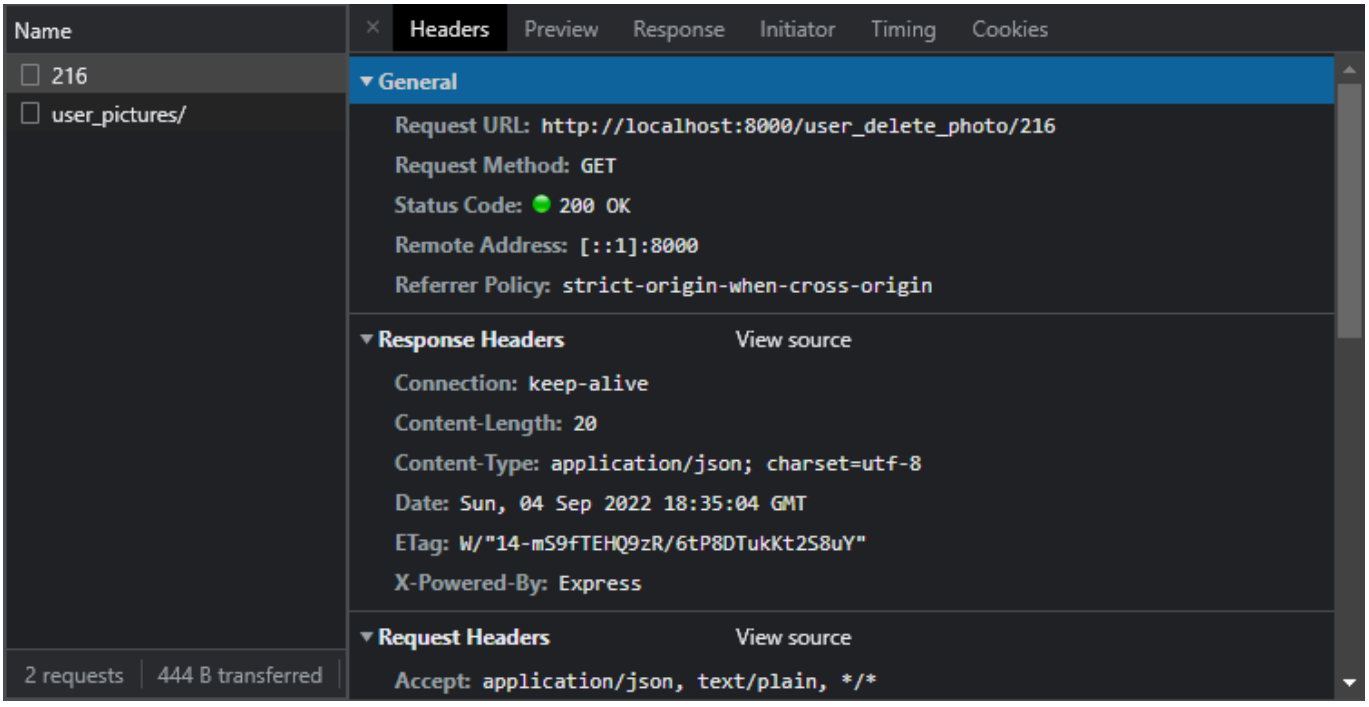
Práctica de CSRF

1. *Clonar el repositorio https://github.com/DevSlop/Pixi y posicionarse en el directorio.*
2. *Ejecutar docker-compose up*
3. *Ingresar a http://localhost:8000.*
4. *Iniciar sesión con un usuario creado por ustedes y buscar la funcionalidad de Share pictures.*
5. *Subir una imagen cualquiera.*
6. *Buscar la funcionalidad de borrado y observar cómo funciona y cómo podrían ejecutar un ataque de CSRF entre ustedes (evaluar si se dan las condiciones para poder ejecutar este ataque, de acuerdo a lo visto en el teórico). Pueden trabajar en grupo e intentar hacer el ataque entre ustedes mismos. A los fines de exponer su localhost de manera pública, pueden hacerlo mediante la herramienta ngrok (https://ngrok.com/). En caso de trabajar solos, pueden probar con distintos usuarios en la aplicación.*
7. *Determinar dónde implementarían el fix en el código.*

