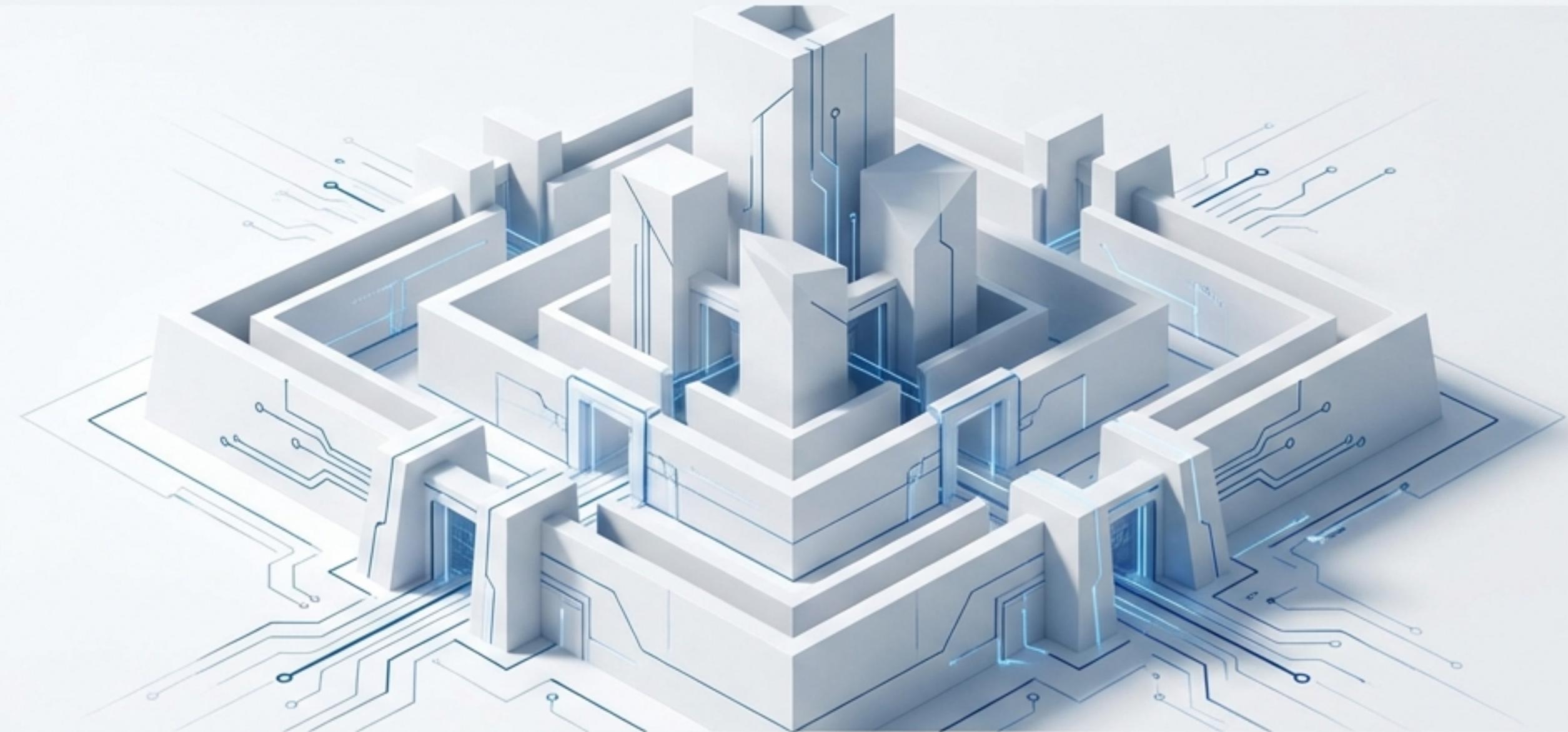


Construindo a Fortaleza: Segurança e Autenticação em Projetos CAP

Uma abordagem de defesa em profundidade para proteger suas aplicações.



A segurança em CAP não é um recurso único, mas uma filosofia de 'defesa em profundidade'. Nesta apresentação, exploraremos cada camada de proteção – das muralhas externas da plataformaia até as defesas internas do seu código – para construir aplicações robustas e seguras.



