SAW – Trabalho Prático

segurança das Aplicações web

André Pinto (8200613) / SERGIO Félix (8200615)

2022

Índice

[Introdução 1](#_Toc92837032)

[Requisitos 1](#_Toc92837033)

[Arquitetura do Projeto 1](#_Toc92837034)

[Método de Funcionamento 1](#_Toc92837035)

[Implementação e Segurança 1](#_Toc92837036)

[Roles e Áreas Restritas 1](#_Toc92837037)

[Sanitização de Dados 1](#_Toc92837038)

[Validação de Dados 1](#_Toc92837039)

[Register 1](#_Toc92837040)

[Atempt Login 1](#_Toc92837041)

[Remember me 1](#_Toc92837042)

[Encriptação Cookies 1](#_Toc92837043)

[Recover Password 1](#_Toc92837044)

[Error Handling 1](#_Toc92837045)

[Logs / Erros 1](#_Toc92837046)

[Bloqueio de Utilizadores 1](#_Toc92837047)

[Upload de Imagens 1](#_Toc92837048)

[Conclusão 1](#_Toc92837049)

[Bibliografia 1](#_Toc92837050)

[Apresentações da Disciplina SAW 1](#_Toc92837051)

[Documentação PHP 1](#_Toc92837052)

[Pesquisa e Aprendizagem 1](#_Toc92837053)

# Introdução

No âmbito da disciplina de Segurança das Aplicações WEB do 2º ano, 1º semestre do Curso Técnico Superior Profissional em Desenvolvimento para a Web e Dispositivos Moveis, foi pedido o desenvolvimento de uma aplicação Web em PHP, que simule uma aplicação de venda de produtos usados. Neste, deve ser utilizado todo o conhecimento adquirido nas aulas, implementando soluções de segurança aprendidas e que contemplasse todas as funcionalidades pedidas no enunciado.

# Requisitos

Esta aplicação Web foi exclusivamente desenvolvida em código PHP 8. Para tal, será necessário ter em conta alguns requisitos para conseguir executar a aplicação. As vulnerabilidades não estão apenas no código implementado, algumas delas são provocadas por configurações defeituosas nos servidores que abrem portas para invasões. Para tal segue todos os requisitos necessários para correr a aplicação de forma mais segura.

Em primeiro lugar é necessário disponibilizar um serviço Apache para correr o código PHP 8. Para tal foi utilizado o programa XAMP, em que é possível ligar um serviço desta natureza. Para além disso, foram aplicadas algumas configurações para aumentar segurança e a confidencialidade dos dados, nomeadamente:

* Desativação da lista de diretórios.
* Desativação de módulos e serviços PHP desnecessários.
* Desativação do serviço Apache server-info e server status.
* Desativação da diretiva ServerSignature.
* Desativação do rastreamento HTTP Request.
* Redução do tempo de timeout para prevenir ataques DoS.
* Ligação HTTPS – para permitir a encriptação dos dados de ponto a ponto (User - Servidor).

Em relação á persistência de dados, foi utilizado um banco de dados MySql. Para ativar este serviço, igualmente ao servidor Apache, foi utilizado o programa XAMP, que possibilita a criação deste serviço. Foram aplicadas algumas configurações para aumentar segurança e a confidencialidade dos dados, nomeadamente:

* Alteração da password da conta ROOT
* Criação de um utilizador administrador, com todas as permissões apenas para a base de dados a utilizar.
* Criação uma conta apenas com permissões de leitura e escrita na base de dados a utilizar para configurar na aplicação.

Assim, são estes os requisitos básicos e alterações necessárias para aumentar a segurança da nossa aplicação WEB.

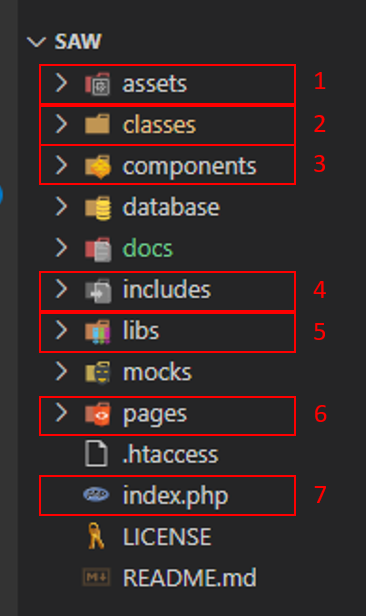
# Arquitetura do Projeto

Este trabalho, não seria um projeto de grandes dimensões, mas por nossa opção decidimos criar uma estrutura do projeto que fosse capaz de organizar todos os módulos necessários de uma forma intuitiva e organizada.

