Visão Geral

O que é a integração Senior X?

A plataforma Senior X permite que aplicações externas sejam embarcadas como **widgets** dentro do seu ecossistema. Sua aplicação será carregada em um **iframe** e receberá automaticamente os dados de autenticação do usuário logado na plataforma.

Benefícios

- **Single Sign-On (SSO)** : Usuários não precisam fazer login separado
- **Contexto corporativo**: Acesso a dados do usuário e tenant
- **APIs autenticadas**: Token Bearer para consumir APIs Senior X
- **Experiência integrada**: Aplicação aparece como parte nativa da plataforma

Pontos-chave

- **Comunicação**: Via `window.postMessage()` entre plataforma pai e iframe
- **Origem validada**: `https://platform.senior.com.br`

Autenticação via postMessage

Estrutura do Payload

Quando sua aplicação é carregada, a Senior X envia automaticamente uma mensagem com a seguinte estrutura:

```
interface SeniorXAuthPayload {
  token: {
    access_token: string;    // Bearer token para APIs
    username: string;        // Formato: "usuario@tenant"
    fullName: string;        // Nome completo (pode usar + como espaço)
    email: string;           // Email corporativo
    tenantName: string;      // Nome do tenant/empresa
    locale: string;          // Localização (ex: "pt-BR")
  };
  servicesUrl: string;       // URL base dos serviços
}
```

Formato do Username

O `username` vem no formato `usuario@tenant` . Para operações internas, você pode precisar normalizar:

```
function normalizeUsername(username: string): string {
  // Remove a parte @tenant se existir
  if (username && username.includes('@')) {
    return username.split('@')[0];
  }
  return username;
}

// Exemplo: "joao.silva@senior" → "joao.silva"
```

Formato do Nome Completo

O `fullName` pode vir com `+` no lugar de espaços:

```
function normalizeFullName(fullName: string): string {
  return fullName?.replace(/\+/g, ' ') || '';
}

// Exemplo: "João+da+Silva" → "João da Silva"
```

Implementação do Listener

Código Base

```
// Constantes
const SENIOR_X_ORIGIN = 'https://platform.senior.com.br';
const STORAGE_KEY_USER = 'senior_user';
const STORAGE_KEY_TOKEN = 'senior_token';

// Interface do usuário
interface SeniorUser {
  id: string;
  username: string;
  fullName: string;
  email: string;
  tenantDomain: string;
}

// Variáveis em memória (mais seguro que só localStorage)
let currentUser: SeniorUser | null = null;
let accessToken: string | null = null;

/**
 * Processa a mensagem de autenticação recebida da Senior X
 */
function handleAuthMessage(data: any): void {
  // Aceita string JSON ou objeto
  const payload = typeof data === 'string' ? JSON.parse(data) : data;

  // Extrai o token
  const token = payload.token || payload;

  if (token?.access_token) {
    // Armazena o token
    accessToken = token.access_token;
    localStorage.setItem(STORAGE_KEY_TOKEN, accessToken);

    // Monta o objeto do usuário
    const username = token.username || '';
    currentUser = {
      id: normalizeUsername(username),
      username: normalizeUsername(username),
      fullName: normalizeFullName(token.fullName || ''),
      email: token.email || '',
      tenantDomain: token.tenantName || ''
    };

    localStorage.setItem(STORAGE_KEY_USER, JSON.stringify(currentUser));

    console.log('Usuário autenticado:', currentUser.fullName);
  }
}

/**
 * Inicializa o listener de autenticação
 */
function initAuthListener(): void {
  // Carrega dados existentes do localStorage (se houver)
  const storedUser = localStorage.getItem(STORAGE_KEY_USER);
  const storedToken = localStorage.getItem(STORAGE_KEY_TOKEN);

  if (storedUser) currentUser = JSON.parse(storedUser);
  if (storedToken) accessToken = storedToken;
```

```
  // Listener para mensagens da plataforma pai
  window.addEventListener('message', (event) => {
    // CRÍTICO: Validar origem da mensagem
    if (event.origin !== SENIOR_X_ORIGIN) {
      return; // Ignora mensagens de outras origens
    }

    try {
      handleAuthMessage(event.data);
    } catch (error) {
      console.error('Erro ao processar mensagem de autenticação:', error);
    }
  });

  // Solicita dados de autenticação à plataforma pai
  if (window.parent && window.parent !== window) {
    window.parent.postMessage('requestInitialData', SENIOR_X_ORIGIN);
  }
}

// Funções auxiliares para acesso aos dados
function getCurrentUser(): SeniorUser | null {
  return currentUser;
}

function getAccessToken(): string | null {
  return accessToken;
}

function isAuthenticated(): boolean {
  return !!accessToken && !!currentUser;
}

function clearAuth(): void {
  currentUser = null;
  accessToken = null;
  localStorage.removeItem(STORAGE_KEY_USER);
  localStorage.removeItem(STORAGE_KEY_TOKEN);
}
```

Proteção de Rotas

Usar quando deseja que a aplicação funcione APENAS quando estiver dentro do iframe do Senior X

