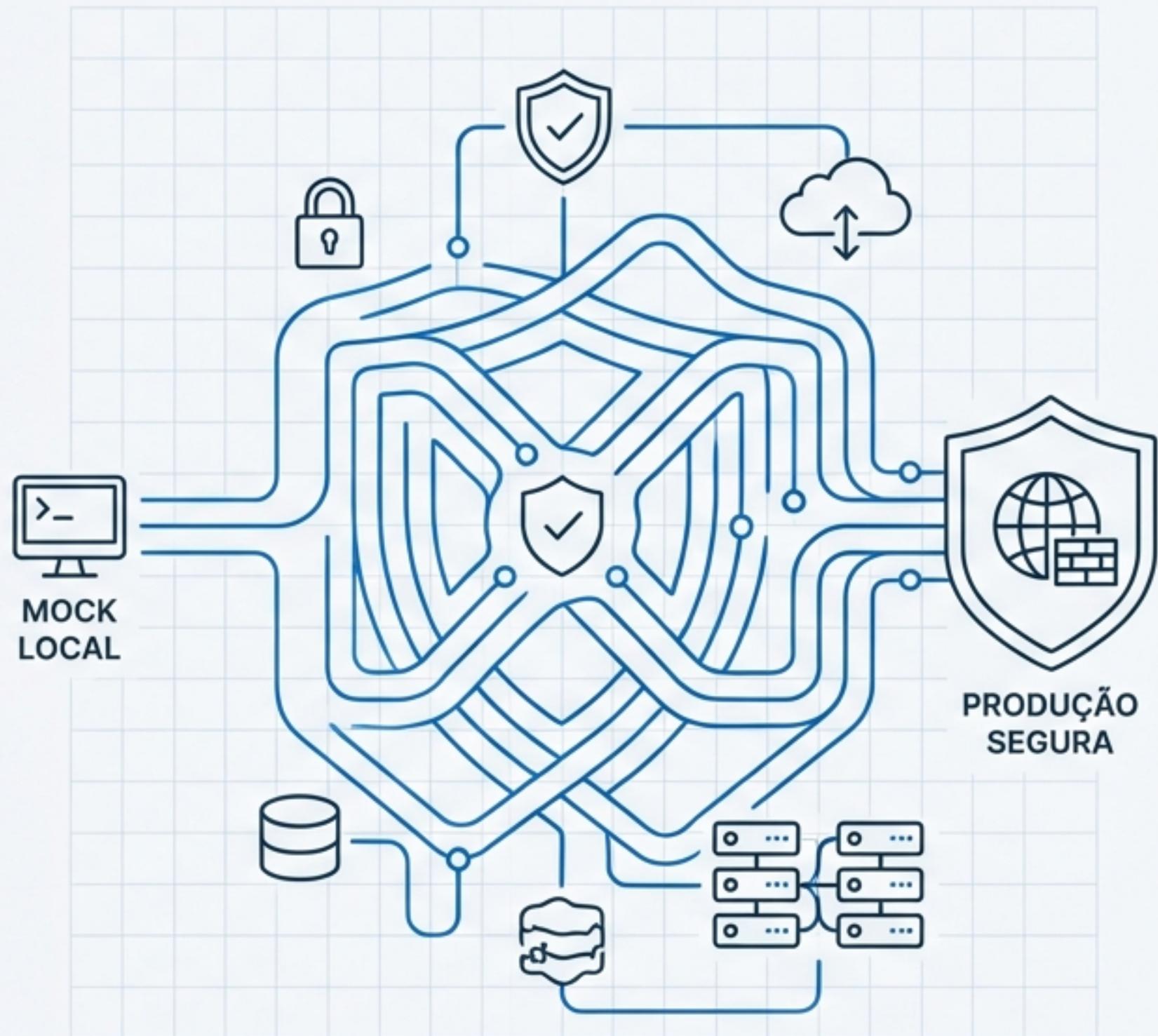


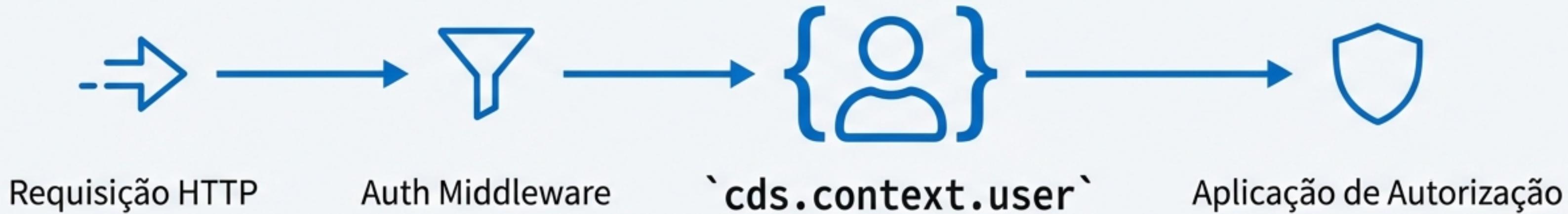
Dominando a Autenticação no SAP CAP com Node.js

Um Guia do Desenvolvedor: do
Mock Local à Produção Segura



O Coração da Autenticação: `cds.context.user`

Todo o mecanismo de autenticação no CAP Node.js gira em torno de um único objeto: `cds.context.user`. Middlewares de autenticação são responsáveis por preencher este objeto em cada requisição HTTP.



Este objeto de usuário é a fonte da verdade para todas as decisões de autorização em sua aplicação, sejam elas declarativas (em modelos CDS) ou programáticas (em código).

A Anatomia do Objeto `cds.User`

O objeto `user` não é apenas um ID. Ele carrega as roles, atributos e informações essenciais para a segurança.



.id

O ID único do usuário.
Corresponde a `'\$user` nas anotações `@restrict` do CDS.

```
const user = new  
cds.User('alice')
```



.is('<role>')

Verifica se o usuário possui uma determinada role.
Corresponde a `@requires` e `@restrict.grants.to`.

```
if (req.user.is('admin'))  
...
```



.attr

Atributos adicionais do usuário, geralmente extraídos de um JWT.
Corresponde a `'\$user.<x>'`.

where: `'\$user.level > 2'



.authInfo?

