# INTRODUÇÃO

## Contextualização

Com o crescimento e a popularização de APIs para o desenvolvimento de aplicações baseadas em Asynchronous Javascript and XML (AJAX), como jQuery, Prototype e ExtJS, é cada vez maior o número de aplicações que combinam dados e códigos provenientes de diversas fontes espalhadas pela internet, proporcionando a criação de inúmeros novos serviços e funcionalidades.

Quando este tipo de aplicação é desenvolvido, encontram-se diversas dificuldades inerentes a realização de comunicações AJAX com serviços de outras origens. Nesta contexto, uma origem é definida pela combinação de um protocolo, um domínio e uma porta [FLANAGAN, 2011].

O principal obstáculo na execução de uma chamada AJAX Cross-Domain é o conceito de segurança Same-Origin Policy (SOP), introduzido pelo navegador Netscape 2.0 em Março de 1996 [PRATES, 2010]. Este conceito de segurança é aplicado a linguagens de programação que são executadas no navegador, como JavaScript, e limita o acesso à maioria das propriedades entre páginas de domínios diferentes, como funções JavaScript, cookies e dados de formulários. Tais restrições foram criadas para mitigar ataques de Cross-site scripting (XSS) e Cross-site request forgery (CSRF/XSRF) [FLANAGAN, 2011].

A política de mesma origem, devido a sua característica de forte restrição, maximizou consideravelmente a segurança dos navegadores e é adotada em todos os navegadores web modernos. Porém, a comunicação AJAX entre origens distintas não é por natureza insegura ou mal, é de fato essencial para muitas das aplicações mais populares e utilizadas do mundo [JSON-P, 2008].

Diante da emergente necessidade da troca de informações entre aplicações de diferentes origens via AJAX, foram desenvolvidos diversos mecanismos para contornar as limitações impostas pelo SOP. Alguns dos mais conhecidos são:

1. Equiparação de domínio: este método iguala os domínios das aplicações através da propriedade “document.domain”, permitindo assim a execução de requisições AJAX. Esta técnica, porém, só funciona para cenários de cross-subdomain.
2. Cross-Origin Messaging: especificada pelo HTML 5, essa é uma nova forma de comunicação assíncrona entre janelas do navegador, inclusive para documentos carregados em iframes. Sua utilização se dá pela invocação da função “window.postMessage”. Pela sua finalidade, entretanto, esta técnica não prevê suporte para comunicação AJAX.
3. Server-side proxy: consiste em efetuar as chamadas AJAX para uma URL da sua própria aplicação e, no lado servidor, fazer a chamada para o serviço cross-domain. Este mecanismo atende ao cenário de cross-domain para requisições que utilizem os mais variados verbos HTTP, porém sua implementação, além de intrusiva, demanda muito esforço de configuração e aumenta a complexidade de implantação da aplicação.
4. JavaScript Object Notation with Padding (JSONP): esta técnica utiliza a simples técnica do servidor retornar um código Javascript que passa um JSON para uma função de callback provida pelo lado cliente via parâmetro da requisição. Este método funciona para cenários cross-domain, porém apenas com requisições que utilizam GET como verbo HTTP, além de não ser aderente ao modelo de comunicação AJAX.

## Problema

Atualmente para implementarmos um diálogo cross-domain via AJAX temos poucas alternativas técnicas disponíveis, sendo que estas possuem muitas limitações de segurança e contra-indicações. A falta de um modelo robusto e padronizado de comunicação entre origens dissemelhantes gera uma lacuna que muitas vezes é preenchida por soluções inseguras e muito complexas, que acabam, respectivamente, expondo dados sensíveis e degradando a manutenabilidade da aplicação.

## Problematização

Com o contexto mencionado, questiona-se:

1. Como abstrair a complexidade no desenvolvimento de sistemas que necessitem consumir, através de requisições AJAX, serviços expostos em diversos domínios?
2. Como expor serviços na internet que possam ser consumidos apenas por origens autorizadas?
3. Como fazer o controle do nível de acesso de acordo com o recurso solicitado e origem da requisição?
4. Como estender a API de chamadas AJAX, disponibilizada pelos navegadores web, para saber trabalhar requisições cross-domain mantendo sua interface original?
5. Como tornar possível o uso dos mais variados verbos HTTP, como: GET, POST, PUT, DELETE e OPTIONS, na realização de requisições AJAX cross-domain?

## Objetivos

### Objetivos gerais

O objetivo deste projeto é apresentar a especificação Cross-Origin Resource Sharing (CORS), desenvolvida pelo pelo consórcio W3C para o compartilhamento de recursos consumíveis por aplicações de outros domínios através de chamadas AJAX, estendendo o conceito de segurança Same-Origin Policy.

### Objetivos específicos

Os objetivos específicos deste projeto, que visa abordar a especificação CORS, são:

1. Detalhar o funcionamento teórico da especificação CORS, apresentando toda a mecânica envolvida no diálogo entre as aplicações envolvidas em uma comunicação cross-domain;
2. Desenvolver um estudo de caso para demonstrar a aplicabilidade prática do CORS. Este estudo de caso deverá abranger o desenvolvimento passo a passo de duas aplicações:
   * Aplicação api-provider, cuja finalidade será disponibilizar serviços RESTful consumíveis por diversas origens;
   * Aplicação api-consumer, cujo objetivo será fazer o consumo, via AJAX, dos serviços disponibilizados pela aplicação api-provider.
3. Informar a atual aderência dos principais navegadores web para com a especificação CORS, indicando quais características já se tornaram realidade e quais ainda são tendência.

## Relevância

Este projeto visa simplificar e melhorar a qualidade do desenvolvimento de aplicações que efetuem interações com aplicações externas as suas fronteiras de domínio utilizando o conceito de requisições assíncronas AJAX.

A simplificação deste desenvolvimento é dada através da minimização do esforço de configurações externas ao escopo da aplicação, como intervenções em Web Servers. A remoção desta dependência arquitetural é de grande valia em ambientes corporativos pois, como consequência, diminuímos os esforços delegados a área de infra-estrutura.

A forma semi transparente com que a solução apresentada é aplicada também torna os elementos participantes da comunicação cross-domain mais coesos, ou seja, eles não precisam saber nada além do que solicitar e do que responder. Toda negociação de acesso aos recursos fica apartada e abstraída, tornando o código do cliente e do servidor mais limpo. Isso maximiza a legibilidade do código e o seu nível de manutenabilidade.

Como resultado das vantagens apresentadas previamente, este projeto tem como finalidade contribuir com a disseminação de conhecimento da especificação CORS, proporcionando melhorias de qualidade do código, aumento na segurança das informações trafegadas e redução de complexidade e de recursos necessários para o desenvolvimento, como tempo e pessoas.