A estrutura criada é sólida e escalável. Temos os módulos devidamente separados a funcionar de forma independente. Assim, permite-nos uma maior reutilização de código e uma melhor legibilidade do projeto.

## Método de Funcionamento

**Pastas**

1. Pasta para armazenar, scripts, estilos e as imagens incluindo os uploads.
2. Classes responsáveis pela logica da aplicação e inclui validação dos dados
3. Componentes de layout. Navbar , Footer…
4. Classes auxiliares, normalmente funções reutilizáveis.
5. Bibliotecas externas
6. Páginas a serem carregadas. Inclui sanitização dos dados.
7. Ficheiro inicial da aplicação

**Index.php**

Uma imagem com texto

Descrição gerada automaticamenteO projeto inicia-se no ficheiro “index.php”. Este, apenas é responsável por iniciar a aplicação. Cria um objeto do tipo “Application”, em que a suas principais funções é explodir o link URL para possibilitar uma navegação por rotas (1). Também é possível definir alguns parâmetros globais para posteriormente serem utilizados em outros locais (1). Por fim importa o ficheiro que contem o layout (2)

**layout.php**

Em relação ao ficheiro do “layout.php”, é responsável por importar a estrutura da aplicação. É visível que importa o componente da NavBar e o Footer. O mais importante é que este ficheiro importa o controlador das páginas. O que possibilita as rotas da aplicação.

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamenteConsoante o URL, a aplicação carrega páginas diferentes, permitindo a navegação sem ter de repetir o Navbar e o Footer.

# Implementação e Segurança

A implementação de sistemas de segurança foi o principal objetivo deste trabalho. Para tal, implementamos vário conceito aprendido durante as aulas.

## Roles e Áreas Restritas

A aplicação conta com a implementação de restrições de acesso e roles. Foram definidos 3 grandes grupos de autenticação distintos, em que cada um terá diferentes permissões de acesso às páginas. Os grupos de acesso são os seguintes:

* **Utilizadores não autenticados** - Todos os utilizadores que não possuírem autenticação, poderão apenas navegar pelos artigos listados pelos vendedores. No entanto não serão mostrados todos os dados do vendedor, visto que é um utilizador não autenticado, não será mostrado o contacto do vendedor.
* **Utilizadores autenticados** - Todos os utilizadores que possuem autenticação, poderão navegar pelos artigos sem nenhuma restrição. Para alem disso têm acesso ao painel de administração, onde podem criar artigo, alterar os seus artigos e visualizar e alterar o seu perfil. Por questões de segurança, não é possível alterar o email.
* **Utilizadores administradores** – Um utilizador Administrador, herda todas as características de um utilizador autenticado. No então tem a possibilidade, através do painel de administração, visualizar todos os utilizadores, podendo bloquear ou desbloqueá-los. Como funcionalidade extra, para os administradores também é possível aceder à página de administração de log, onde podem monitorizar o funcionamento da aplicação.

## Sanitização de Dados

O primeiro ponto a ser implementado foi a sanitização dos dados. Esta funcionalidade visa filtrar todos os inputs do utilizador para evitar ataques como SQL Injection entre outros. Após o utilizador inserir os dados, a primeira camada é imediatamente a sanitização e está colocada em todos os locais onde existe inputs do utilizador.

Para sua implementação, passamos todos os valores para o array data. De seguida, no array args, definimos o tipo de sanitização que pretendemos aplicar. Por último realizamos filter\_var\_array para executar as sanitizações.

Este procedimento é incluído em todos os inputs do utilizador.

Uma imagem com texto

Descrição gerada automaticamente

## Validação de Dados

Com a validação de dados, permite-nos restringir as características dos valores que o utilizador pode inserir. Com isto podemos definir limites ou obrigatoriedades para os valores inseridos como a password ter obrigatoriamente 8 dígitos e símbolos.

