuário final.

Segundo, normalmente a configuração de uma VPN exige trabalho de mudança no servidor do Transmissor, o que seria necessário para cada novo Receptor que queira se integrar.

Por fim, em terceiro lugar, soluções comerciais de VPN tendem a ter limites bastante restritivos do número de VPNs que podem estar conectadas simultaneamente. Dado que a VPN precisa estar conectada constantemente, isso praticamente impossibilita o uso por todos os possíveis Receptores no caso de Open Banking atingir o grande sucesso desejado por todos os participantes.

O uso de TLS Mútuo não é afetado por tais restrições, pois uma vez configurado, pode ser utilizado por qualquer número de conexões. E como o certificado é apresentado por conexão, tal conexão não precisa ficar ativa o tempo inteiro para o funcionamento, reduzindo os requerimentos de infraestrutura.

COMPARAÇÃO COM JWE

JWE é capaz de dar garantias semelhantes à TLS Mútuo, embora de forma mais trabalhosa e custosa.

Por TLS Mútuo ser uma forma diferente de se utilizar TLS, todo o trabalho necessário de configuração de TLS é reaproveitado no uso de sua variante mutuamente autenticada, exigindo pouco esforço adicional.

Já no caso de JWE, todos os parâmetros precisariam ser definidos, não sendo capaz de reaproveitar as definições do protocolo TLS.

Além desse esforço extra de especificação, ele também é substancialmente menos eficiente em termos de consumo de banda. Devido à necessidade de se representar qualquer requisição em formato JSON ou válido em URLs, JWE não é capaz de representar dados de forma arbitrária e precisa então encodar todos os dados em formato de base 64, aumentando o tamanho das requisições em 25%. Embora seja possível pensar em formas de mitigar esse aumento, como compressão dos dados, isso novamente adicionaria maior complexidade para um ganho que não seria do total de 25%.

DISTRIBUIÇÃO DE CERTIFICADOS

É possível criar um mecanismo de distribuição de certificados análogo aos utilizados para distribuir tokens OAuth. Para tanto, é necessário entender o processo de emissão e distribuição de um certificado.

O processo de distribuição de certificados normalmente envolve um processo chamado Requisição de Assinatura de Certificado (CSR da sigla em inglês), que possui os seguintes passos:

01 _ [Receptor] Gera uma chave privada e uma pública

02 _ [Receptor] Gera uma CSR e assina a CSR com a chave privada

03 _ [Receptor] Envia a CSR ao emissor

04 _ [Transmissor] Valida CSR e emite o certificado

05 _ [Transmissor] Envia o certificado para o cliente

A razão do processo funcionar dessa forma é garantir a propriedade 7, de não repúdio. Se a chave privada não pode ser descoberta e nunca saiu do controle do cliente, somente ele pode ter feito a requisição (ou assinatura). Essa é uma propriedade fundamental das assinaturas digitais nas cadeias do ICP-Brasil.

Para o caso de uso específico de Open Banking, essa propriedade de irrefutabilidade (7), embora interessante do ponto de vista sistêmico, não é fundamental e obter-se as propriedades de 1 a 6 já seria um ganho de segurança substancial quando comparado ao uso de OAuth (que poderia oferecer apenas 5 e 6)

Abrindo mão da propriedade 7, o processo de emissão pode ser adaptado para:

01 _ [Transmissor] Gera uma chave privada e pública para o cliente

02 _ [Transmissor] Emite o certificado

03 _ [Transmissor] Envia para o cliente o certificado e a chave privada para o cliente.

Notem que, da mesma forma que na distribuição de um token OAuth, o cliente apenas recebe a credencial pronta no final do processo, não precisando de conhecimento específico para a geração da credencial, substancialmente simplificando o processo.

Outro ponto notável é que não existe um impeditivo tecnológico para que essa abordagem simplificada coexista com a abordagem mais complexa, sendo possível o Transmissor, dado sua análise de capacidade técnica do cliente, optar por uma ou outra estratégia.

GESTÃO DE CERTIFICADOS PELO PSP

A gestão de certificados nos remete às restrições e técnicas exigidas para a gestão de um certificado de emissor pelo ICP-Brasil. Também não é necessário valer-se dessas técnicas e restrições.

Para a utilização de OAuth, normalmente o PSP teria que gerir uma chave (ou uma por cliente) de assinatura para os tokens. Processos semelhantes podem ser adotados para a gestão de certificados digitais.

Também existem projetos de código aberto e produtos comerciais para gestão de cadeias de certificados, o que facilitaria tais processos por parte dos Transmissores.

FACILITADORES PARA IMPLEMENTAÇÃO

É fundamental para a adoção de Open Banking que as implementações do padrão sejam feitas da forma mais estrita possível, garantindo que os Receptores consigam integrar uma única vez e conseguir se conectar com qualquer Transmissor.

Para tanto, apenas a especificação é insuficiente por não ser capaz de garantir uma correta implementação por parte dos Transmissores, criando APIs muito semelhantes, mas com pequenos desvios que dificultam a utilização pelos Receptores.

As técnicas a seguir auxiliam substancialmente na redução de inconsistências, sem incorrer de custos substancialmente elevados para o sistema.

AMBIENTE DE HOMOLOGAÇÃO PROVIDO POR TRANSMISSORES

A base para a construção de um sistema que assegure a consistência das implementações é cada Transmissor disponibilizar uma versão de suas APIs para uso em testes automatizados.

