



# Frontend Implementation Guide - EmployeeVirtual

## Guia Prático para Desenvolvedores Frontend

**Sistema:** EmployeeVirtual API com JWT

**Versão:** 1.0

**Data:** Julho 2025

---



## Índice

1. [Setup Básico](#)
  2. [Autenticação](#)
  3. [Fazendo Requests](#)
  4. [Endpoints Principais](#)
  5. [Tratamento de Erros](#)
  6. [Exemplos Práticos](#)
  7. [Checklist Final](#)
- 

## 1. Setup Básico

### URL Base da API

```
javascript  
  
const API_BASE = 'http://localhost:8000/api';
```

### Configuração Padrão para Requests

```
javascript  
  
const defaultFetchConfig = {  
  credentials: 'include', // ← OBRIGATÓRIO: envia cookies automaticamente  
  headers: {  
    'Content-Type': 'application/json'  
  }  
};
```

---

## 2. Autenticação

## Login (Obrigatório antes de qualquer coisa)

javascript

```
async function login(email, password) {
  try {
    const response = await fetch(`${API_BASE}/auth/login`, {
      method: 'POST',
      credentials: 'include', // ← IMPORTANTE
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ email, password })
    });

    if (!response.ok) {
      throw new Error('Login falhou');
    }

    const data = await response.json();
    console.log('✅ Login realizado:', data.user.name);

    // Cookie é salvo automaticamente pelo navegador! 🍪
    return { success: true, user: data.user };

  } catch (error) {
    console.error('❌ Erro no login:', error);
    return { success: false, error: error.message };
  }
}
```

## Logout

javascript

```
async function logout() {
  try {
    await fetch(`${API_BASE}/auth/logout`, {
      method: 'POST',
      credentials: 'include'
    });

    // Redirecionar para login
    window.location.href = '/login';

  } catch (error) {
    console.error('Erro no logout:', error);
  }
}
```

## Verificar se está logado

javascript

```
async function checkAuth() {
  try {
    const response = await fetch(`${API_BASE}/auth/me`, {
      credentials: 'include'
    });

    if (response.ok) {
      const user = await response.json();
      return { authenticated: true, user };
    } else {
      return { authenticated: false };
    }

  } catch (error) {
    return { authenticated: false };
  }
}
```

## 3. Fazendo Requests

### ✓ Template Padrão (Copie e Cole)

javascript

```

async function apiRequest(endpoint, options = {}) {
  try {
    const response = await fetch(`${API_BASE}${endpoint}`, {
      credentials: 'include', // ← SEMPRE incluir
      headers: {
        'Content-Type': 'application/json',
        ...options.headers
      },
      ...options
    });

    // Tratar token expirado
    if (response.status === 401) {
      console.warn('🔒 Token expirado, redirecionando...');
      window.location.href = '/login';
      return null;
    }

    if (!response.ok) {
      const error = await response.json();
      throw new Error(error.detail || 'Erro na requisição');
    }

    return await response.json();
  } catch (error) {
    console.error('❌ Erro na API:', error);
    throw error;
  }
}

```

## Exemplos de Uso

javascript

```
// GET
const agents = await apiRequest('/agents/');

// POST
const newAgent = await apiRequest('/agents/', {
  method: 'POST',
  body: JSON.stringify({
    name: 'Meu Agente',
    description: 'Descrição do agente'
  })
});

// PUT
const updated = await apiRequest(`/agents/${agentId}`, {
  method: 'PUT',
  body: JSON.stringify({ name: 'Novo Nome' })
});

// DELETE
await apiRequest(`/agents/${agentId}`, { method: 'DELETE' });
```

## 4. Endpoints Principais

### Agentes

#### Listar todos os agentes do usuário

```
javascript

async function getMyAgents() {
  return await apiRequest('/agents/?include_system=true');
}

// Uso:
const agents = await getMyAgents();
console.log('Meus agentes:', agents);
```

#### Criar novo agente

```
javascript
```

```
async function createAgent(agentData) {
  return await apiRequest('/agents/', {
    method: 'POST',
    body: JSON.stringify(agentData)
  });
}

// Uso:
const newAgent = await createAgent({
  name: 'Assistente Vendas',
  description: 'Especialista em vendas B2B',
  prompt: 'Você é um especialista em vendas...',
  model: 'gpt-3.5-turbo'
});
```

