

# 2º projeto

## Application Security Verification Standard (ASVS)



universidade  
de aveiro

[Alfredo Matos](#)  
[Vitor Cunha](#)  
[Paulo Bartolomeu](#)

Guilherme Lopes (103896)  
Gabriel Teixeira (107876)  
Pedro Rodrigues (102431)

<b>Introdução</b>	<b>2</b>
<b>Razão porque escolhemos esses problemas</b>	<b>3</b>
Proteção de Diretórios e Metadados Sensíveis	4
Segurança dos Tokens da sessão	7
Configuração CORS	9
Configuração path dos tokens de sessão	10
Gerar novo token	12
Tokens da sessão em ambiente seguro.	13
Limite tamanho de ficheiros	14
Anti-automation controls	15
Google reCAPTCHA	15
<b>Features</b>	<b>16</b>
Password strength evaluation	16
Multi-factor Authentication	17
<b>BIBLIOGRAFIA</b>	<b>18</b>

# Introdução

Neste projeto, foi realizada uma auditoria de segurança em uma loja online de memorabilia da DETI para atender aos requisitos do nível 1 do Application Security Verification Standard (ASVS). Com objetivo de identificar e corrigir vulnerabilidades sem comprometer a funcionalidade original da loja. Através de uma análise cuidadosa, abordamos falhas de segurança e implementamos soluções eficazes. A qualidade da documentação produzida é essencial, pois ela detalha tanto as vulnerabilidades encontradas quanto às correções aplicadas, refletindo a aplicação prática das nossas habilidades de codificação e conhecimento em segurança da informação.

## **Razão porque escolhemos esses problemas**

A principal razão que o levou a escolher os 6 problemas mais graves que optamos por revolver foi a sua relevância crítica para a integridade e eficiência do nosso sistemas.

Essa seleção baseou-se no impacto potencial na segurança operacional, a vulnerabilidade aos ataques cibernéticos. Usamos uma abordagem que visa não apenas remediar falhas imediatas, mas também fortalecer nossa postura de segurança a longo prazo, assegurando a proteção de dados e a confiança dos nossos clientes.

**Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS\_Store, .git or .svn folders**

Para resolver esse problema de segurança foi usado a diretiva (Foto 1 e Foto 2) no httpd.conf do Apache é uma configuração multifuncional. Onde Options -Indexes desativa a listagem de diretórios no servidor, prevenindo a visualização dos conteúdos dos diretórios, o +FollowSymLinks permite que o Apache siga links simbólicos dentro do sistema de arquivos, o +Includes habilita o suporte a Server Side Includes (SSI), e +ExecCGI permite a execução de scripts CGI. Essa combinação de opções otimiza a funcionalidade e segurança do servidor web.

Este código em baixo visa desativar a navegação em diretórios:

```
Options -Indexes +FollowSymLinks +Includes +ExecCGI
```