Tais APIs devem ser idênticas às APIs utilizadas no ambiente de produção, com a exceção que os dados devem todos ser falsos, embora respeitando as características de formato definida pela especificação. Preferencialmente tais APIs devem se valer do mesmo artefato de implantação utilizado no ambiente de produção, garantindo que credenciais sejam diferentes.

O processo para um Receptor obter acesso a tais APIs deve ser o mais simples e rápido possível, para que ele possa verificar possíveis inconsistências.

EXEMPLOS DE REQUISIÇÕES E RESPOSTAS

Esse ambiente de homologação deve possuir um conjunto de requisições e respostas esperadas, também para facilitar o desenvolvimento de forma consistente. Esse conjunto deve ser consistente entre todos os Transmissores.

É importante também distribuir tal conjunto de requisições e respostas de forma independente, para que Receptores consigam colocá-los como testes automatizados em suas infraestruturas, novamente auxiliando a consistência das implementações.

IMPLEMENTAÇÃO DE REFERÊNCIA

Também na linha de facilitar o desenvolvimento e teste para Receptores e Transmissores, convém existir uma implementação de referência das APIS, mantida pelo mesmo grupo mantenedor da especificação de Open Banking.

A implementação de referência tem, por definição, o objetivo de reduzir a subjetividade de uma especificação, criando um produto concreto onde podem ser testadas dúvidas encontradas na especificação, novamente auxiliando em uma implementação consistente.

TESTE AUTOMÁTICO DE CONFORMIDADE

Valendo-se das três técnicas descritas acima, é possível criar testes automáticos de conformidade, capazes de verificar o estado de conformidade de todos os Transmissores e alertá-los para novas inconsistências ou regressões antes delas chegarem ao ambiente produtivo, o que poderia causar disrupção do serviço.

Essa é uma forma bastante eficiente de se monitorar a conformidade das diversas implementações e incentivar os Transmissores a possuírem implementações o mais compatível possível com a especificação.

EVOLUTIBILIDADE

Para permitir a evolutibilidade do ecossistema de forma gradual, é fundamental se valer dos mecanismos de extensão do Http e evitar escolhas ou tecnologias que acoplem decisões diferentes do sistema. Uma delas seria, por exemplo, utilizar JWE, que efetivamente acopla um mecanismo de proteção de dados (criptografia) ao mecanismo de tipos de conteúdo (JSON).

CABEÇALHO ACCEPT

O Cabeçalho Accept serve para que quem está fazendo a requisição estabeleça quais os tipos de conteúdo (Content Type) que consegue processar no tratamento da resposta.

Isso permite que Transmissores suportem tipos de conteúdo mais compactos que JSON, sem implicar em possível dano ao ecossistema, dado que esse tipo de conteúdo só será utilizado se apresentado na requisição do Receptor.

Nossa recomendação é que todos os Transmissores sejam obrigados a suportar **JSON** (Content-Type: application/json) e na ausência desse cabeçalho na requisição deve ser assumido que a resposta deve ser em formato JSON.

CABEÇALHO ACCEPT-ENCODING

O Cabeçalho Accept-Encoding serve para definir qual algoritmo deve ser utilizado para comprimir os dados da resposta da requisição. Caso esse cabeçalho seja apresentado, os Transmissores podem compactar os dados da resposta, reduzindo o tráfego de rede.

Também é possível passar múltiplos valores nesse cabeçalho, o que permite que Transmissores se valham de algoritmos mais efetivos para aquele tipo de dado caso o Receptor sinalize dessa forma.

Nossa recomendação é que todos os Transmissores sejam obrigados a suportar **gzip** (Content-Encoding: gzip) e que todos os Receptores sejam obrigados a informar pelo menos suporte a **gzip** no Cabeçalho Accept-Encoding.

LEITURA TOLERANTE (TOLERANT READER)

O último mecanismo que deve ser implementado é a chamada Leitura Tolerante (Tolerant Reader). Diferentemente dos dois mecanismos antecedentes, este deve ser implementado de forma unilateral pelas instituições Receptoras.

Trata-se da determinação de que dados fora da especificação sejam ignorados pelos Receptores. Naturalmente, a definição de quais dados pessoais trafegam entre participantes deve ser sempre sujeita ao consentimento do cliente.

O que esse tipo de arranjo permite é que sejam criadas novas funcionalidades mais elaboradas pelos Transmissores, permitindo aplicações mais avançadas que talvez não sejam possíveis com a infraestrutura básica de Open Banking, além de permitir que a especificação evolua de forma gradual, sem necessitar mudança coordenada entre todos os participantes.

Conclusão

As propostas desenvolvidas neste documento representam um esforço de direcionar o open banking em uma direção segura, inclusiva, eficiente, democrática, evolutiva e centrada no consumidor final. Esperamos que esse trabalho instigue o debate de ideias, e ajude a identificar convergências de objetivos gerais dentre os participantes da discussão técnica.

Não por acaso, começamos o documento explicando nossa motivação, seguido pela demonstração das propriedades ou princípios básicos que queremos preservar no Open Banking, para apenas então entrar nas propostas específicas. Acreditamos que a cooperação dos participantes do ecossistema em torno desses objetivos comuns pode garantir a posição do Brasil na vanguarda dessa revolução do sistema financeiro. Os princípios e propostas técnicas aqui descritos são, na nossa visão, indispensáveis para que o Open Banking seja de fato democrático, aberto e evolutivo.

Nubank