## Buscar agente específico

```
javascript

async function getAgent(agentId) {
  return await apiRequest(`/agents/${agentId}`);
}

// Uso:
const agent = await getAgent(1);
```

## Atualizar agente

```
javascript

async function updateAgent(agentId, updateData) {
  return await apiRequest(`/agents/${agentId}`, {
    method: 'PUT',
    body: JSON.stringify(updateData)
  });
}

// Uso:
const updated = await updateAgent(1, {
  name: 'Nome Atualizado',
  description: 'Nova descrição'
});
```

## Deletar agente

javascript

```
async function deleteAgent(agentId) {  
  return await apiRequest(`/agents/${agentId}`, {  
    method: 'DELETE'  
  });  
}
```

// Uso:

```
await deleteAgent(1);  
console.log('Agente deletado!');
```

## Chat com Agente

### Chat rápido

javascript

```
async function chatWithAgent(agentId, message, context = {}) {  
  return await apiRequest(`/agents/chat`, {  
    method: 'POST',  
    body: JSON.stringify({  
      agent_id: agentId,  
      message: message,  
      context: context  
    })  
  });  
}
```

// Uso:

```
const response = await chatWithAgent(1, 'Como posso melhorar minhas vendas?');  
console.log('Resposta do agente:', response.agent_response);
```

### Executar agente (mais completo)

javascript

```
async function executeAgent(agentId, userMessage, context = {}) {
  return await apiRequest(`/agents/${agentId}/execute`, {
    method: 'POST',
    body: JSON.stringify({
      user_message: userMessage,
      context: context
    })
  });
}

// Uso:
const result = await executeAgent(1, 'Preciso de uma estratégia de vendas', {
  industry: 'tecnologia',
  company_size: 'startup'
});
```

## Histórico

### Buscar execuções anteriores

```
javascript

async function getAgentHistory(agentId, limit = 50) {
  return await apiRequest(`/agents/${agentId}/executions?limit=${limit}`);
}

// Uso:
const history = await getAgentHistory(1, 20);
console.log('Últimas 20 execuções:', history.executions);
```

## 5. Tratamento de Erros

### Códigos de Erro Comuns

```
javascript
```



```
async function handleApiError(response) {
  const error = await response.json();

  switch (response.status) {
    case 401:
      // Token expirado ou inválido
      console.log('🔒 Não autorizado - redirecionando para login');
      window.location.href = '/login';
      break;

    case 402:
      // Plano insuficiente
      showUpgradeModal(error.detail);
      break;

    case 403:
      // Sem permissão
      showError('Você não tem permissão para esta ação');
      break;

    case 404:
      // Não encontrado
      showError('Item não encontrado');
      break;

    case 429:
      // Rate limit
      const retryAfter = response.headers.get('Retry-After');
      showError(`Muitas tentativas. Aguarde ${retryAfter} segundos`);
      break;

    default:
      showError(error.detail || 'Erro desconhecido');
  }
}
```

## 💡 Função de Erro Universal

javascript

```
function showError(message) {  
  // Implementar conforme sua UI  
  console.error('❌ Erro:', message);  
  
  // Exemplo com toast/notification  
  // toast.error(message);  
  
  // Ou alert simples  
  alert('Erro: ${message}');  
}  
  
function showSuccess(message) {  
  console.log('✅ Sucesso:', message);  
  // toast.success(message);  
}  
  
function showUpgradeModal(message) {  
  // Implementar modal de upgrade  
  console.log('💎 Upgrade necessário:', message);  
  // modal.show('upgrade-plan');  
}
```