Para sua implementação, incluímos para cada parâmetro as verificações necessárias para a sua validação. Caso esta falhe, será utilizado um sistema de ErrorHandling para mostrar ao utilizador os parâmetros inválidos.

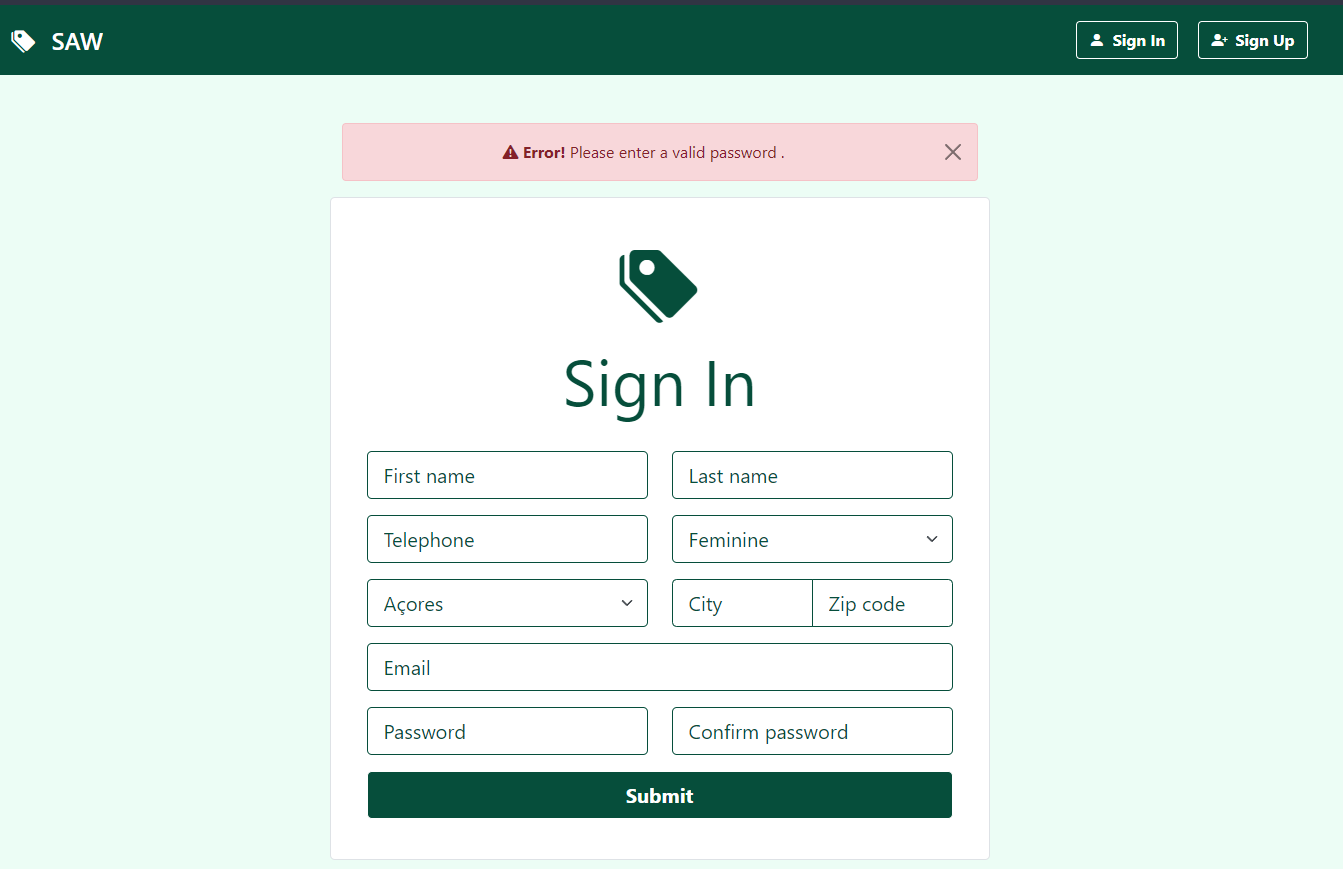
Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente Esta implementação é colocada para todos os dados inseridos pelo utilizador.

## Register

O registo de utilizadores é umas das funcionalidades pedidas do enunciado, que do qual é necessário ter em atenção alguns critérios para manter a segurança da aplicação.