Um contêiner opcional para informações específicas da autenticação (por exemplo, a instância `SecurityContext` do `@sap/xssec`).



O conteúdo de `cds.User.authInfo` depende da biblioteca de autenticação utilizada. O CAP não garante seu conteúdo. Use com cautela e sempre fixe suas dependências.

A Sandbox do Desenvolvedor: Estratégias para Testes Locais

Durante o desenvolvimento, você precisa de flexibilidade para simular diferentes usuários e cenários de segurança sem a complexidade de um ambiente de produção. O CAP oferece estratégias “plug-and-play” para isso.

Estratégia	Propósito Principal	Configuração (package.json)
dummy	Desabilita temporariamente todas as checagens de autorização. O usuário passa em todos os testes (<code>.is()</code> retorna `true`).	<code>"auth": "dummy"</code>
mocked	Usa uma lista pré-definida ou customizada de usuários mock. Ideal para testar roles e atributos específicos.	<code>"auth": "mocked"</code>
basic	Semelhante ao ‘mocked’, mas sem usuários padrão. Exige que você defina todos os usuários mock.	<code>"auth": "basic"</code>

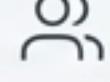


Dica Profissional: Para testar diferentes usuários no navegador, use sempre uma janela anônima para evitar que as informações de login sejam reutilizadas.