## 6. Exemplos Práticos

### Exemplo Completo: Dashboard de Agentes

html

```
<!DOCTYPE html>
<html>
<head>
  <title>Meus Agentes IA</title>
  <style>
    .agent-card { border: 1px solid #ddd; padding: 15px; margin: 10px; }
    .chat-box { border: 1px solid #ccc; height: 300px; overflow-y: auto; padding: 10px; }
    .message { margin: 5px 0; padding: 5px; }
    .user { background: #e3f2fd; text-align: right; }
    .agent { background: #f3e5f5; }
  </style>
</head>
<body>
  <div id="app">
    <header>
      <h1>Meus Agentes IA</h1>
      <span id="user-info"></span>
      <button onclick="logout()">Logout</button>
    </header>

    <main>
      <section>
        <h2>Agentes</h2>
        <button onclick="showCreateForm()">+ Criar Agente</button>
        <div id="agents-list"></div>
      </section>

      <section id="chat-section" style="display: none;">
        <h2>Chat</h2>
        <div id="chat-box" class="chat-box"></div>
        <input type="text" id="chat-input" placeholder="Digite sua mensagem...">
        <button onclick="sendMessage()">Enviar</button>
        <button onclick="closeChat()">Fechar Chat</button>
      </section>
    </main>
  </div>

  <script>
    // Configuração
    const API_BASE = 'http://localhost:8000/api';
    let currentAgentId = null;
    let userAgents = [];
```

```
// Inicializar app
async function initApp() {
  console.log('🚀 Inicializando app...');

  // Verificar se está logado
  const auth = await checkAuth();
  if (!auth.authenticated) {
    window.location.href = '/login.html';
    return;
  }

  // Mostrar info do usuário
  document.getElementById('user-info').textContent =
    `Olá, ${auth.user.name} (${auth.user.plan})`;

  // Carregar agentes
  await loadAgents();
}

// Carregar lista de agentes
async function loadAgents() {
  try {
    const agents = await apiRequest('/agents/');
    userAgents = agents;
    displayAgents();
    console.log('✅ ${agents.length} agentes carregados');
  } catch (error) {
    showError('Erro ao carregar agentes: ' + error.message);
  }
}

// Exibir agentes na tela
function displayAgents() {
  const agentsList = document.getElementById('agents-list');
  agentsList.innerHTML = userAgents.map(agent => `
    <div class="agent-card">
      <h3>${agent.name}</h3>
      <p>${agent.description}</p>
      <button onclick="openChat(${agent.id})">💬 Chat</button>
      <button onclick="editAgent(${agent.id})">✏️ Editar</button>
      <button onclick="deleteAgent(${agent.id})">🗑️ Deletar</button>
    </div>
  `).join("");
}
```

*// Abrir chat com agente*

```
function openChat(agentId) {  
  currentAgentId = agentId;  
  const agent = userAgents.find(a => a.id === agentId);  
  
  document.getElementById('chat-section').style.display = 'block';  
  document.getElementById('chat-box').innerHTML = "";  
  
  addChatMessage('system', `Chat iniciado com ${agent.name}`);  
}
```

*// Fechar chat*

```
function closeChat() {  
  document.getElementById('chat-section').style.display = 'none';  
  currentAgentId = null;  
}
```

*// Enviar mensagem*

```
async function sendMessage() {  
  const input = document.getElementById('chat-input');  
  const message = input.value.trim();  
  
  if (!message || !currentAgentId) return;  
  
  // Mostrar mensagem do usuário  
  addChatMessage('user', message);  
  input.value = "";  
  
  try {  
    // Enviar para o agente  
    const result = await chatWithAgent(currentAgentId, message);  
    addChatMessage('agent', result.agent_response);  
  
  } catch (error) {  
    addChatMessage('system', 'Erro: ' + error.message);  
  }  
}
```

*// Adicionar mensagem ao chat*

```
function addChatMessage(sender, message) {  
  const chatBox = document.getElementById('chat-box');  
  const messageDiv = document.createElement('div');  
  messageDiv.className = `message ${sender}`;
```

```

const time = new Date().toLocaleTimeString();
messageDiv.innerHTML = `
  <strong>${sender}</strong> ${message}
  <small style="opacity: 0.6;"> - ${time}</small>
`;

chatBox.appendChild(messageDiv);
chatBox.scrollTop = chatBox.scrollHeight;
}

// Deletar agente
async function deleteAgentConfirm(agentId) {
  if (!confirm('Tem certeza que deseja deletar este agente?')) return;

  try {
    await apiRequest(`/agents/${agentId}`, { method: 'DELETE' });
    await loadAgents(); // Recarregar lista
    showSuccess('Agente deletado com sucesso!');
  } catch (error) {
    showError('Erro ao deletar agente: ' + error.message);
  }
}

// Funções utilitárias (copiar das seções anteriores)
async function apiRequest(endpoint, options = {}) {
  // ... código da seção 3
}

async function checkAuth() {
  // ... código da seção 2
}

async function chatWithAgent(agentId, message) {
  // ... código da seção 4
}

function showError(message) {
  console.error('❌', message);
  alert('Erro: ${message}');
}

function showSuccess(message) {
  console.log('✅', message);
}