Este formulário é responsável por recolher os dados inseridos pelo utilizador e criar um novo registo na base de dados. Mas, antes disso, são aplicados vários passos de segurança. Nomeadamente, sanitização para prevenir ataques de SQL Injection e validação para restringir possíveis dados que o utilizador coloque que não possam ser aceites.

Após passar por estas validações o utilizador já pode ser criado. No entanto, temos de ter em conta uma característica importante. Temos de garantir a confidencialidade da password do utilizador. Devido á sua enorme importância é necessário garantir a confidencialidade, mesmo em caso de vazamento de dados. Assim é necessário armazenar de forma encriptada este valor na base de dados. Para tal, em vez da password guardamos um HASH dela própria. Utilizamos uma função PHP “password\_hash” com o algoritmo “bcrypt” que nos fornece um hash de 128-bit, perfeitamente capaz para a nossa necessidade.

Assim garantimos que o processo de desencriptação é impossível. Alguma comparação futura terá de ser através da password original que apena só utilizador conhece.

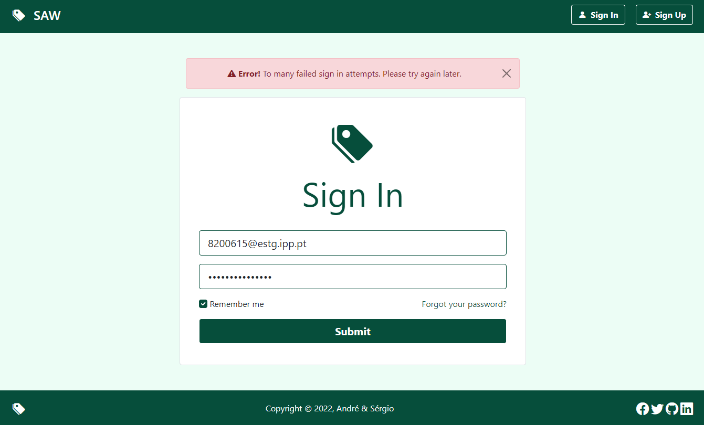
## Atempt Login

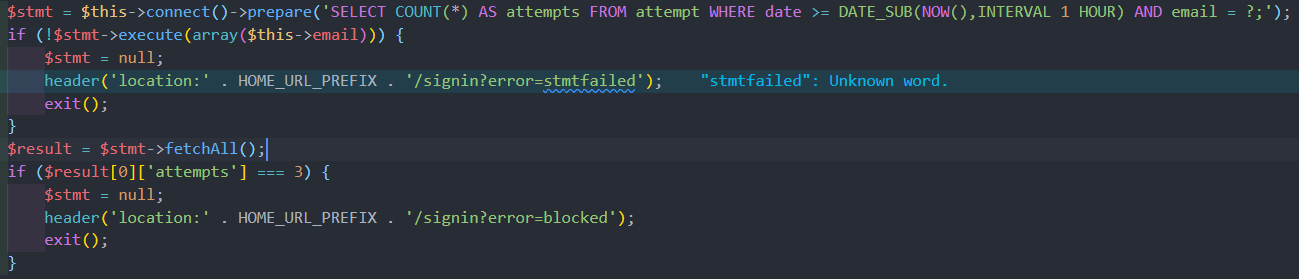
Os ataques de “BrutForce” são uma realidade, para prevenir, implementar uma camada de segurança á nossa aplicação para conseguir mitigar este problema. Estes ataques provem de inúmeras tentativas de login para conseguir adivinhar a password do utilizador. Para tal, definimos que a password tem de cumprir um conjunto de requisitos mínimos.

Uma imagem com texto

Descrição gerada automaticamenteA password tem de conter letras maiúsculas e minúsculas, números, caracteres especiais e pelo menos 8 caracteres. Implementação feita através da camada de validação de dados que realizamos em todas as entradas de dados do utilizador.

Para além de obrigar o utilizador a utilizar uma password mais segura, adicionamos um sistema de “AtemptLogin”. Sempre que um utilizador realiza uma tentativa de Login falhada, é registado na base de dados. Caso exista 3 tentativas falhadas nos últimos 1 horas, a conta é bloqueada. Apenas é possível realizar login na hora seguinte.