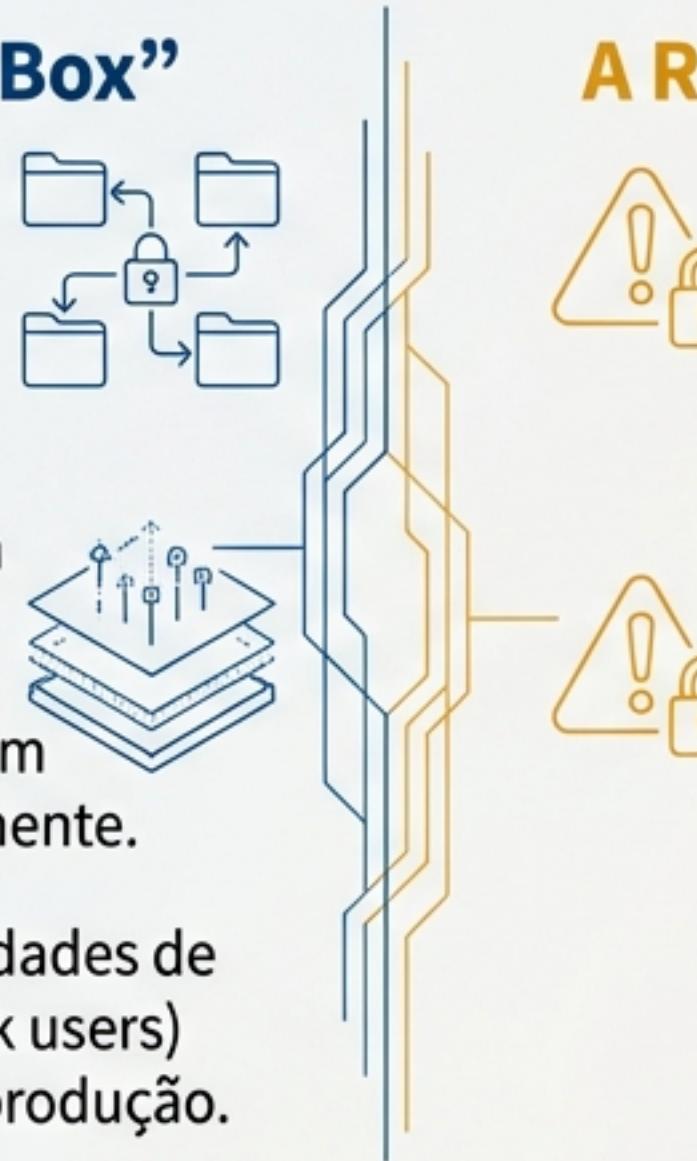
A Filosofia: “Secure by Default and by Design”

O CAP é projetado para ser seguro por padrão, minimizando a necessidade de configurações de segurança complexas e propensas a erros.



O que o CAP oferece “Out-of-the-Box”

-  **Comunicação Segura:** Não é necessário gerenciar senhas ou certificados para a comunicação entre serviços.
-  **Endpoints Autenticados:** Endpoints são autenticados por padrão ao vincular uma instância XSUAA.
-  **Isolamento de Tenants:** Multitenancy com isolamento de dados é fornecido nativamente.
-  **Perfis de Produção Seguros:** Funcionalidades de desenvolvimento (página de índice, mock users) são desativadas por padrão no perfil de produção.



A Responsabilidade do Desenvolvedor

-  **Configuração Explícita:** Aspectos como autorizações, dimensionamento da aplicação e políticas de segurança específicas precisam ser configurados ativamente.



-  **Testes de Segurança:** É recomendado garantir as configurações de segurança com testes de integração automatizados.



“O CAP lida com a complexidade, mas a estratégia de segurança final é sua responsabilidade.”



Camada 1: As Muralhas Externas - A Plataforma SAP BTP

Garantindo a integridade e confidencialidade da comunicação.

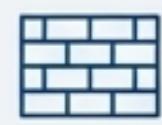
Ponto Chave: A comunicação entre cliente-servidor e serviço-serviço é protegida por padrão.