```

```
    alert(`Sucesso: ${message}`);
  }

  // Inicializar quando página carregar
  document.addEventListener('DOMContentLoaded', initApp);

  // Enter no chat
  document.addEventListener('keypress', function(e) {
    if (e.key === 'Enter' && e.target.id === 'chat-input') {
      sendMessage();
    }
  });
</script>
</body>
</html>
```

## Exemplo React/Vue/Angular

javascript

*// React Example*

```
import React, { useState, useEffect } from 'react';

function AgentDashboard() {
  const [agents, setAgents] = useState([]);
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    initializeApp();
  }, []);

  const initializeApp = async () => {
    // Verificar auth
    const auth = await checkAuth();
    if (!auth.authenticated) {
      window.location.href = '/login';
      return;
    }

    setUser(auth.user);

    // Carregar agentes
    try {
      const agentsList = await apiRequest('/agents/');
      setAgents(agentsList);
    } catch (error) {
      console.error('Erro ao carregar agentes:', error);
    }

    setLoading(false);
  };

  const handleChatWithAgent = async (agentId, message) => {
    try {
      const result = await chatWithAgent(agentId, message);
      return result.agent_response;
    } catch (error) {
      console.error('Erro no chat:', error);
      return null;
    }
  };
};
```



```
if (loading) return <div>Carregando...</div>;

return (
  <div>
    <h1>Bem-vindo, {user?.name}</h1>
    <div className="agents-grid">
      {agents.map(agent => (
        <AgentCard
          key={agent.id}
          agent={agent}
          onChat={handleChatWithAgent}
        />
      ))}
    </div>
  </div>
);
}
```

## 7. Checklist Final

### ✅ Antes de fazer deploy:

#### Configuração:

- ☐ URL da API configurada corretamente
- ☐ `credentials: 'include'` em todas as requests
- ☐ Tratamento de erro 401 (redirecionar login)
- ☐ Headers Content-Type configurados

#### Funcionalidades básicas:

- ☐ Login funciona e salva cookie
- ☐ Logout limpa cookie e redireciona
- ☐ Listar agentes funciona
- ☐ Criar agente funciona
- ☐ Chat com agente funciona
- ☐ Tratamento de erros implementado

#### UX/UI:

- ☐ Loading states durante requests
- ☐ Mensagens de sucesso/erro

- ☐ Confirmação antes de deletar
- ☐ Feedback visual para ações

### Testes:

- ☐ Testar login com credenciais corretas
- ☐ Testar login com credenciais incorretas
- ☐ Testar acesso sem estar logado
- ☐ Testar todas as operações CRUD
- ☐ Testar chat com diferentes agentes

---

## Resumo Executivo

### O que você precisa lembrar:

1. **SEMPRE** usar `credentials: 'include'` em todas as requests
2. **Login primeiro** antes de qualquer operação
3. **Tratar erro 401** redirecionando para login
4. **Não precisa gerenciar tokens** manualmente (cookies fazem isso)
5. **Usar a função** `apiRequest()` como base para todas as chamadas

### Template mínimo para qualquer request:

```
javascript

const response = await fetch(`${API_BASE}/endpoint`, {
  method: 'GET', // ou POST, PUT, DELETE
  credentials: 'include',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(data) // se POST/PUT
});
```

## Pronto para produção! 🚀

Sua API está 100% funcional e segura. O frontend é simples de implementar!

---

Versão: 1.0

Status: ☒ Pronto para desenvolvimento

Próxima revisão: Após feedback da equipe