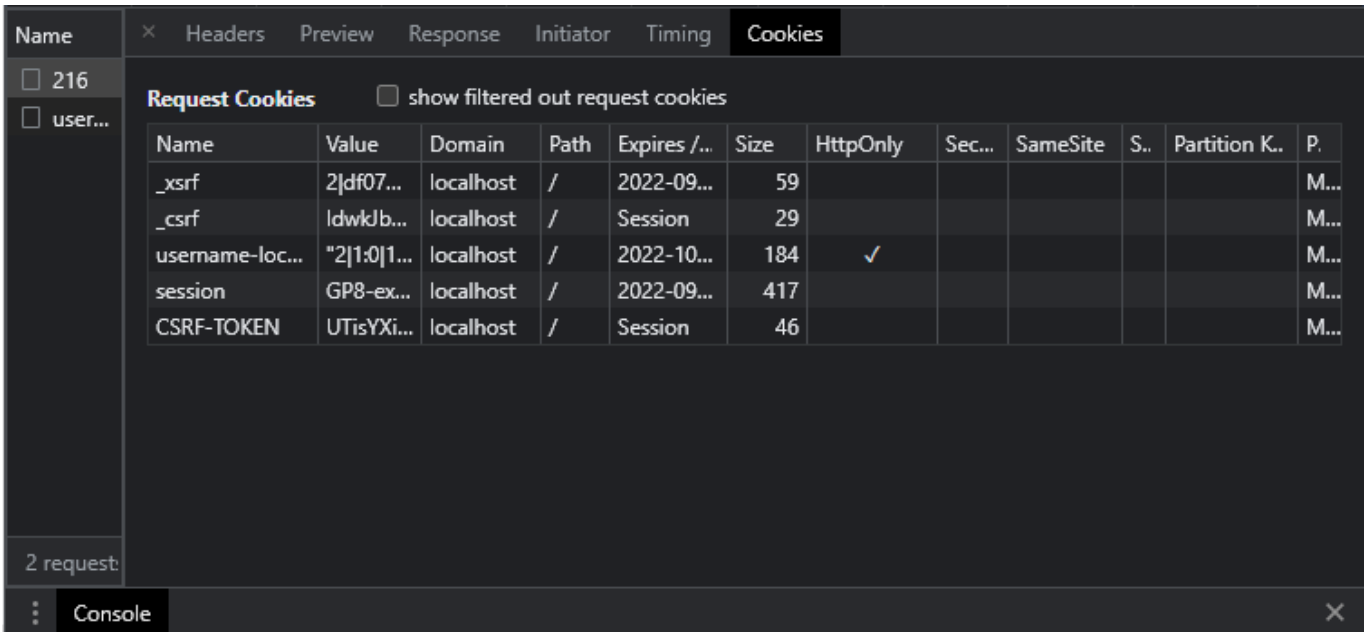
Material de ayuda: https://portswigger.net/web-security/csrf

6. **Condiciones**
   1. el usuario puede realizar acciones como subir o borrar imágenes
   2. sesión mediante cookies
   3. el borrado no recibe parámetros impredecibles, solo el id de la imagen

Petición GET que invoca el botón de borrar imagen



Cookies presentes en la petición, vemos presente el uso del token CSRF pero no el uso de las cookies SameSite



Código **html** del botón de borrado

```
<a href="#" ng-click="user_delete_photo(picture._id)" class="btn btn-info"> == $0
    "Delete Photo "
```

Código **javascript** del botón de borrado

```
$scope.user_delete_photo = function(photo){
    $http.get('/user_delete_photo/' + photo)
        .then(function success(response){

            console.log('in delete');
            $scope.user_pictures();
            $scope.picture_alerts = [
                { type: 'success',  msg: "Buh-Bye, Photo Deleted! " },
            ];

            $scope.closeAlert = function(index) {
            $scope.alerts.splice(index, 1);
        }
        })
}
```