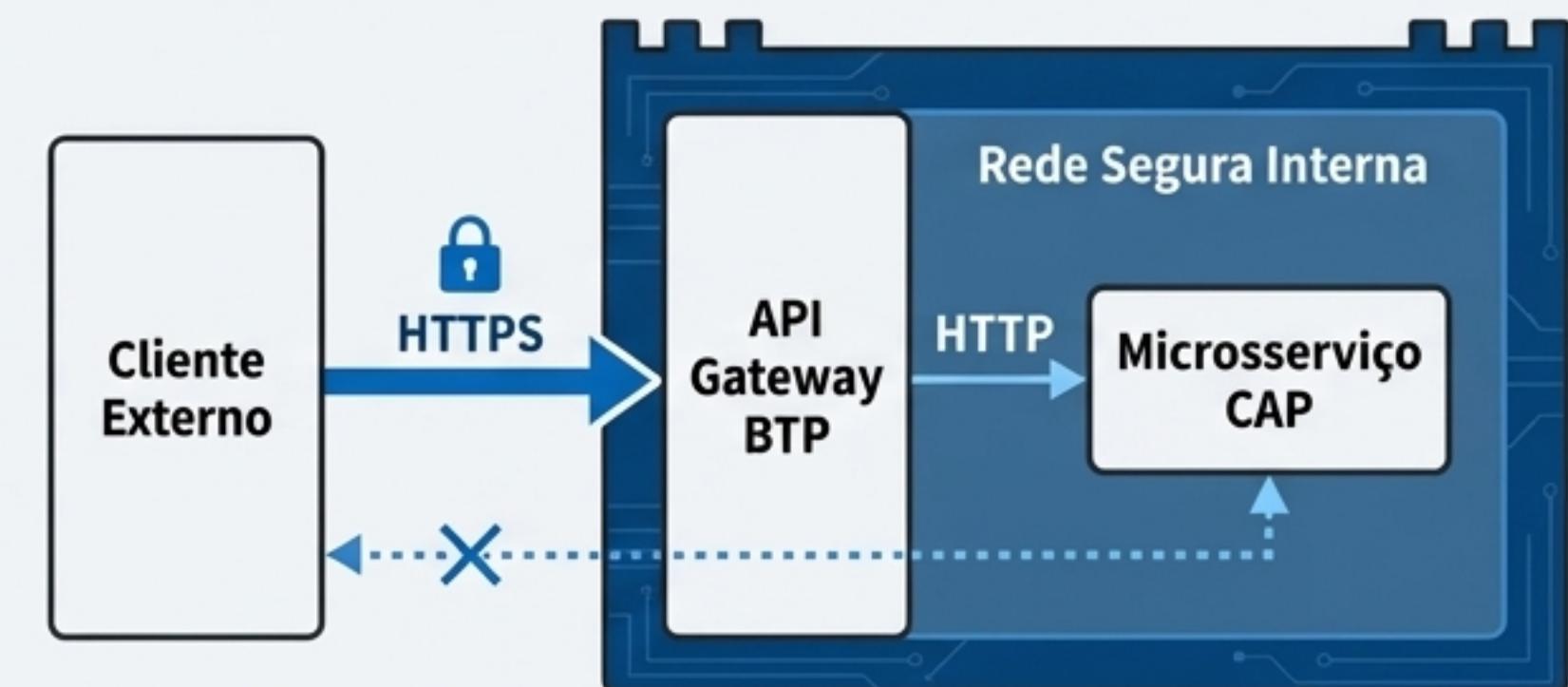
HTTPS/TLS por Padrão: A SAP BTP estabelece exclusivamente canais de comunicação criptografados baseados em HTTPS/TLS.



API Gateway/Ingress Router: Atua como o ponto de entrada, fornecendo endpoints TLS e encaminhando as requisições para os microsserviços via HTTP dentro de uma rede segura.



Segurança de Perímetro: Os endpoints HTTP dos microsserviços não são visíveis para clientes externos, sendo acessíveis apenas pelo router.



Sua aplicação já está dentro de um perímetro seguro. Nenhuma configuração de TLS é necessária no seu código CAP.

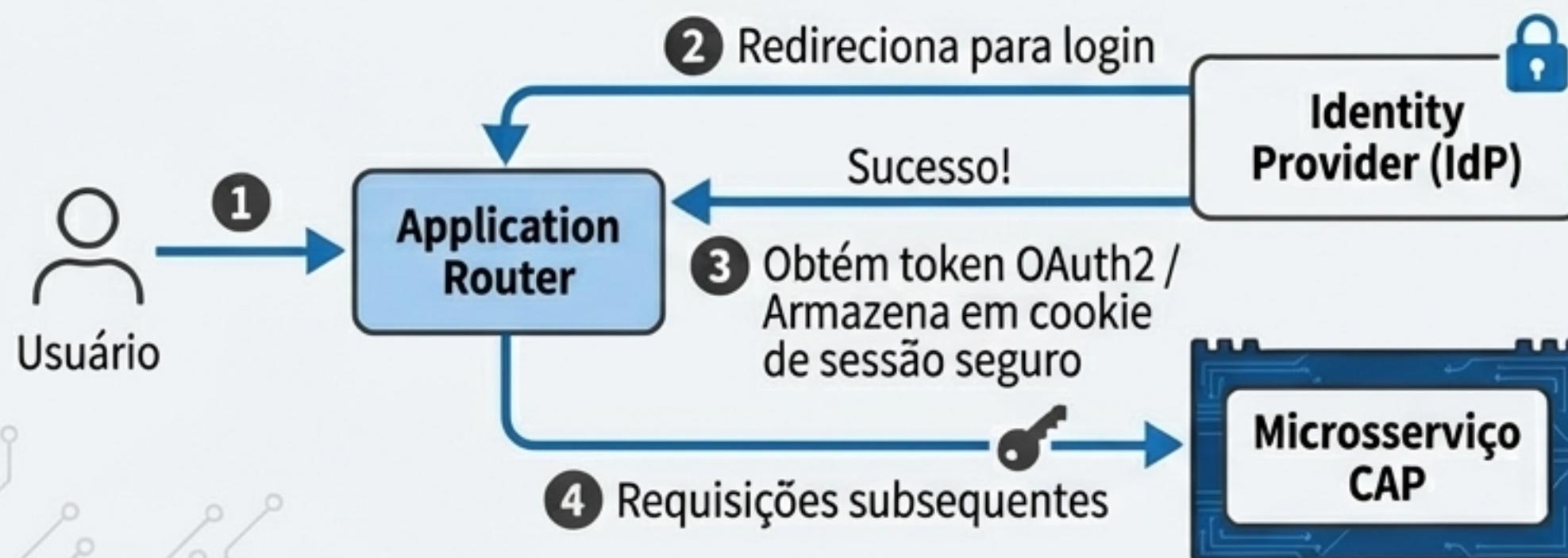


Camada 2: Os Portões - Autenticação de Usuários e Serviços

Garantindo a integridade e confidencialidade da comunicação.