Este procedimento visa o bloqueio da conta caso o utilizador exceda o número de tentativas máximo.



## Remember me

No meio de tanta segurança, a usabilidade pode ficar um pouco afetada, para isso temos de adicionar funcionalidade que melhorem a usabilidade do utilizador. Para tal, colocamos a propriedade “Remember Me” na magina de login. Assim o utilizar não necessita de escrever sempre os dados de log in por estes estão guardados em cookies. Caso estes dados não estivessem encriptados, estaríamos a cometer uma enorme falha de segurança pois estes podem ser facilmente acedidos.

Uma imagem com texto

Descrição gerada automaticamente Para tentar garantir uma maior proteção para o utilizador utilizamos algoritmos mais complexos e modernos de encriptação. Estes dados são guardados em cookies após passarem pelo um processo de encriptação e é agendado um prazo para a cookie expirar.

### Encriptação Cookies

Inicialmente utilizamos a função PHP “openssl\_encrypt” com uma chave ($key), cifra “AES-128-CBC” com o algoritmo “sha256”. Pós este passo, geramos um hash das propriedades de encriptação do primeiro passo e voltamos a realizar uma segunda camada de encriptação através da função PHP “base64\_encode” em que a chave de encriptação é o hash gerado anteriormente.

Aplicando este 2 nível de encriptação, aumentamos a segurança dos dados que estão bastante expostos.

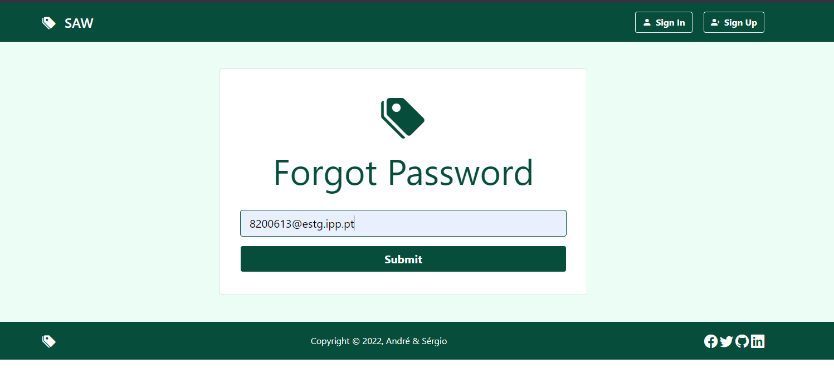
Para a desencriptação é aplicado exatamente o processo oposto.

Uma imagem com texto

Descrição gerada automaticamente

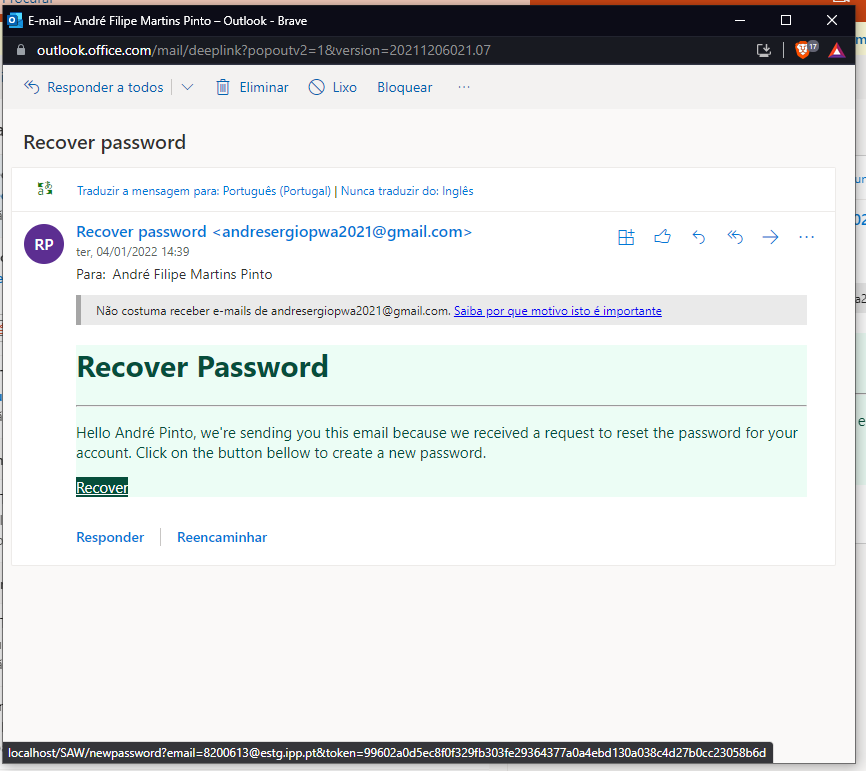
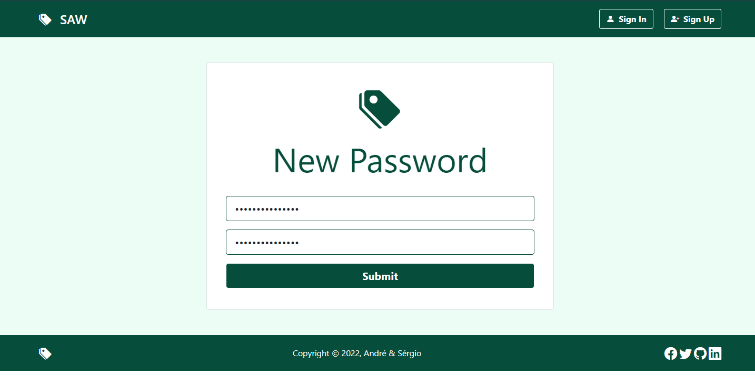
## Recover Password

