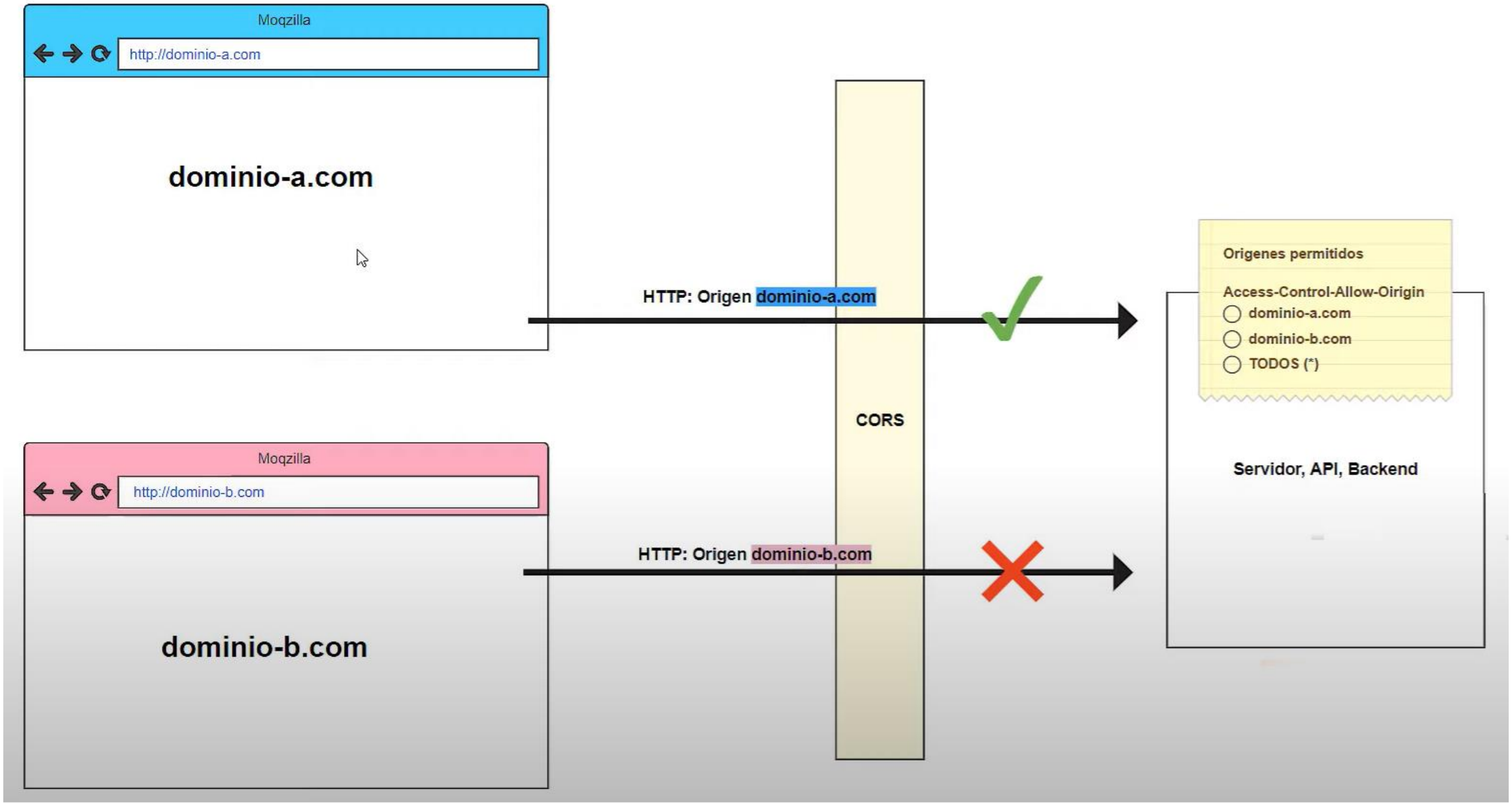
Configurando CROS en una API REST

Intercambio de recursos de origen cruzado (CORS)