(FOTO 1)

```
# Virtual Hosts
<VirtualHost *:443>
    DocumentRoot "C:/xampp/htdocs/"
    ServerName localhost
    SSLEngine on
    SSLCertificateFile "conf/ssl.crt/server.crt"
    SSLCertificateKeyFile "conf/ssl.key/server.key"
    <Directory "C:/xampp/htdocs">
        Options -Indexes
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

(FOTO 2)

As diretivas <Files "Thumbs.db"> e <Files ".DS\_Store"> no arquivo httpd.conf do Apache foram usadas para negar acesso a arquivos específicos - Thumbs.db e .DS\_Store. Estes arquivos geralmente contêm metadados que podem revelar informações sobre a estrutura de diretórios e preferências de visualização, e seu acesso não é desejável em um ambiente web. A expressão Require all denied dentro destas seções bloqueia efetivamente o acesso a esses arquivos, aumentando a segurança da aplicação ao prevenir a exposição de informações potencialmente sensíveis

Código implementado:

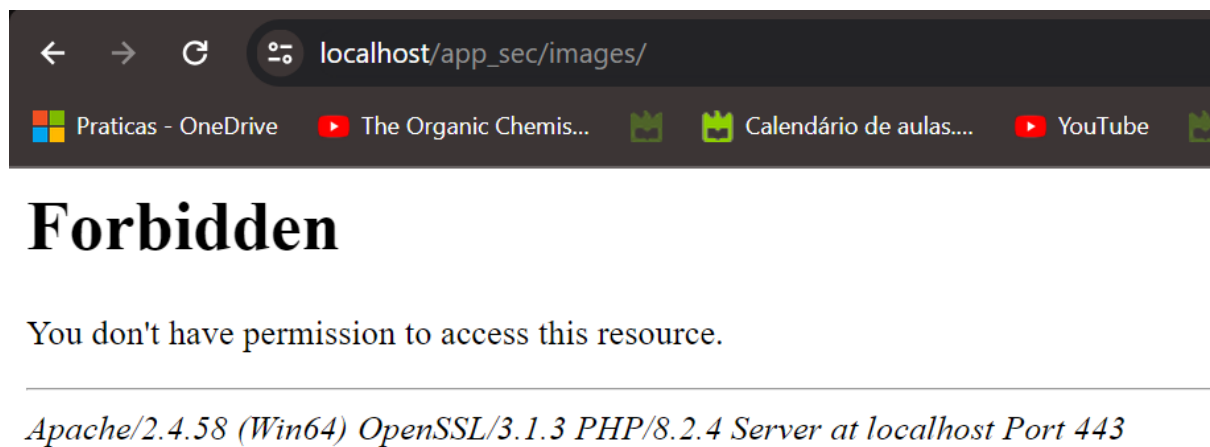
```
<Files "Thumbs.db">
|   Require all denied
</Files>
<Files ".DS_Store">
|   Require all denied
</Files>
```

Foram implementadas as diretivas <Directory ~ "\.git"> e <Directory ~ "\.svn"> no httpd.conf do Apache servem para bloquear o acesso aos diretórios .git e .svn pela web. Esses diretórios contêm informações sensíveis sobre o código-fonte e a estrutura do projeto, que não devem ser acessíveis publicamente. A expressão Require all denied impede o acesso a esses diretórios, protegendo-os de serem visualizados ou baixados por usuários não autorizados.

Código implementado:

```
<Directory ~ "\.git">
|   Require all denied
</Directory>
<Directory ~ "\.svn">
|   Require all denied
</Directory>
```

Exemplo em funcionamento **/images**:



Exemplo em funcionamento **/Thumbs-db**:



## Verify that cookie-based session tokens have the 'Secure' attribute set.

([C6](<https://owasp.org/www-project-proactive-controls/#div-numbering>))

Este problema é desenvolvido quando múltiplas aplicações web estão hospedadas sob um mesmo domínio, e elas compartilham o mesmo espaço de cookies, o que pode levar a problemas de segurança relacionados aos cookies de sessão. As quais incluem a possibilidade da aplicação substituir ou acessar os cookies de sessão de outra, comprometendo a confidencialidade e a integridade das informações da sessão.

Os principais riscos são:

**Substituição de Cookies:** Se duas aplicações configurarem cookies com o mesmo nome, o cookie mais recente pode substituir o anterior, causando conflitos na aplicação.

**Divulgação de Sessão:** Cookies de sessão de uma aplicação podem ser inadvertidamente acessados por outras aplicações hospedadas no mesmo domínio, levando a potenciais vazamentos de informações.

Logo para resolver esse problemas e futuros riscos, foi implementado o atributo path nos cookies de sessão. Esse atributo corresponde ao caminho específico da aplicação (/app\_sec/), assim assegura que os cookies sejam enviados e acessados apenas pela aplicação destinada, auxiliando a manter o isolamento e a segurança entre aplicações compartilhando o mesmo domínio.

Código php:

```
<?php
session_set_cookie_params(['path' => '/app_sec/']);
require_once 'vendor/autoload.php';
session_unset();
session_start();
$sessionId = session_id();
setcookie("PHPSESSID", $sessionId, [
    'expires' => time() + 3600,
    'path' => '/app_sec/',
    'secure' => true,
    'httponly' => true,
    'samesite' => 'Lax'
]);
```



Este código é executado no login de um utilizador, é responsável por redefinir os parâmetros do token da sessão, tornando a mais segura, o cookie terá a validade de 1 hora. Apenas deve ser enviado sobre conexões seguras HTTPS e é inacessível pelo cliente, ajudando a prevenir ataques XSS. O cookies também só será enviado em navegações de nível superior de outros domínios.

Outro código que auxilia na proteção:

```
<IfModule headers_module>
    Header always edit Set-Cookie (.*) "$1; Secure; HttpOnly"
    RequestHeader unset Proxy early
</IfModule>
```

Ao mesmo tempo nós configuramos no servidor web Apache, dentro do arquivo .htaccess (httpd.conf), isto visa reforçar a segurança dos atributos 'Secure' e 'HttpOnly' a todos os cookies exercidos.

O módulo headers\_module permite a manipulação dos cabeçalhos HTTP das requisições e respostas. A seguinte linha "Header always edit Set-Cookie (.\*) \"\$1; Secure; HttpOnly\"", esta visa modificar o cabeçalho Set-Cookie de todas as respostas HTTP enviadas pelo servidor, ao mesmo tempo vai atribuir os atributos aos cookies.

Onde atributo Secure vai segurar que o cookie só possa ser enviado através de uma conexão HTTPS, enquanto HttpOnly previne que os cookies sejam acessados via JavaScript, reduzindo o risco de ataques de cross-site scripting.

O RequestHeader unset Proxy early: Esta linha remove o cabeçalho Proxy das requisições recebidas pelo servidor, prevenindo os atacantes interceptarem as comunicações entre servidores, esta linha já é predefinida usamos o apache.

Foto de um exemplo de resultado:

The screenshot shows the Chrome DevTools Network tab with the 'Cookies' sub-tab selected. The address bar shows 'cookie-domain:localhost'. The network log shows a request to 'produto.php?id=1'. The 'Request Cookies' table is displayed below the network log.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Sec...	SameSite	Partition...	Priority
PHPSESSID	2akn37v13n61qfndj3eq7pq89	localhost	/app_sec/	2023-12-28T17:59:14.482Z	35	✓	✓	Lax		Medium
PHPSESSID	hfbq474t98d14k9km4fia950r4	localhost	/	Session	35					Medium

## Verify that the Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin header uses a strict allow list of trusted domains and subdomains to match against and does not support the "null" origin

Ao abordar a configuração do CORS (Cross-Origin Resource Sharing) em servidores web, como no nosso caso o Apache. O CORS é um mecanismo que permite que recursos em um servidor web sejam solicitados por um domínio diferente daquele que serviu o primeiro recurso. Isso é importante em ondas de aplicações web que interagem frequentemente com múltiplos serviços e APIs hospedados em domínios distintos.

Código:

```
<IfModule mod_headers.c>
    Header set Access-Control-Allow-Origin "https://localhost/app_sec"
    Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"
    Header set Access-Control-Allow-Headers "Content-Type"
    Header always set Access-Control-Max-Age "1000"
    Header always set Access-Control-Allow-Credentials "true"
</IfModule>
```

Este código é uma configuração do Apache que define os cabeçalhos de resposta para a Política de Compartilhamento de Recursos de Origem Cruzada (CORS). Vamos analisar cada linha para perceber a sua função:

Na primeira linha, **<IfModule mod\_headers.c>**, esta permite a modificação dos cabeçalhos HTTP nas respostas. Em seguida a **Header set Access-Control-Allow-Origin "https://localhost/app\_sec"** serve para ver quais origens tem acesso aos recursos do servidor, se não tiver ele restringe todas as que não tiverem origem no servidor. O **Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"**, são os métodos permitidos no servidor, seguida a **Header set Access-Control-Allow-Headers "Content-Type"** aqui o servidor diz os cabeçalhos que podem ser usados durante a sua solicitação, neste caso só permite cabeçalho Content-Type. De seguida a **Header always set Access-Control-Max-Age "1000"** esta limita o tempo de resposta ao pedido que pode ser armazenado em cache (1000 segundos) e na última linha **Header always set Access-Control-Allow-Credentials "true"** última e não menos importante indica se as credenciais podem ser usadas em solicitações cruzadas se tiverem são permitidas como no nosso caso e por fim o seu fechamento do bloco **</IfModule>**.

Em síntese, esta configuração foi usada para implementar uma política CORS específica no nosso servidor Apache, definindo quais origens, métodos, cabeçalhos e credenciais são permitidos para solicitações de origem cruzada, mantendo a segurança da nossa aplicação.

Exemplo da sua implementação:

```
C:\Users\Gabri>curl -H "Origin: http://exemplodominio.com" -I http://localhost/app_sec/index.php
HTTP/1.1 200 OK
Date: Thu, 28 Dec 2023 18:45:22 GMT
Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
Access-Control-Max-Age: 1000
Access-Control-Allow-Credentials: true
X-Powered-By: PHP/8.2.4
Set-Cookie: PHPSESSID=1778r05gr577142rp80jtna757; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Access-Control-Allow-Origin: https://localhost/app_sec
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers: Content-Type
Content-Type: text/html; charset=UTF-8
```

**Verify that if the application is published under a domain name with other applications that set or use session cookies that might override or disclose the session cookies, set the path attribute in cookie-based session tokens using the most precise path possible.**

([C6](<https://owasp.org/www-project-proactive-controls/#div-numbering>))

O problema exercido aqui é quando temos várias aplicações sob um mesmo nome de domínio, isso pode provocar que elas usem cookies de sessão dos utilizadores. Mas se esses cookies não forem configurados corretamente visa ocorrer problemas de segurança por causa da sobreposição ou divulgação não intencional das informações da aplicação.

E para resolver este problema for acrescentado ao código usado para resolver o problema acima a seguinte frase: Header set Set-Cookie "Path=/app\_sec/; HttpOnly; Secure"

```
<IfModule mod_headers.c>
    # CORS headers
    Header set Access-Control-Allow-Origin "https://localhost/app_sec"
    Header set Access-Control-Allow-Methods "GET, POST, OPTIONS"
    Header set Access-Control-Allow-Headers "Content-Type"
    Header always set Access-Control-Max-Age "1000"
    Header always set Access-Control-Allow-Credentials "true"

    Header set Set-Cookie "Path=/app_sec/; HttpOnly; Secure"

</IfModule>
```

Para isso foram implementadas configurações específicas em cookies de sessão. Isso incluiu a restrição do escopo do cookie para URLs que começam com '/app\_sec/', garantindo que o cookie seja válido apenas para um conjunto específico de páginas ou aplicativos. Além disso, foram aplicados os atributos 'HttpOnly' e 'Secure' aos cookies, que impedem o acesso por scripts JavaScript e asseguram que o cookie só seja transmitido em conexões seguras (HTTPS), respectivamente. Essas medidas contribuem significativamente para proteger informações sensíveis, como tokens de autenticação, e melhorar a integridade das sessões dos utilizadores, diminuindo os riscos de conflitos ou divulgação inadequada de informações de sessão.

Foto do resultado da sua implementação:

```
C:\Users\Gabri>curl -I http://localhost/app_sec/
HTTP/1.1 200 OK
Date: Fri, 29 Dec 2023 15:04:00 GMT
Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
Access-Control-Max-Age: 1000
Access-Control-Allow-Credentials: true
X-Powered-By: PHP/8.2.4
Set-Cookie: PHPSESSID=hfiuh5gr3nktndni7kkslha6em; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Access-Control-Allow-Origin: https://localhost/app_sec
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers: Content-Type, Authorization, X-Requested-With
Content-Type: text/html; charset=UTF-8
```

**Verify the application generates a new session token on user authentication.**

**([C6]([https://www.owasp.org/index.php/OWASP\\_Proactive\\_Controls#tab=Formal\\_Numbering](https://www.owasp.org/index.php/OWASP_Proactive_Controls#tab=Formal_Numbering)))**

Apesar de criar uma nova sessão o token algumas vezes tinha o mesmo valor da sessão anterior. O que causa uma insegurança das sessões anteriores. Para resolver a insegurança é regenerado um novo valor do token de sessão utilizando a função nativa de php **session\_regenerate\_id()**. No logout da sessão foi adicionado a expiração da validade; de modo que o token se torne inválido.

Após estas alterações os tokens das sessões são sempre geradas no seu início e depois do logout o token da sessão não aparece mais no browser.

## Verify the application only stores session tokens in the browser using secure methods such as appropriately secured cookies (see section 3.4) or HTML 5 session storage.

Para garantir a segurança dos tokens da sessão é necessário ser guardado num ambiente seguro.

Inicialmente é atribuído ao token o atributo **Httponly**, que impede que seja acessado por scripts maliciosos.

Para tornar o ambiente seguro configuramos um certificado SSL.

O XAMPP tem uma ferramenta **makecert**, que permite criar um certificado SSL auto assinado. O certificado deve ser instalado em Trusted Root Certification Authorities, para o certificado ser aceito pelos browsers.

Depois é necessário configurar o ficheiro **httpd-vhosts.conf**, para redireccionar automaticamente para HTTPS, garantindo que todas as solicitações sejam redireccionadas para a versão segura do site. Já a porta 443 lida com as solicitações seguras.