O CAP não reinventa a roda. Ele se integra perfeitamente com os serviços de identidade da SAP BTP (Identity Authentication / XSUAA) para gerenciar o ciclo de vida da autenticação.

Fluxo de Autenticação (Usuário de UI)



“Com a autenticação configurada, **todos os endpoints CAP são autenticados por padrão.**”



O **Application Router** não esconde os endpoints do backend. A autenticação ainda é obrigatória para os microsserviços CAP.

Configuração Inicial

```
> cds add xsuaa
```

Isso gera a configuração necessária para criar uma instância de serviço XSUAA, com escopos e roles derivados diretamente do seu modelo de autorização CDS.

Camada 3: Os Guardas - Autorização com Anotações CDS

Conceito Central: Defina suas regras de acesso diretamente no seu modelo de dados e serviços com uma sintaxe simples e expressiva.

Principais Anotações

- 👤 `@requires: 'role'`: Exige que o usuário tenha um determinado *role* para acessar o serviço ou entidade.
- ☒ `@restrict` : Um conjunto de privilégios que define quem pode realizar operações específicas (READ, WRITE, CREATE, DELETE).

```
cds
service AdminService @(requires: 'Admin') {
    entity Incidents @(restrict: [
        { grant: 'READ', to: 'Viewer' },
        { grant: 'WRITE', to: 'Admin' }
    ]) { ... }
}
```

- Apenas usuários com o role `Admin` podem acessar o `AdminService`.
- Dentro do serviço, usuários com o role `Viewer` podem ler `Incidents`.
- Apenas usuários com o role `Admin` podem escrever em `Incidents`.



Aviso Crítico ! : Garanta que roles técnicos como `cds.Subscriber` ou `mtcallback` nunca sejam incluídos em roles de negócio atribuídos a usuários finais.

