Politécnico de Coimbra

COIMBRA BUSINESS SCHOOL
iscac.pt

Projeto e Desenvolvimento Informático

3.º Ano da Licenciatura em Informática de Gestão

Ano Letivo

2023-2024

Atividade 4

Relatório de Testes do Projeto DevLancer

Autores

Joana Esteves (2021143113)

Tomás Figueiredo (2021141066)

Docentes

Jorge Henriques

Paulo Soares

Elaborado em

05/2024

ÍNDICE

COIMBRA BUSINESS SCHOOL
iscac.pt

Índice de figuras

1 Introdução

Este projeto foi desenvolvido no âmbito da unidade curricular de Projeto e Desenvolvimento Informático da licenciatura em Informática de Gestão do Instituto Superior de Contabilidade e Administração de Coimbra.

O presente relatório relata a realização de vários tipos de testes a um website que consiste num Marketplace destinado à prestação de serviços freelancer na área da tecnologia.

Este relatório visa comprovar que o projeto DevLancer está devidamente testado e que as funcionalidades críticas estão a funcionar corretamente. Além disso, vem garantir a segurança dos dados que passam pela plataforma. Inclui uma visão da metodologia adotada para os testes, os testes executados e o seu resultado, assim como possíveis melhorias a implementar para aprimorar o projeto.

2 Metodologia

De modo a garantir a qualidade e a robustez do website DevLancer, realizámos três tipos de testes: testes funcionais, testes de integração e testes de segurança. Cada um destes tipos de teste foi selecionado para abranger diferentes aspetos do sistema:

- Com recurso aos testes funcionais verificámos se as funcionalidades principais do website funcionam conforme esperado. Nomeadamente: adição de serviços à plataforma; realização de pedidos de serviço; criação de pagamentos; autenticação e filtração de serviços.

- Complementarmente, realizámos um teste de integração do DevLancer com o sistema PayPal que garante que os clientes podem realizar os pagamentos dos serviços, ponto fulcral ao funcionamento da plataforma.

- Por fim, e de forma a garantir a segurança dos utilizadores que venham a utilizar o DevLancer, realizámos testes de segurança para assegurar que os dados e as transações estão protegidos contra possíveis ameaças.

3 Testes funcionais e de integração

Objetivo: Verificar se as principais funcionalidades do website funcionam conforme esperado e se o PayPal está corretamente integrado.

Ferramenta selecionada:

- pytest-django: Ferramenta que permite escrever e executar testes automatizados, tanto funcionais como de integração, especificamente para aplicações Django.

Cenários de Teste:

1. **Teste funcional para adição de serviços:**

Objetivo: Verificar se os prestadores de serviços podem adicionar novos serviços à plataforma corretamente.

2. **Teste funcional para efetuar pedido de serviço:**

Objetivo: Verificar se os clientes podem efetuar pedidos de serviços corretamente em relação a serviços existentes na base de dados

3. **Teste funcional e de integração para criação de pagamento com PayPal:**

Objetivo: Verificar se os clientes podem criar pagamentos para os serviços na plataforma (funcional) e se o redireccionamento para a página de aprovação do PayPal funciona corretamente (integração).

4. **Teste funcional de autenticação:**

Objetivo: Testar se os dois tipos de utilizadores registados (clientes e prestadores) podem efetuar o login corretamente. Este teste está incluído na automatização dos três testes descritos acima, dado que é necessário efetuar o login como cliente ou prestador (dependendo das permissões necessárias) para ter acesso às funcionalidades em questão.

5. **Teste funcional para filtragem de serviços:**

Objetivo: Verificar se os utilizadores podem filtrar os serviços corretamente com base em categoria, preço e tecnologias utilizadas.

O código para a automatização dos testes pode ser consultado no repositório do github destinado a este projeto, no diretório devlancer/test/test_funcional.py.

Resultado dos Testes:

Todos os testes mencionados passaram, garantindo a correta implementação das funcionalidades e ligação ao API do PayPal. O resultado dos testes com recurso à ferramenta pytest-django pode ser consultado na Figura 1.

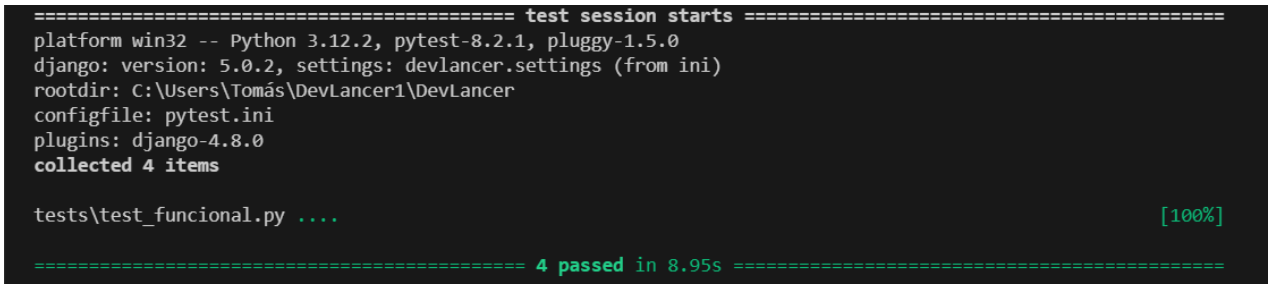

Figura 1 - Resultados dos testes funcionais com recurso à ferramenta pytest-django

4 Testes de segurança

Objetivo: Garantir que os dados dos utilizadores e as transações estão protegidos contra ameaças.