Código **javascript** del backend

```
app.delete('/user_delete_photo/:picture_id', login_check, function(req, res) {
    console.log('in delete' + req.params.picture_id);
    if(!req.params.picture_id) {
        res.json('NO PICTURE SPECIFIED TO DELETE');
    }
    else {
        mongo.connect(dbname, function(err, db){
            db.collection('pictures').remove( { _id : Number(req.params.picture_id) },
                function(err, delete_photo){
                    if (err) { return err }
                    if (!delete_photo) {
                        res.json('Photo not found');
                        }
                    else {
                        res.json('Photo ' +  req.params.picture_id + ' deleted!');
                        console.log(delete_photo);
                        console.log(err);
                        }
                })
        })
    }
});
```

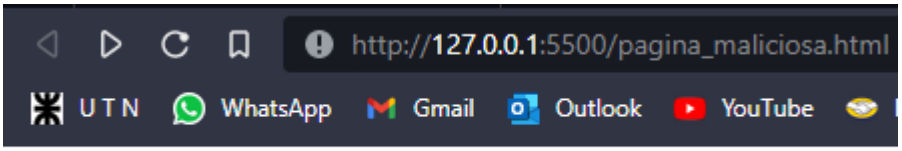
```
app.get('/user_delete_photo/:picture_id', login_check, function(req, res) {
    console.log('in delete' + req.params.picture_id);
    if(!req.params.picture_id) {
        res.json('NO PICTURE SPECIFIED TO DELETE');
    }
    else {

        mongo.connect(dbname, function(err, db){
            db.collection('pictures').findOne({ _id: Number(req.params.picture_id) },
                function(err, result) {
                    if (result){
                    console.log(result.creator_id);
                    console.log(req.session.user._id);
                    if (req.session.user._id == result.creator_id) {
                        db.collection('pictures').remove( { _id : Number(req.params.picture_id) },
                            function(err, delete_photo){
                                if (err) { return err }
                                if (!delete_photo) {
                                    res.json('Photo not found');
                                    }
                                else {
                                    res.json('Photo ' +  req.params.picture_id + ' deleted!');
                                    console.log(delete_photo);
                                    console.log(err);
                                    }
                            })
                        }
                    else {
                        res.json('Sorry, cannot delete, not your photo');
                    }
                }
                else {
                    res.sendFile('./pixi.html', {root: __dirname})
                }
            })
        })
```

Podría generar un sitio web malicioso con una petición GET a *http://localhost:8000/user_delete_photo/id_imagen* donde el id de la imagen pertenezca a una imagen propia del usuario logueado en la aplicación de Pixi.

Código de una página html simple (muy) con un enlace

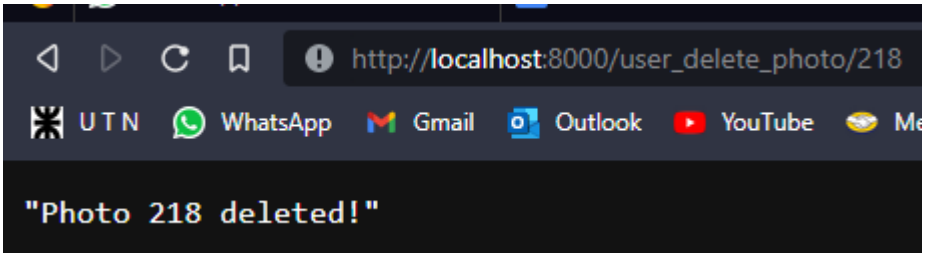
```
pagina_maliciosa.html > html > body > a
1  <html>
2      <body>
3          <a href="http://localhost:8000/user_delete_photo/218">Link misterioso</a>
4      </body>
5  </html>
```

Sitio levantado en mi maquina local



Link misterioso

Luego de que el usuario hizo click en el link



La foto fue eliminada correctamente!

7. **Correcciones a implementar:**
    1. Uso del flag SameSite para las cookies en “strict”, para que las cookies no se envien desde otros dominios.
    2. Utilizaría el método DELETE en lugar del método GET para eliminar las fotos, ya que los CSRF tokens tienden a no proteger las peticiones GET.
    3. Tampoco utilizaría un id autoincremental para las fotos (ni para los usuarios), de esta forma el atacante no podrá saber los identificadores de las imágenes.