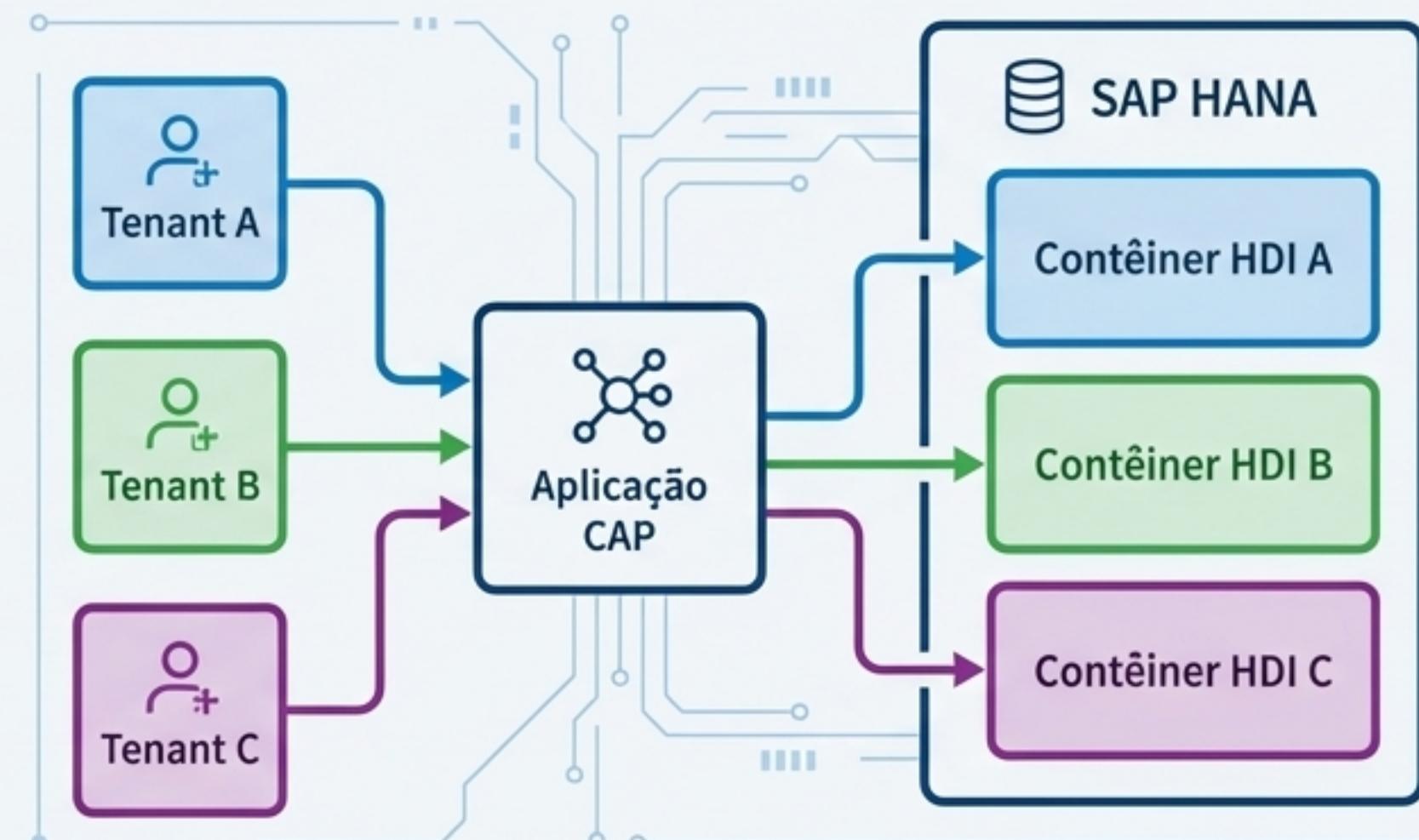
Camada 4

Camada 4: As Muralhas Internas - Isolamento de Tenants

O Desafio SaaS: Como garantir que os dados de um cliente (tenant) nunca sejam acessados por outro?

A Solução Automática do CAP:

- ✓ **Contêineres HDI Dedicados:** Ao servir uma requisição, o CAP mira automaticamente em um contêiner HDI isolado no SAP HANA Cloud, dedicado ao tenant da requisição.
- 📦 **Esquemas e Usuários de DB Separados:** Cada contêiner possui esquemas de banco de dados e usuários técnicos de DB separados, garantindo o isolamento no nível mais baixo.
- ↗️ **API de Query Agnóstica:** O código customizado (custom handlers) pode ser implementado de forma agnóstica à multitenancy. Você escreve a lógica de negócios, o CAP cuida do roteamento para o banco de dados correto.



Ponto de Atenção para Desenvolvedores: O isolamento de dados pode ser quebrado por código customizado mal escrito. Cuidado com variáveis de escopo global ou caches que não são sensíveis ao contexto da requisição (`cds.context`).

Ameaça Interna: Como Quebrar o Isolamento de Tenants

Um erro comum que vaza dados entre requisições e tenants.

```
// srv/cat-service.js
module.exports = srv => {
  let books // <- Vaza dados entre tenants e requisições concorrentes
  srv.on('READ', 'Books', async function(req, next) {
    if (books) return books
    return books = await next()
  })
}
```

Análise do Problema

- A variável `books` é declarada no escopo do módulo, fora do handler do evento `on`.
- A primeira requisição (de qualquer tenant) que chegar irá preencher esta variável.
- Todas as requisições subsequentes, *de qualquer tenant*, receberão os dados cacheados da primeira requisição, quebrando completamente o isolamento de dados.

A Solução

Nunca armazene dados específicos de uma requisição em variáveis de escopo mais amplo. Utilize sempre o escopo do handler ou o objeto de contexto (`req`). Propague dados através da pilha de chamadas via `cds.context`.

Aviso !: Dimensionamento e escalabilidade são de responsabilidade dos desenvolvedores.
Os valores padrão do CAP não são adequados para todas as aplicações.

Camada 5: As Defesas do Castelo - Proteção Contra Entradas Maliciosas

A Ameaça nº 1: Injeção de SQL



Proteção Intrínseca do CAP:

- O motor de queries do CAP (baseado em CQN/CQL) é imune a injeções de SQL via parâmetros de consulta.
- **Como?** As declarações CQL são transformadas em *prepared statements* que são executados com segurança em bancos de dados como o SAP HANA. Os valores de entrada do usuário são tratados como dados, não como código executável.

Onde a Vulnerabilidade Persiste:

- A injeção ainda é possível se a estrutura da query (entidade de destino, colunas, etc.) for construída dinamicamente com base na entrada do usuário.

Exemplo de Código Vulnerável (e como corrigir):

```
// ! CUIDADO: a estrutura da query depende da entrada do usuário
const entity = <from user input>
const column = <from user input>

// ✅ NECESSÁRIO: Validação explícita contra uma lista de permissão
validate(entity, column)

SELECT.from(entity).columns(column)
```



Diretriz para Desenvolvedores: Seja cuidadoso com código customizado que cria ou modifica queries CQL dinamicamente. Validação de entrada adicional é necessária.