Ferramentas selecionadas:

- Django security check: Realiza uma verificação das configurações de segurança do projeto para identificar potenciais vulnerabilidades e assegurar as melhores práticas de segurança.
- OWASP ZAP: Ferramenta de análise de segurança utilizada para realizar testes de penetração e identificar vulnerabilidades no sistema.

1. Django Security check

Resultado:

```
WARNINGS:
?: (security.W004) You have not set a value for the SECURE_HSTS_SECONDS settin
g. If your entire site is served only over SSL, you may want to consider setti
ng a value and enabling HTTP Strict Transport Security. Be sure to read the do
cumentation first; enabling HSTS carelessly can cause serious, irreversible pr
oblems.
?: (security.W008) Your SECURE_SSL_REDIRECT setting is not set to True. Unless
 your site should be available over both SSL and non-SSL connections, you may
want to either set this setting True or configure a load balancer or reverse-p
roxy server to redirect all connections to HTTPS.
?: (security.W009) Your SECRET_KEY has less than 50 characters, less than 5 un
ique characters, or it's prefixed with 'django-insecure-' indicating that it w
as generated automatically by Django. Please generate a long and random value,
 otherwise many of Django's security-critical features will be vulnerable to a
ttack.
?: (security.W012) SESSION_COOKIE_SECURE is not set to True. Using a secure-on
ly session cookie makes it more difficult for network traffic sniffers to hija
ck user sessions.
only CSRF cookie makes it more difficult for network traffic sniffers to steal
?: (security.W018) You should not have DEBUG set to True in deployment.
?: (security.W020) ALLOWED_HOSTS must not be empty in deployment.

System check identified 7 issues (0 silenced).
```

Figura 2 - Resultado do teste de segurança com recurso à ferramenta django security check

De acordo com o resultado da Figura 2, a configuração para a fase de implantação deve ser a descrita na Figura 3:

```
SECURE_HSTS_SECONDS = 2592000  #(30 dias) intervalo de tempo em que os
navegadores devem lembrar os utilizadores de usar HTTPS
SECURE_HSTS_INCLUDE_SUBDOMAINS = True  # obriga a que os subdomínios
sejam acedidos apenas via HTTPS
SECURE_HSTS_PRELOAD = True  # garantir que todos os navegadores usam
HTTPS mesmo na 1ª visita ao site, submetendo-o para a lista de preload de
HSTS dos navegadores
SECURE_SSL_REDIRECT = True #redireciona automaticamente todas as
requisições HTTP para HTTPS
SESSION_COOKIE_SECURE = True #assegura que os cookies de sessão sejam
enviados apenas em HTTPS
CSRF_COOKIE_SECURE = True  # Garante que os cookies CSRF só são enviados
por HTTPS
```

Figura 3 - Configurações a implementar na fase de implantação

Considerações:

- O servidor de desenvolvimento do Django utiliza HTTP e no ambiente de desenvolvimento local é suficiente esta abordagem, pelo que não iremos implementar para já estas configurações. No entanto estas alterações devem ser aplicadas na fase da implantação para garantir a segurança do website.
- Da mesma forma, o "DEBUG = True" deve ser alterado para "False" nessa mesma fase, no entanto, ainda é necessário que se mantenha como "True".

2. **OWASP ZAP:**

Cenários de Teste:

1. **Proteção CSRF** (Cross-Site Request Forgery): Verificar a implementação de tokens CSRF em todos os formulários.

2. **Proteção XSS** (Cross-Site Scripting): Testar a proteção contra ataques XSS.

3. **Autenticação e Autorização:** Verificar se o sistema de autenticação e autorização do Django está corretamente configurado.

4. **Encriptação de Dados Sensíveis:** Testar a encriptação de palavras-passe e a segurança das transações com PayPal.

Resultados:

1. Proteção CSRF:

O sistema aparenta estar protegido contra CSRF, dado que não houve alertas específicos que indicassem a ausência de tokens CSRF nos formulários. Também verificámos manualmente a presença de tokens CSRF nos formulários para confirmar essa conclusão.

2. Proteção XSS:

Foram encontrados potenciais pontos de XSS através da vulnerabilidade "User Controllable HTML Element Attribute (Potential XSS)". Há pontos na aplicação onde entradas controladas pelo utilizador podem levar a ataques XSS.