Possibilitar ao utilizador a recuperação da password é uma funcionalidade muito importante, que da qual, necessita especial atenção para a sua implementação. É necessário garantir que todo o processo seja fácil e intuitivo sem esquecer que manter a segurança acima de tudo para evitar roubos ou acesso a informação privada.

Inicialmente, o utilizador insere o email e realiza um pedido de alteração de password.

Posteriormente, a aplicação irá tomar uma série de paços para responder a este pedido:

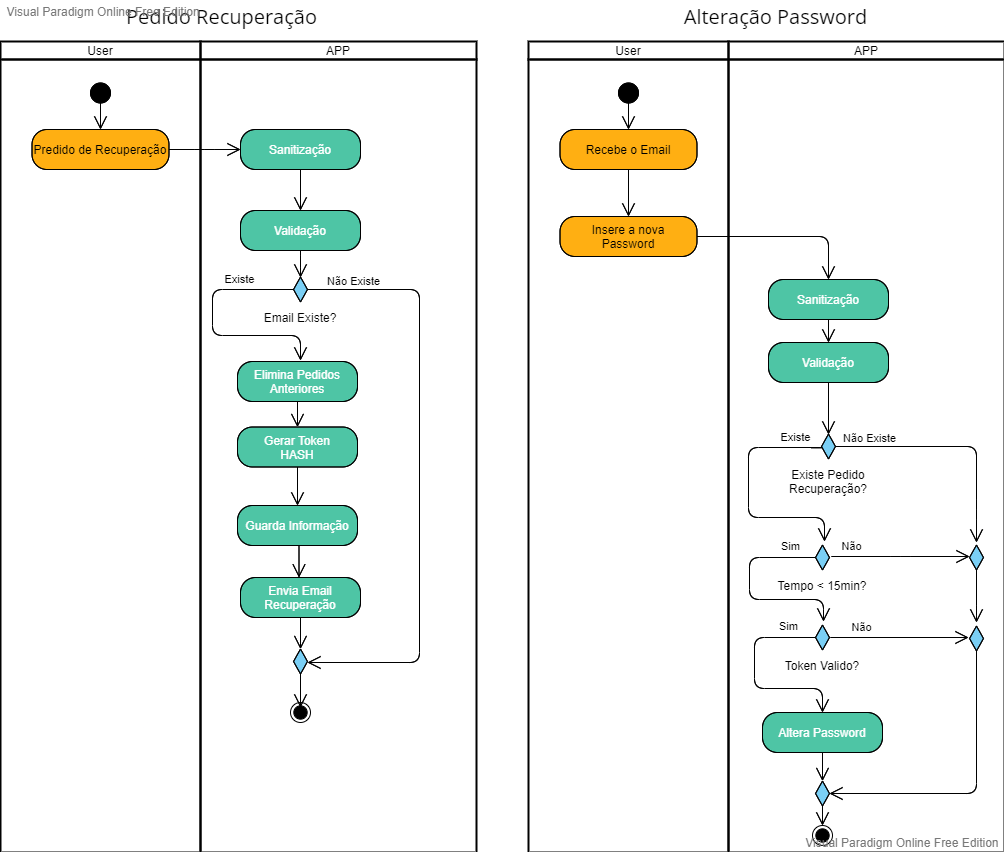
* Realiza a sanitização e validação do email
* Verifica a existência do email
* Elimina pedidos de recuperação anteriores que possam existir na tabela.
* Gera um token aleatório de 32 bytes para que posteriormente este seja enviado para o utilizador e seja armazenado na base de dados passando pelo um processo de HASH através da função PHP “password\_hash” com o algoritmo “bcrypt” que nos fornece um hash de 128-bit, perfeitamente capaz para a nossa necessidade.
* Envia um email ao utilizador com os próximos passos a seguir. Assim garantimos que esta alteração apenas será feita pelo proprietário.