Fortalecendo as Defesas: Validação, CSRF e CSP

1. Validação de Entrada



- **Ferramenta CAP:** Use a anotação `@assert` diretamente no seu modelo CDS para definir verificações de entrada específicas do campo.
- **Responsabilidade:** Aplicações devem validar ou higienizar todas as variáveis de entrada de acordo com o contexto de negócio.

2. Proteção Contra Cross-Site Request Forgery (CSRF)



- **Solução CAP:** O Application Router gerencia a sessão com o cliente e impõe a proteção de token CSRF (via header `x-csrf-token`) por padrão.
- **Responsabilidade:** Serviços CAP não precisam lidar com proteção CSRF, desde que não mantenham suas próprias sessões com o cliente.

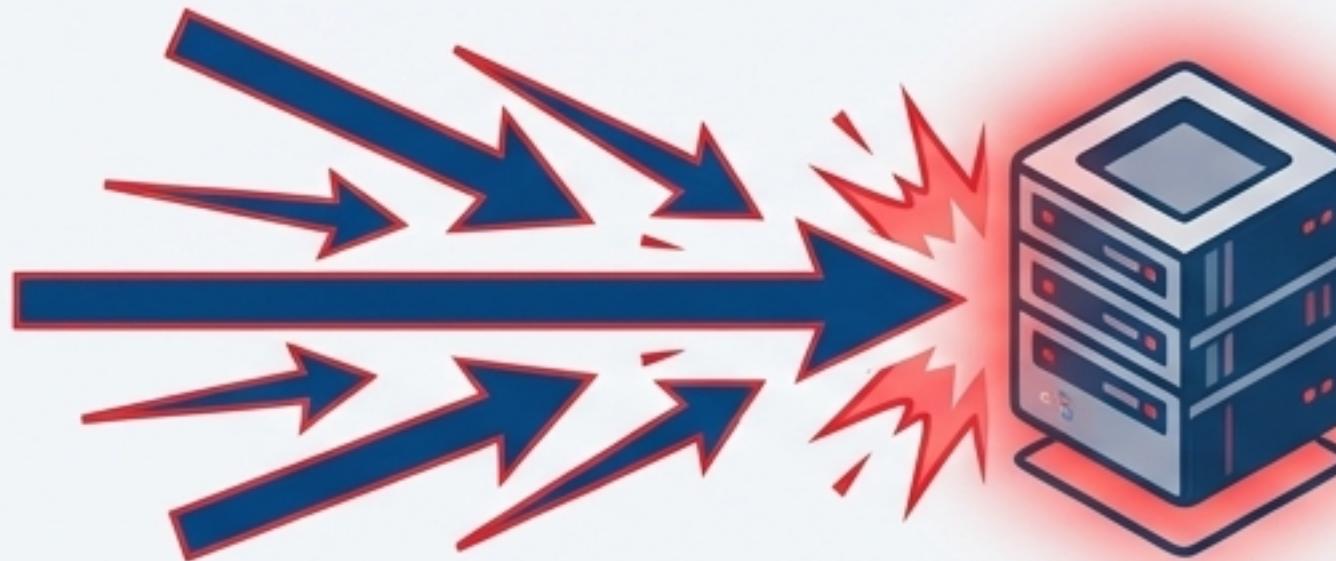
3. Content Security Policy (CSP)



- **O que é:** Uma camada extra de segurança que ajuda a detectar e mitigar ataques de injeção de código, restringindo os recursos que um navegador pode carregar.
- **Responsabilidade:** Aplicações **devem** configurar o header `Content-Security-Policy` no Application Router para atender aos requisitos básicos de conformidade.

Gerenciando Recursos Contra Ataques de Negação de Serviço (DoS)

O Ataque: Esgotar os recursos do servidor (CPU, memória, conexões) com requisições maliciosas para tornar o serviço indisponível.



Mitigações em Camadas do CAP

- **Servidor HTTP (Spring/Express)**
Limites padrão razoáveis para tamanho de header/body, fila de conexões, timeouts.
- **Adaptadores de Protocolo CAP**
 - Paginação automática de resultados de query para limitar picos de memória. Customizável com `@cds.query.limit`.
 - Limite do número total de requisições em batches OData.

⚠️ Configurações Críticas do Desenvolvedor ⚠️

- **Limitar `'\$expand'**
Aplicações CAP têm que limitar a quantidade de `$expands` por requisição em um handler customizado. Tenha cuidado especial ao expor associações.
- **Configurar Limite de Batch**
A quantidade máxima de requisições por `$batch` precisa ser configurada com `cds.odata.batch_limit = <max_requests>`.
- **Gestão de Carga do Banco de Dados**
Aplicações precisam estabelecer uma *Workload Management* adequada para controlar o uso de recursos do DB.
- **Rate Limiting**
Aplicações precisam estabelecer uma estratégia de *rate limiting* adequada.