Melhoria: Sanitização das entradas controladas pelo utilizador com a biblioteca “bleach”. No entanto, o Django já oferece proteção contra a maioria dos ataques XSS, conforme é explicado na documentação do mesmo, fornecendo uma boa mitigação para este tipo de ataques. Concluímos que poderá ser uma consideração a ter no futuro.

3. Autenticação e Autorização:

O sistema de autenticação foi detetado, apesar da configuração exata não ter sido verificada pela ferramenta ZAP. Verifica-se no painel do admin do Django que as permissões de acesso estão corretamente implementadas, garantindo que apenas utilizadores autorizados podem aceder a recursos protegidos.

Melhoria: Proteger contra-ataques de força bruta ao implementar limitação de tentativas de login usando django-axes. Considerámos esta melhoria importante, pelo que a adicionámos ao nosso projeto. Configurámos, entre outras definições, um limite de cinco tentativas de login com informações incorretas e uma página para redirecionar o utilizador quando isto sucede.

4. Encriptação de Dados Sensíveis:

A encriptação de palavras-passe e a segurança das transações com PayPal não foram verificadas diretamente pelo ZAP, pelo que iremos realizar testes adicionais.

Ferramenta adicional: pytest-django.

a) **Verificar o algoritmo de encriptação das palavras-passe:**

   Realizámos um teste para confirmar se, quando um utilizador é criado, a sua palavra-passe é armazenada com o algoritmo de hash esperado, de forma a garantir a segurança das palavras-passe dos utilizadores.

b) Verificar a certificação SSL da API do PayPal:

Realizámos um teste de segurança que verifica vários pontos de forma a garantir a segurança na comunicação entre a plataforma e o PayPal:

- Verifica se a requisição feita ao API do Paypal é feita com HTTPS e é bem-sucedida (reforça a conclusão do teste de integração).
- Verifica se o certificado SSL existe e é válido.

O código para a automatização dos testes pode ser consultado no repositório do github deste projeto, no diretório devlancer/test/test_segurança.py.

Resultado: Ambos os testes passaram. Confirma-se que as palavras-passe são encriptadas com o algoritmo PBKDF2, padrão do Django. Para além disso, podemos também afirmar que há segurança na comunicação com o PayPal. O resultado dos testes com recurso à ferramenta pytest-django pode ser consultado na Figura 4.

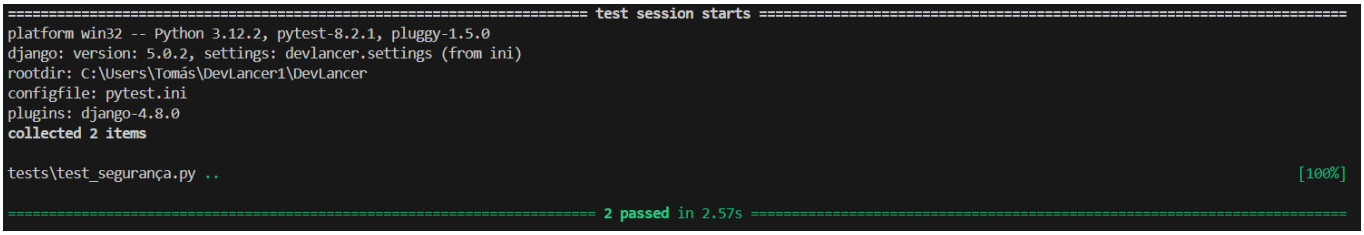

Figura 4 – Resultado dos testes de segurança com recurso à ferramenta pytest-django

Link repositório github: https://github.com/tomas02f/DevLancer

5 Conclusão

Consideramos que a realização de testes ao website DevLancer revelou-se uma mais-valia, dado que indicou possibilidades de melhoria no projeto. Os resultados dos testes funcionais e de integração foram altamente favoráveis, demonstrando que todas as funcionalidades foram implementadas corretamente, inclusive a integração com o PayPal. Em relação à segurança, o resultado foi positivo, mas ainda pode ser aprimorado. Assim sendo, implementámos uma das possíveis melhorias identificadas, estando as restantes em consideração futura.

Concluímos que a metodologia que definimos garantiu uma abordagem abrangente e detalhada para a validação do website DevLancer, assegurando que todas as funcionalidades essenciais estão operacionais e que a plataforma é segura.

6 Referências

pytest-django. (2024). Documentação do pytest-django. Disponível em https://pytest-django.readthedocs.io/en/latest/

Django. (2024). Segurança no Django. Disponível em https://docs.djangoproject.com/en/5.0/topics/security/

Django. (n.d.). Gestão de senhas. Disponível em https://docs.djangoproject.com/en/5.0/topics/auth/passwords/

OWASP ZAP. (n.d.). Início Rápido. Disponível em https://www.zaproxy.org/docs/desktop/addons/quick-start/

PayPal Developer. (n.d.). API de Pagamentos. Disponível em https://developer.paypal.com/docs/api/payments/v1/

Django-Axes. (n.d.). Documentação do Django-Axes. Disponível em https://django-axes.readthedocs.io/en/latest/