Seguidamente o utilizador abre o link indicado no email e será redirecionado para a página de alteração da password e o utilizador terá de colocar a nova password e efetivar a alteração. Caso o token presente no link não seja válido ou tenha excedido os 15 min para realizar a recuperação o utilizador é automaticamente redirecionado para a página de login e não lhe é permitido realizar a alteração.

Todos os dados também são sanitizados e posteriormente validados e por fim é efetivada a alteração da password.

**Diagrama de Atividades de todo o processo de recuperação**.

****

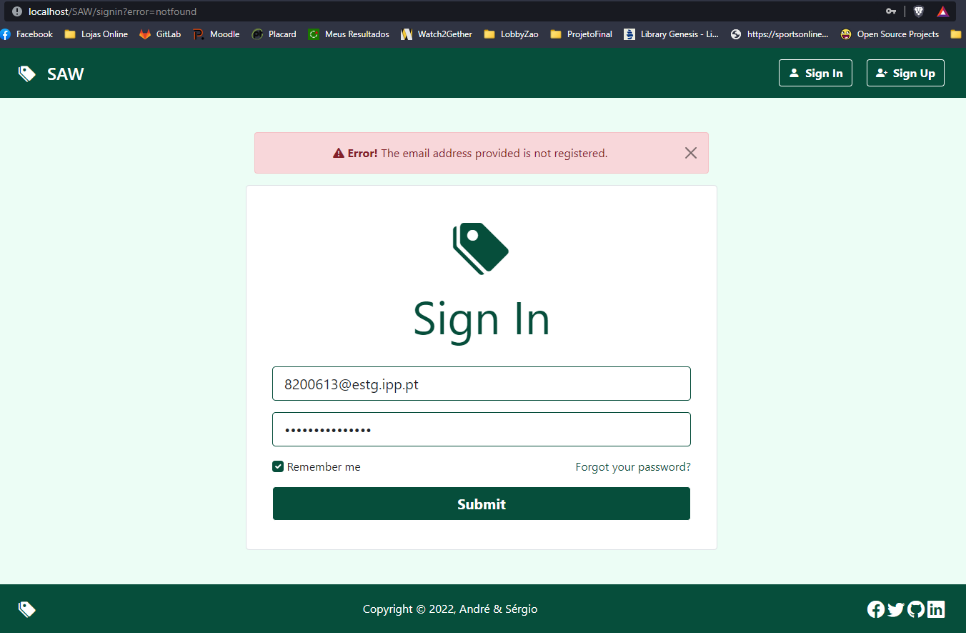
## Error Handling

Uma imagem com texto

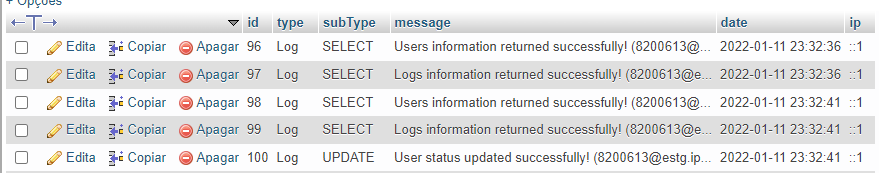
Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamente Esta é uma funcionalidade básica, utilizada para conseguir mostrar aos utilizadores erros que possam estar a acontecer durante a utilização da aplicação. Para tal, criamos numa classe responsável pela tradução de códigos de erros internos para mensagens visíveis para o utilizador. Assim quando acontecer algum erro, redirecionamos o utilizador para a mesma página com o parâmetro no URL “error” com o código do erro. Posteriormente é apresentado a mensagem de erro ao utilizador.