```
# Virtual Hosts
<VirtualHost *:80>
ServerName localhost
Redirect / https://localhost/
</VirtualHost>

# Virtual Hosts
<VirtualHost *:443>
    DocumentRoot "C:/xampp/htdocs/"
    ServerName localhost
    SSLEngine on
    SSLCertificateFile "conf/ssl.crt/server.crt"
    SSLCertificateKeyFile "conf/ssl.key/server.key"
    <Directory "C:/xampp/htdocs">
        Options -Indexes
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Deste modo os tokens de sessão apenas são usados em ligações seguras (HTTPS).

Verify that the application will not accept large files that could fill up storage or cause a denial of service.

Ao adicionar um produto no site é necessário introduzir uma imagem do mesmo. Ao não haver um limite para o tamanho dessa imagem pode levar a problemas de esgotamento de espaço em disco, causando a interrupção de serviços essenciais.

Para limitar o tamanho da imagem carregada, o tamanho da imagem é verificado antes do produto ser inserido para database. O produto só será adicionado se a imagem não for maior que 2MB.

```
$maxFileSize = 2 * 1024 * 1024;

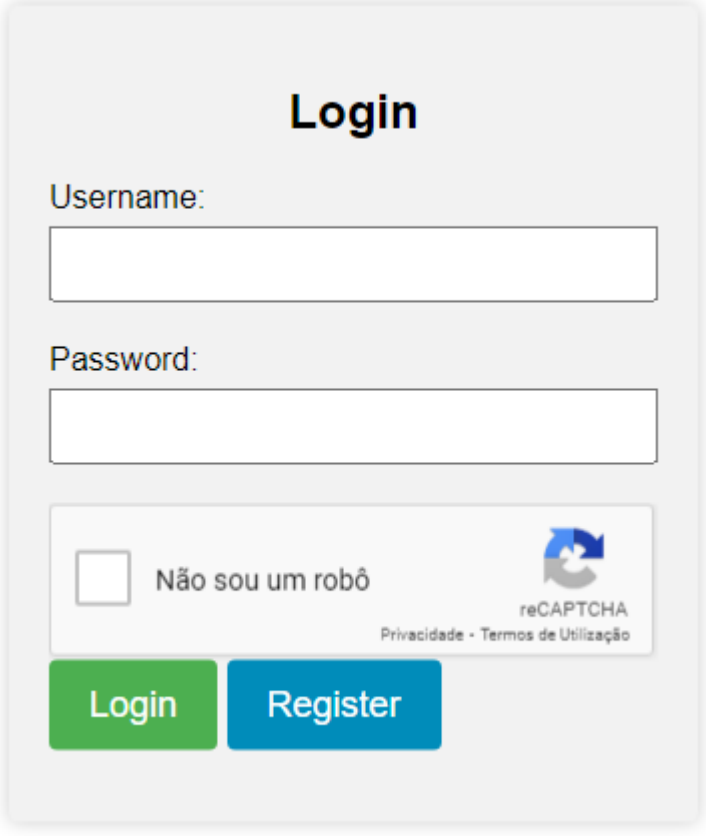
if (in_array($imageFileType, $extensions_arr)) {
    // Check if the file size is within the allowed limit
    if ($_FILES["product_img"]["size"] > $maxFileSize) {
        echo "Error: File size exceeds the maximum allowed limit.";
    } else {
        $insert_product = "INSERT INTO products (name, price, descr, stock, img, category)
        mysqli_query($con, $insert_product);
        move_uploaded_file($_FILES['product_img']['tmp_name'], $target_dir . $product_img);
        header("location: admin.php");
    }
} else {
    echo "Error: Invalid file format. Only JPG, JPEG, PNG, and GIF files are allowed.";
}
```

# Anti-automation controls

## Google reCAPTCHA

Para distinguir entre interações humanas e atividades automáticas no processo de login e registo foi implementado o reCAPTCHA V2. O reCAPTCHA passa por um desafio de selecionar uma caixa de seleção para provar que são humanos. Se se suspeitar da integridade da operação podem ser apresentados desafios adicionais como a identificação de objetos em imagens.

Para a implementação do desafio foi necessário criar um projeto no Google Cloud para gerar a private key e a public key. E integrar o script dado pela Google para proteger os forms contra ataques automatizados.