Foco em Desenvolvimento: Configurando Usuários com `mocked`

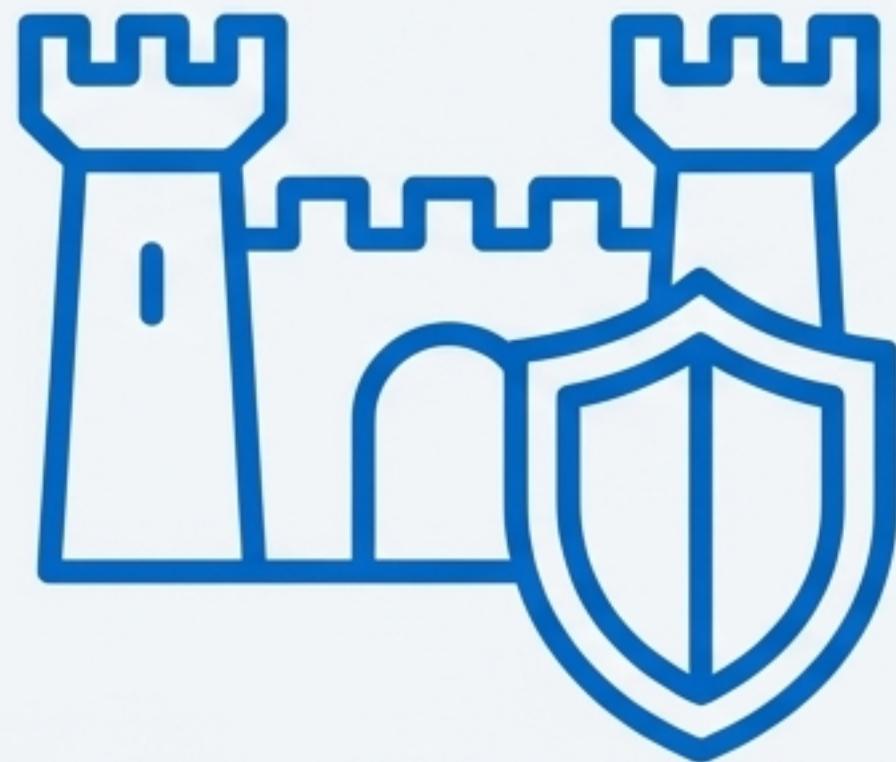
A estratégia `mocked` vem com um conjunto de usuários padrão (`alice`, `bob`, etc.) para um início rápido.

```
"cds": {  
  "requires": {  
    "auth": {  
      "kind": "mocked",  
      "users": {  
        "alice": {  
          "roles": ["admin"]  
        },  
        "bob": { "roles": [] },  
        "*": false  
      }  
    }  
  }  
}
```

-  **Definindo Usuários:** Você pode definir usuários personalizados com senhas, roles e atributos.
-  **Usuários Padrão:** Por padrão, a sua configuração é mesclada com a lista de usuários padrão do CAP.
-  **Travando o Acesso:** Para permitir o login apenas dos usuários que você especificou, adicione `*: false`. Isso é crucial para testes de segurança mais rigorosos.

Pronto para Produção: Estratégias Robustas e Seguras

Em produção, a segurança é a prioridade. O CAP inverte o padrão: todos os serviços são protegidos por padrão.



`@requires: 'authenticated-user'` é implicitamente adicionado a todos os serviços que não possuem anotações `@restrict` ou `@requires`.

As Estratégias de Produção

- `'jwt'` (Padrão): Utiliza JWT (JSON Web Tokens) de um serviço UAA (User Account and Authentication) vinculado.
- `'xsuaa'`: Uma configuração explícita para usar o serviço XSUAA do SAP BTP.
- `'ias'`: Utiliza o SAP Identity Authentication Service (IAS), também via JWT.

Pré-requisito Essencial

Para todas as estratégias baseadas em JWT, você precisa do pacote `@sap/xssec`.

```
npm add @sap/xssec
```

O Padrão de Produção: Autenticação com JWT e XSUAA

A identidade do usuário, suas roles e atributos são fornecidos em tempo de execução através de um token JWT no cabeçalho **Authorization** da requisição.



`jwt` - O Padrão Implícito

É a estratégia padrão em produção. O CAP automaticamente a utiliza quando detecta um ambiente produtivo.

```
"cds": {  
  "requires": { "auth": "jwt" }  
}
```

Configuração Explícita (package.json)



`xsuaa` - A Configuração Explícita

Funcionalmente similar ao `jwt`, mas declara explicitamente a intenção de usar o XSUAA.

```
"cds": {  
  "requires": { "auth": "xsuaa" }  
}
```

Configuração (package.json)



Ambas as estratégias preenchem o objeto **cds.context.user.authInfo** com dados do token, permitindo acesso a informações mais detalhadas de segurança.