Deste modo conseguimos garantir uma maior segurança para a nossa aplicação pois não serão mostrados erros do PHP direto para os utilizadores, podendo expor informação sensível.



## Logs / Erros

 Em todas as aplicações os erros são inevitáveis. No entanto podemos armazená-los para uma possível análise. Não só podem ser detetados bugs de software como também possíveis tentativas de ataques informáticos. Implementamos a funcionalidade de registar logs ou erros. Para tal, criamos uma tabela na base de dados específica para armazená-los sempre que seja necessário.

Posteriormente os logs poderão ser utilizados pelos administradores, para monitorizarem o desempenho da nossa aplicação

Uma imagem com mesa

Descrição gerada automaticamente

## Bloqueio de Utilizadores

O bloqueio de utilizadores é uma funcionalidade que visa restringir o acesso do utilizar sempre que assim seja necessário. Esta funcionalidade apenas pode ser utilizada por utilizadores administradores através do painel de administração. Assim caso o utilizador esteja a prejudicar o bom funcionamento da aplicação ou mesma pela necessidade de aplicar sanções, este poderá ver a sua conta bloqueada e ser-lhe-á negado o acesso.

Uma imagem com mesa

Descrição gerada automaticamente

## Upload de Imagens

Uma imagem com mesa

Descrição gerada automaticamente No que toca a upload de imagens, também implementamos algumas técnicas para evitar o acesso fácil ao conteúdo. Sendo imagens publicas, o upload é realizado para uma pasta publica da nossa servidora, no entanto, esta não é indexada. Ou seja, caso o utilizador aceda diretamente á pasta, não lhe será mostrado todos o conteúdo das imagens. Apenas terá acesso caso saiba o link exato da mesma. Mas não é só, todas as imagens são renomeadas antes de serem guardadas. Assim, geramos um nome aleatório para cada imagem, dificultando a obtenção da mesma.

Como podemos verificar, todas as imagens têm um nome aleatório, o que será muito difícil para uma mal-intencionado ter acesso a todas imagens de uma forma fácil.

# Conclusão

Neste trabalho, foi desenvolvido uma aplicação web de venda de produtos usados. Nela, para além das funcionalidades básicas, implementamos várias soluções para aumentar a segurança da nossa aplicação de modo a efetivar o que foi aprendido nas aulas lecionadas.

Graças á nossa dedicação, foi possível cumprir todos os objetivos idealizados para a realização deste projeto e melhorar os nossos conhecimentos sobre a área de segurança de aplicações.

Conseguimos organizar todos o código para proporcionar uma maior escalabilidade e organização do mesmo. Implementamos um sistema para navegação por rotas de modo a evitar repetição de código; sanitização para evitar ataques indesejado; validação para restringir os dados que podemos aceitar; encriptação para evitar o acesso a dados não autorizados entre muitas mais funcionalidades.

Não só crescemos a nível de conhecimento, como também aumentou a nossa sensibilização para esta área. Muitas das vezes o objetivo é ter uma aplicação funcional e não damos o devido valor á segurança. No entanto, não nos interessa ter uma aplicação funcional, caso esta seja facilmente invadida e se torne inútil, ou por outro lado, o vazamento de informação. Ambos podem causar grandes prejuízo às empresas, por isso segurança é um dos pontos mais importantes durante o desenvolvimento de aplicações. É importante para nós como também para os utilizadores.

Achamos que foi um projeto bem conseguido, que do qual estamos orgulhosos de todos o processo de aprendizagem e o resultado final.

# Bibliografia

## Apresentações da Disciplina SAW

* Aulas Teóricas
* Apresentações

## Documentação PHP

* <https://www.php.net/>

## Pesquisa e Aprendizagem

* <https://pt.stackoverflow.com/>
* <https://www.youtube.com/>
* <https://www.udemy.com/pt/>
* <https://github.com/>