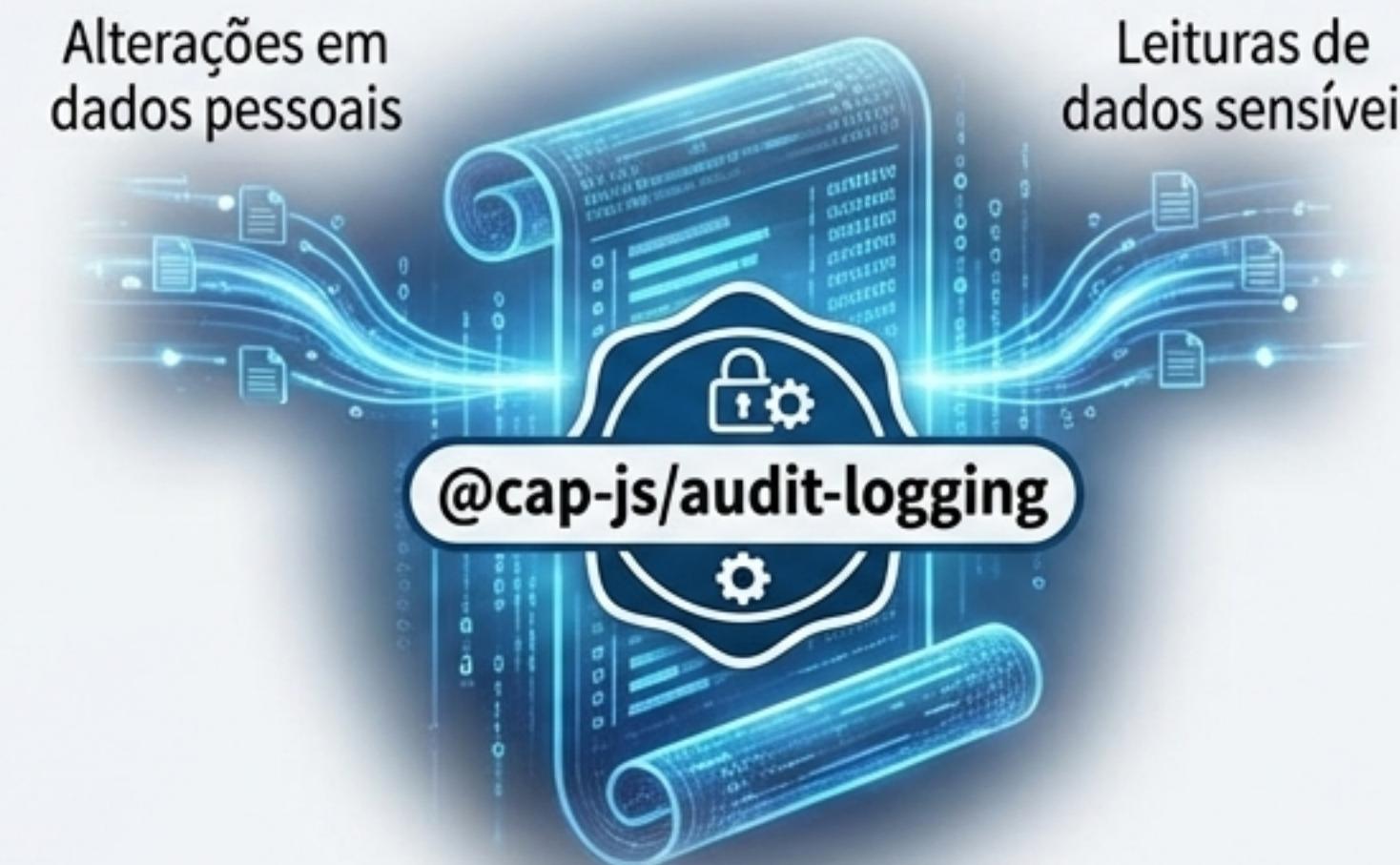
Camada 6: O Cronista do Reino - Log de Auditoria

Por que Log de Auditoria?

Para conformidade com regulamentações (ex: GDPR), rastreabilidade de acessos a dados sensíveis e investigação de incidentes de segurança.

A Solução CAP

O plugin `@cap-js/audit-logging`.



A Melhor Parte

A implementação é trivial e se integra perfeitamente ao ciclo de vida do CAP.

O que ele faz?

Fornece suporte “out-of-the-box” para o registro automático de eventos relacionados à privacidade de dados.

- ✓ Registra automaticamente:
 - ✓ Alterações em dados pessoais.
 - ✓ Leituras de dados sensíveis.

Log de Auditoria Automático em 2 Passos

Passo 1: Anotar Dados Pessoais

Identifique entidades e elementos que contêm dados pessoais usando as anotações `@PersonalData` e `@PersonalData.IsPotentiallySensitive`.

```
entity Customers {  
    key ID : UUID;  
    @PersonalData.FieldCategory: 'DataSubjectID'  
    ...  
    @PersonalData.IsPotentiallySensitive  
    creditCardNo : String;  
}
```

Resultado

O plugin intercepta automaticamente operações de escrita e leitura em dados anotados, constrói as mensagens de log e as envia para o serviço de log de auditoria.