Cenários Avançados: IAS e Autenticação Customizada



`ias` - Identity Authentication Service

Casos de Uso:

Integração com o IAS do SAP BTP.

Destaques:

- Validações adicionais de token (`x5t` thumbprint, proof-of-possession).
- Suporte a fallback para XSUAA para facilitar a migração.

```
"cds": {  
  "requires": {  
    "auth": "ias",  
    "xsuaa": true // Habilita o fallback  
  }  
}
```

Configuração (`package.json`)



`impl` - Sua Própria Lógica

Casos de Uso:

Integração com provedores de identidade não padrão, lógicas de autenticação legadas, etc.

O Contrato a Cumprir:

1. Atribuir uma instância de `cds.User` para `cds.context.user`.
2. Em ambientes multitenant, definir `cds.context.tenant`.

```
"cds": {  
  "requires": {  
    "auth": { "impl": "srv/custom-auth.js" }  
  }  
}
```

Configuração (`package.json`)

Otimizações e Detalhes de Produção



Segurança Reforçada por Padrão

Em produção, todos os serviços recebem implicitamente `@requires: 'authenticated-user'`.

Como Desabilitar (se necessário):

Use a feature flag. Este é um controle de segurança importante e deve ser usado com conhecimento.

```
"cds": {  
  "requires": {  
    "auth": {  
      "restrict_all_services": false  
    }  
  }  
}
```

Configuração (package.json)



Performance com Cache por Padrão

A validação de tokens JWT pode ser intensiva em CPU. `@sap/xssec` implementa cache para otimizar requisições subsequentes.

- **Cache de Assinatura:** Valida a assinatura criptográfica do token.
- **Cache de Decodificação de Token:** Armazena o conteúdo decodificado do token.

O cache de assinatura pode ser configurado via `cds.requires.auth.config`. O cache de decodificação é configurado programaticamente (ex: em `server.js`).

Guia Prático: Configurando o XSUAA em Ambiente Híbrido (Parte 1/4)

Objetivo: Testar localmente uma autenticação de nível de produção, conectando sua máquina ao serviço XSUAA no BTP.

Passo 1: Preparar o Ambiente Local

1. Login no Cloud Foundry:

```
cf l -a <api-endpoint>
```

2. Adicionar Configuração XSUAA ao Projeto:

```
cds add xsuaa --for hybrid
```

3. Configurar o `xs-security.json`:

Verifique se `xsappname` está definido e `tenant-mode` é `dedicated`.

4. Adicionar URI de Redirecionamento no `xs-security.json`:

```
"oauth2-configuration": {  
    "redirect-uris": [  
        "http://localhost:5000/login/callback"  
    ]  
}
```

5. Criar a Instância do Serviço XSUAA:

```
cf create-service xsuaa application bookshop-uaa -c xs-security.json
```

Guia Prático: Conectando sua App Local ao BTP (Parte 2/4)

Objetivo: Vincular sua aplicação local às credenciais do serviço XSUAA que você criou na nuvem.

Passo 2: Configurar a Aplicação

1. Criar uma Chave de Serviço:

```
cf create-service-key bookshop-uaa bookshop-uaa-key
```

2. Vincular (Bind) à Chave de Serviço:

```
cds bind -2 bookshop-uaa
```

3. O Que Acontece?:

Este comando cria ou atualiza o arquivo `.`.cdsrc-private.json` com as credenciais do serviço. Este arquivo **não** deve ser commitado no seu repositório.

```
{  
  "requires": {  
    "[hybrid)": {  
      "auth": { "kind": "xsuaa", "binding": { ... }  
    }  
  }  
}
```

NÃO COMMITE ESTE ARQUIVO

4. Verificar a Configuração:

```
cds env list requires.auth --resolve-bindings --profile hybrid
```

Guia Prático: Gerenciando Roles no BTP Cockpit (Parte 3/4)

Objetivo: Criar uma "Coleção de Roles" e atribuir as permissões (definidas no `xs-security.json`) a usuários reais.