The image shows a login form with a light gray background. At the top, the word "Login" is centered in a bold, black font. Below it, there are two input fields: "Username:" and "Password:". Each label is followed by a white rectangular input box with a thin gray border. Below the password field, there is a reCAPTCHA widget. It consists of a small square checkbox on the left, followed by the text "Não sou um robô". To the right of the checkbox is the reCAPTCHA logo, which is a blue and gray circular icon. Below the logo, the text "reCAPTCHA" is visible, followed by a smaller link "Privacidade - Termos de Utilização". At the bottom of the form, there are two buttons: a green "Login" button and a blue "Register" button, both with white text.



# Features

## Password strength evaluation

A avaliação da segurança da password é definida segundo ASVS 2.1.

A password strength meter foi implementada em javascript para ajudar o utilizador a definir uma password segura. No entanto existe uma verificação adicional do lado do server.

O último passo da verificação é testar a password a usando Have I Been Pwned API.

A verificação passa por fazer um request com o prefixo (os 5 primeiros caracteres) da hash da password, a API responde com uma lista de sufixos e a contagem de quantas vezes esses sufixos aparecem na base de dados do Pwned Passwords.

Se o sufixo da password não se encontrar na lista recebida podemos considerar a nossa password como segura.

## User Registration

Username:

Email:

Password:

Confirm Password:

[Register](#)

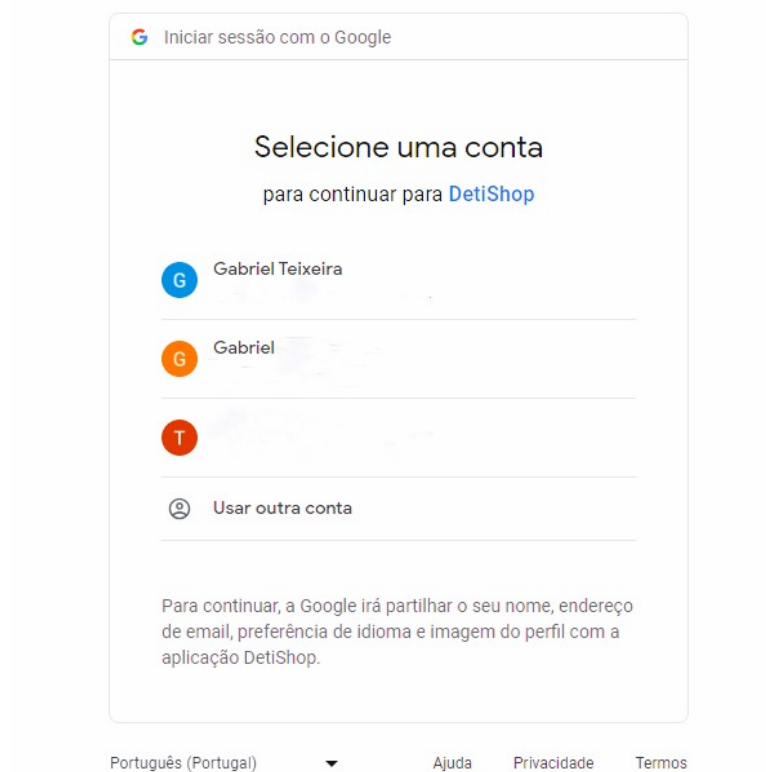
# Multi-factor Authentication

Como segundo fator de autenticação foi escolhido o Google Identity.

A sua implementação é similar ao do reCAPTCHA, sendo também criado no Google Cloud.

A autenticação é chamada após o login 'principal' ser feito, a sessão apenas é válida após a autenticação pelo Google. É importante realçar que nenhuma informação da conta Google é guardada.

Este processo em conjunto com o reCAPTCHA reforça a autenticidade de quem está a conectar ao sistema.



# BIBLIOGRAFIA

<https://fedingo.com/how-to-disable-directory-browsing-in-apache/>  
<https://docs.vultr.com/how-to-disable-directory-browsing-on-apache>  
<https://linuxconfig.org/turn-off-directory-browsing-on-apache>  
[https://enable-cors.org/server\\_apache.html](https://enable-cors.org/server_apache.html)  
<https://tecadmin.net/enable-cors-apache/>

<https://owasp.org/www-community/controls/SecureCookieAttribute>  
[https://cheatsheetseries.owasp.org/cheatsheets/Session\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html)

<https://www.youtube.com/watch?v=eqrDHkIFe8U>

<https://code.tutsplus.com/create-a-google-login-page-in-php--cms-33214t>