El intercambio de recursos de origen cruzado (CORS, por sus siglas en inglés), es un mecanismo basado en cabeceras HTTP que permite a un servidor indicar cualquier dominio, esquema o puerto con un origen (en-US) distinto del suyo desde el que un navegador debería permitir la carga de recursos. CORS también se basa en un mecanismo por el cual los navegadores realizan una solicitud de "verificación previa" al servidor que aloja el recurso de origen cruzado, con el fin de comprobar que el servidor permitirá la solicitud real. En esa comprobación previa, el navegador envía cabeceras que indican el método HTTP y las cabeceras que se utilizarán en la solicitud real.


Moqzilla
http://dominio-a.com
dominio-a.com
HTTP: Origen dominio-a.com
CORS
Origenes permitidos
Access-Control-Allow-Oirigin
dominio-a.com
dominio-b.com
TODOS (*)
Servidor, API, Backend
Moqzilla
http://dominio-b.com
HTTP: Origen dominio-b.com
dominio-b.com

Por seguridad...

- Por razones de seguridad, los navegadores prohíben las llamadas AJAX a recursos que residen fuera del origen actual.
- Por ejemplo, mientras revisas tu cuenta bancaria en una pestaña, podrías tener el sitio web *evil.com* en otra pestaña. Los scripts de *evil.com* no deberían poder realizar solicitudes AJAX a la API de tu banco (¡retirar dinero de tu cuenta!) utilizando sus credenciales.
- CORS es una especificación del W3C implementada por casi todos los navegadores que permite especificar qué dominios estarán autorizados para qué.

¿Qué es CORS?

- Es una política de seguridad.
- **¿Y qué es una política de seguridad?**
  - Es lo que significa *ser seguro* para una entidad.
  - Puede implicar una serie de reglas de control de acceso, autenticación, ...

Entonces... ¿qué es CORS?

- Cross-Origin Resource Sharing (CORS)
- Mecanismo que utiliza cabeceras HTTP adicionales para permitir a un cliente (*User-Agent*) acceder a recursos desde un (servidor) origen diferente al sitio (servidor) actual.

Ventajas del uso de CORS

- Permite indicar quién puede acceder a nuestros recursos
- También permite indicar cómo se puede acceder (métodos HTTP)
  - Puedes habilitar GET…
  - Y deshabilitar PUT, DELETE, …

Configuración a nivel de método

- Spring provee la anotación *@CrossOrigin*
- Permite indicar, entre otras propiedades, los orígenes permitidos para un método o un controlador completo

```
@CrossOrigin(origins = "http://localhost:9001")
@GetMapping("/producto")
public ResponseEntity<?> obtenerTodos() {
    //...
}
```

Configuración Global

```
@Bean
public WebMvcConfigurer corsConfigurer() {
    return new WebMvcConfigurer() {
        @Override
        public void addCorsMappings(CorsRegistry registry) {
            registry.addMapping("/**")
                .allowedOrigins("http://localhost:4200")
                .allowedMethods("*")
                .allowedHeaders("*");
        }
    };
}
```

Configuración global

```
@Bean
public WebMvcConfigurer corsConfigurer() {
    return new WebMvcConfigurer() {
        @Override
        public void addCorsMappings(CorsRegistry registry) {
            registry.addMapping("/producto/**")
                .allowedOrigins("http://localhost:9001")
                .allowedMethods("GET", "POST", "PUT", "DELETE")
                .maxAge(3600);
        }

    };
}
```

```
$.ajax({
    url:"http://localhost:8080/productos",
    type: "GET",
    success:function(data){
        let html = "";
        $.each(data,function(index,rec){
            html += "<tr>";
            html += "<td>" + rec.id + "</td>";
            html += "<td>" + rec.nombre + "</td>";
            html += "<td>" + rec.precio + "</td>";
            html += "<td>" + rec.stock + "</td>";
            html += "</tr>";
        });
        $("#productos").append(html);
    },
    error:function(error){
        console.log(error);
        //alert(error.statusText);
    }
})
```