Passo 3: Configurar as Roles para a Aplicação

1. Navegue até seu Subaccount no **SAP BTP Cockpit**.
2. Vá para **Security > Role Collections**.
3. Clique em “+” para criar uma nova coleção.
4. Dê um nome, por exemplo, ‘BookshopAdmin’, e salve.
5. Edite a nova coleção para adicionar:
 - **Roles:** Adicione as roles da sua aplicação (ex: `admin` do `bookshop!a<XXXX>`).
 - **Users:** Adicione os e-mails dos usuários que terão essas permissões.
6. **Salve** as alterações.

The screenshot shows the SAP BTP Cockpit interface. In the top left, there's a navigation bar with 'Configure', 'Userviews', 'Manager', 'Security' (with a dropdown menu), and 'Role Collections'. The 'Role Collections' item is highlighted. To its right, a table lists several role collections, each with a checkbox next to 'Name' and a 'Description' column. In the top right corner of the table area, there's a search bar and a blue circular button containing a white plus sign (+). The overall theme is light blue and white.

Guia Prático: Executando com o App Router (Parte 4/4)

Objetivo: Iniciar o App Router, que gerencia o fluxo de login interativo, e o serviço CAP, para testar a solução de ponta a ponta.

Passo 4: Executando a Aplicação Completa

1. Adicionar o App Router ao Projeto:

```
cds add approuter  
npm install --prefix app/router
```

2. Terminal 1

```
$ cds bind --exec -- npm start --prefix app/-  
router
```

- Terminal 2

```
$ cds watch --profile hybrid
```

- 
4. Testar: Abra `http://localhost:5000` no seu navegador. Você será redirecionado para a tela de login. Após autenticar, você poderá acessar as rotas protegidas de acordo com as roles atribuídas ao seu usuário.

Solução de Problemas: Corrigindo o Login no SAP BAS

O Problema

Ao rodar o setup híbrido no Business Application Studio, o login pode falhar com um erro de `redirect_uri` inválido. A mensagem de erro mostra a URL correta que é esperada.

Esta é uma configuração específica para o seu dev space e não deve ser enviada para o repositório de código ou compartilhada.

A Solução

1. **Copie a URL Completa** da mensagem de erro.
2. **Atualize o `xs-security.json`**, substituindo `http://localhost:5000/...` pela URL copiada.

```
"oauth2-configuration": {  
    "redirect-uris": [  
        "<url from error message>"  
    ]  
}
```

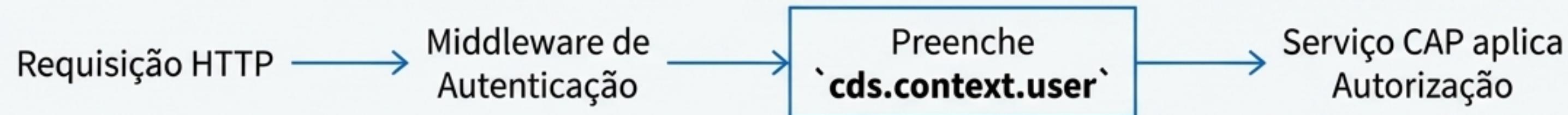
3. **Atualize a Instância do Serviço XSUAA** com a nova configuração:

```
cf update-service bookshop-uaa -c xs-security.json
```

4. **Tente Novamente.**

Sua Jornada de Autenticação no CAP: Do Mock à Produção

O Modelo Mental Central



O Caminho do Desenvolvedor



Inicie Rápido

Use **mocked** para simular usuários e roles durante o desenvolvimento ágil.



Teste com Realismo

Adote o **setup híbrido com xsuaa** para validar a segurança de produção em seu ambiente local.



Implante com Confiança

Use **jwt** ou **xsuua** em produção para uma autenticação robusta e integrada ao BTP.

A segurança no CAP é uma jornada clara e configurável, projetada para dar ao desenvolvedor o controle certo em cada fase do projeto, do primeiro teste ao *deploy* final.