Recurso de Resiliência

Todas as mensagens de log emitidas são enviadas através de um *transactional outbox*, garantindo a entrega mesmo que o serviço de log de auditoria esteja temporariamente indisponível.

Passo 2: Adicionar o Plugin

Execute um único comando no seu projeto:

```
> npm add @cap-js/audit-logging
```

```
{  
    "data_subject": {  
        "id": { "ID": "2b87f6ca-..." },  
        "role": "Customer",  
        "type": "AdminService.Customers"  
    },  
    "attributes": [  
        { "name": "firstName", "old": "Sunny", "new": "Jane" },  
        { "name": "lastName", "old": "Sunshine", "new": "Doe" }  
    ],  
    "user": "alice",  
    "time": "2024-02-08T09:21:45.021Z"  
}
```

Além da Automação: Log de Auditoria Customizado

Aplicações também podem registrar eventos customizados usando a API programática do serviço de auditoria.

Quatro Tipos de Eventos Pré-definidos



SensitiveDataRead

Acesso de leitura a dados pessoais sensíveis.



PersonalDataModified

Alterações em dados pessoais.



ConfigurationModified

Log de alteração de configuração.



SecurityEvent

Eventos de segurança genéricos (ex: tentativa de acesso não autorizado).

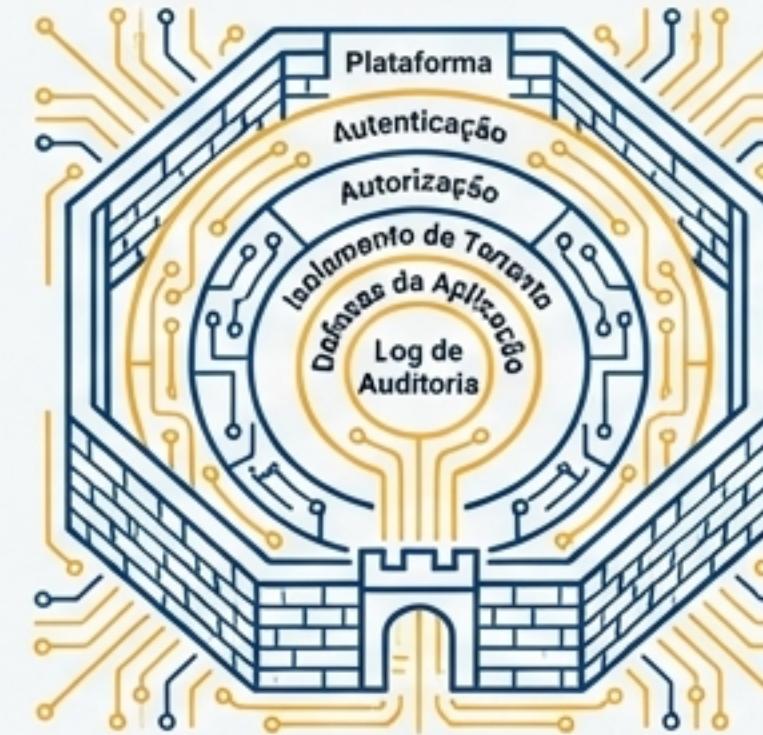
```
const audit = await cds.connect.to('audit-log');
await audit.log('SecurityEvent', {
  data: {
    user: 'alice',
    action: 'Attempt to access restricted service
\"PDMService\"'
  },
  ip: '127.0.0.1'
});
```



Ponto Chave: O serviço de log de auditoria é implementado como um serviço CAP padrão, o que significa que se beneficia de todos os padrões do framework, como mocking, resiliência e extensibilidade.



Resumo: Sua Fortaleza CAP



O que o CAP e a Plataforma Fornecem

Criptografia de rede (TLS).
Integração com Provedores de Identidade (XSUAA).
Proteção contra CSRF (via App Router).
Isolamento de dados para Multitenancy (HDI Containers).
Proteção contra injeção de SQL via CQN.
Paginação automática de resultados.
Log de auditoria automatizado com resiliência (outbox).

Sua Responsabilidade como Desenvolvedor

Configurar o método de autenticação (`cds add xsuaa`).
Definir o modelo de autorização (`@requires`, `@restrict`.
Evitar vazamento de dados em código customizado.
Validar entradas que modificam a *estrutura* das queries.
Configurar CSP, limites de `'\$expand'` e `'\$batch'`, e rate limiting.
Anotar dados pessoais (`@PersonalData`) para log de auditoria.
Implementar logs de auditoria personalizados para eventos de negócio.

A segurança é um esforço conjunto. O CAP fornece uma base sólida e segura, mas a construção de uma fortaleza impenetrável depende de código bem escrito e configurações diligentes.