Verificações Necessárias

Sua aplicação deve verificar:

1. **Está rodando em iframe?** - `window.self !== window.top`
2. **Tem dados de autenticação?** - Token e usuário presentes

Exemplo de AuthGuard (React)

```
import { useEffect, useState, ReactNode } from 'react';
import { getCurrentUser, getAccessToken, initAuthListener } from './auth';

interface AuthGuardProps {
  children: ReactNode;
}

export function AuthGuard({ children }: AuthGuardProps) {
  const [isAuthenticated, setIsAuthenticated] = useState(false);
  const [isInIframe, setIsInIframe] = useState(true);
  const [isLoading, setIsLoading] = useState(true);

  // Bypass para desenvolvimento (REMOVER EM PRODUÇÃO!)
  const isDevelopment = import.meta.env.VITE_DISABLE_AUTH_GUARD === 'true';

  useEffect(() => {
    if (isDevelopment) {
      setIsAuthenticated(true);
      setIsLoading(false);
      return;
    }

    // Verifica se está em iframe
    const inIframe = window.self !== window.top;
    setIsInIframe(inIframe);

    // Inicializa listener de autenticação
    initAuthListener();

    // Verifica autenticação
    const checkAuth = () => {
      const user = getCurrentUser();
      const token = getAccessToken();
      setIsAuthenticated(!!(user && token));
      setIsLoading(false);
    };

    // Verifica imediatamente e após delays (para aguardar postMessage)
    checkAuth();
    const timers = [
      setTimeout(checkAuth, 500),
      setTimeout(checkAuth, 1500),
      setTimeout(checkAuth, 3000)
    ];

    return () => timers.forEach(clearTimeout);
  }, [isDevelopment]);

  if (isLoading) {
    return Carregando...;
  }

  if (!isDevelopment && (!isInIframe || !isAuthenticated)) {
    return (

        Acesso Restrito
        Esta aplicação deve ser acessada através da plataforma Senior X.

    );
  }

  return <>{children};
}
```

Consumo de APIs Senior X

Problema: CORS

Chamadas diretas do navegador para `platform.senior.com.br` são bloqueadas por CORS. A solução é usar um **proxy server-side**.

Solução: Backend Proxy (Edge Function / API Route)

sequenceDiagram participant Browser as Navegador participant Backend as Seu Backend participant SeniorAPI as API Senior X

```
 Browser->>Backend: Requisição com token
Backend->>SeniorAPI: Requisição server-side
SeniorAPI->>Backend: Resposta
Backend->>Browser: Resposta (sem CORS)
```

Exemplo: Edge Function (Deno/Supabase)

```
 // supabase/functions/senior-api-proxy/index.ts
import { serve } from "https://deno.land/std@0.168.0/http/server.ts";

const corsHeaders = {
  'Access-Control-Allow-Origin': '*',
  'Access-Control-Allow-Headers': 'authorization, content-type',
  'Access-Control-Allow-Methods': 'POST, OPTIONS',
};

serve(async (req) => {
  // Handle CORS preflight
  if (req.method === 'OPTIONS') {
    return new Response(null, { headers: corsHeaders });
  }

  try {
    const { endpoint, accessToken, body } = await req.json();

    // Faz a chamada para a API Senior X
    const response = await fetch(`https://platform.senior.com.br${endpoint}`, {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${accessToken}`,
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(body),
    });

    const data = await response.json();

    return new Response(JSON.stringify(data), {
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
    });
  } catch (error) {
    return new Response(JSON.stringify({ error: error.message }), {
      status: 500,
      headers: { ...corsHeaders, 'Content-Type': 'application/json' },
    });
  }
});
```

Exemplo: Chamada do Frontend

```
async function searchUsers(searchTerm: string): Promise {
  const accessToken = getAccessToken();

  if (!accessToken || searchTerm.length < 3) {
    return [];
  }

  const response = await fetch('/api/senior-proxy', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      endpoint: '/t/senior.com.br/bridge/1.0/rest/platform/user/queries/listUsers',
      accessToken,
      body: { searchValue: searchTerm }
    })
  });

  const data = await response.json();
  return data.users || [];
}
```

APIs Senior X Úteis

Endpoint	Descrição
`/t/{tenant}/bridge/1.0/rest/platform/user/queries/listUsers`	Busca usuários por termo
`/t/{tenant}/bridge/1.0/rest/platform/user/queries/getUser`	Detalhes de um usuário

Boas Práticas de Segurança

✅ Faça

- **Sempre valide a origem** das mensagens `postMessage`
- **Armazene tokens em memória** quando possível, localStorage como fallback
- **Use backend proxy** para APIs Senior X (evita CORS e exposição de tokens)
- **Normalize usernames** para comparações case-insensitive
- **Implemente timeout** na verificação de autenticação

❌ Não Faça

- Aceitar mensagens de qualquer origem
- Expor tokens em URLs ou logs
- Fazer chamadas diretas às APIs Senior X do navegador
- Deixar bypass de autenticação em produção
- Confiar apenas em verificações client-side