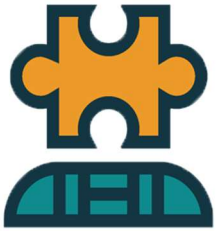


AutMe

Licenciatura em Tecnologias e Sistemas de Informação Para a Web

Documentação dos testes desenvolvidos

UC: Testes e Performance Web

Orientação: Prof. João Ferreira

Trabalho desenvolvido por:

Luís Miguel Pêgas Gomes – nº 40200283

Pedro Henrique Ferreira da Silva – nº 40200241

Vila do Conde, junho de 2022

Índice

1. Avaliação da performance 3

1.1 – Descrição dos testes realizados e avaliação da performance 3

1.2 – Plano de melhoria de performance 3

1.3 – Descrição da implementação do plano de melhoria de performance ao protótipo funcional e plano de testes a implementar 4

2. Plano de testes a implementar 6

2.1 Requisitos a serem testados 6

2.2 Estratégias e ferramentas de teste 6

2.3 Equipa e infraestrutura 6

2.4 Cronograma de atividades 7

2.5 Critérios de entrada: 7

2.6 Critérios de saída: 7

3. Descrição da implementação do plano de testes 8

3.1 Testes unitários 8

3.2 Testes de integração à API 9

3.3 Testes de integração à Base de Dados 11

3.5 Testes de usabilidade 13

5. Links 15

5.1 Link do repositório do código do *front-end* 15

5.2 Link do repositório do código do *back-end* 15

5.3 Link do deploy da api 15

5.4 Link do repositório do código de todos os testes realizados 15

5.5 Link da aplicação web funcional 15

6. Credenciais de acesso à aplicação 15

1. Avaliação da performance

1.1 – Descrição dos testes realizados e avaliação da performance

Durante o desenvolvimento do projeto fomos testando a aplicação de modo em encontrar possíveis erros e caso encontrados foram imediatamente corrigidos. Na fase final do desenvolvimento da aplicação, portanto quando demos por terminada a nossa aplicação, escolhemos um dia para a aplicação ser sujeita a novos testes, assim podemos detetar todos os erros que esta pudesse eventualmente ter e termos a certeza de que todos os erros seriam solucionados testamos a aplicação ao ínfimo pormenor. Para isso tivemos que verificar todos os requisitos, um a um, e solucionar possíveis falhas.

Para a validação de utilizadores a mudança das informações pessoais dos utilizadores, as anotações do psicólogo e marcação de consultas utilizamos a estratégia da caixa preta em que os testes são baseados na especificação de requisitos da aplicação.

Quanto à avaliação da performance do protótipo funcional, temos capturas de ecrã de cada página (já entregues na primeira entrega de webpii) que demonstram a avaliação e o diagnóstico de cada uma, que foi obtido através da ferramenta Page Speed Insights e da ferramenta Google Lighthouse. Através destes dados e do diagnóstico apresentado, foi-nos possível determinar um planeamento da melhoria de performance da nossa aplicação.

1.2 – Plano de melhoria de performance

Após a avaliação da performance das várias páginas que dispomos na nossa aplicação e consultando o diagnóstico que nos era dado pela plataforma de testes *Page Speed Insights,* podemos perceber que esse mesmo diagnóstico era semelhante entre todas elas, por esse mesmo motivo delineamos os parâmetros que havíamos de melhorar de uma forma geral, de modo a fazer um plano que tem como objetivo principal melhorar a performance da nossa aplicação, os parâmetros são os seguintes:

- ✓ O texto deve aparecer visível durante o carregamento.

- ✓ Definir uma largura e uma altura explícitas nos elementos de imagem para reduzir mudanças de esquema de modo a melhorar o *Cumulative Layout Shift.*
- ✓ Listar as tarefas mais longas na "thread" principal, o que é útil para identificar o que mais contribui para o atraso de entrada.
- ✓ Corrigir mudanças de esquemas grandes, pois estes elementos "DOM" são os que mais contribuem para o *Cumulative Layout Shift* da página.
- ✓ Adicionar um ficheiro que defina valores para a quantidade e tamanho dos recursos da página.
- ✓ Reduzir o tamanho das cadeias de pedidos críticos que apresentam os recursos que são carregados com uma prioridade elevada, reduzir o tamanho de transferência dos recursos ou adiar a transferência de recursos desnecessários para melhorar o carregamento de página.
- ✓ Manter a contagem dos pedidos baixa e os tamanhos de transferência pequenos.
- ✓ Colocar botões com nome acessível, nomeadamente o botão de *logout* da página.
- ✓ Colocar atributos (alt) nos elementos das imagens, pois os elementos informativos devem procurar incluir texto curto, descritivo e alternativo.
- ✓ Erros de navegador registados na consola.

1.3 – Descrição da implementação do plano de melhoria de performance ao protótipo funcional e plano de testes a implementar

Após termos definido o plano de melhoria de performance, passamos para a fase de implementação da melhoria de performance do nosso projeto. Através dos testes que desenvolvemos percebemos o nosso maior problema eram as imagens, quer seja o tamanho das imagens, quer seja o formato das mesmas. Convertemos as imagens para o formato "webp" e ajustamos as dimensões destas. Outro componente que implementámos foi "loading="lazy"" este componente faz com que as imagens só sejam carregadas na página à medida em que em é feito o scroll na página.

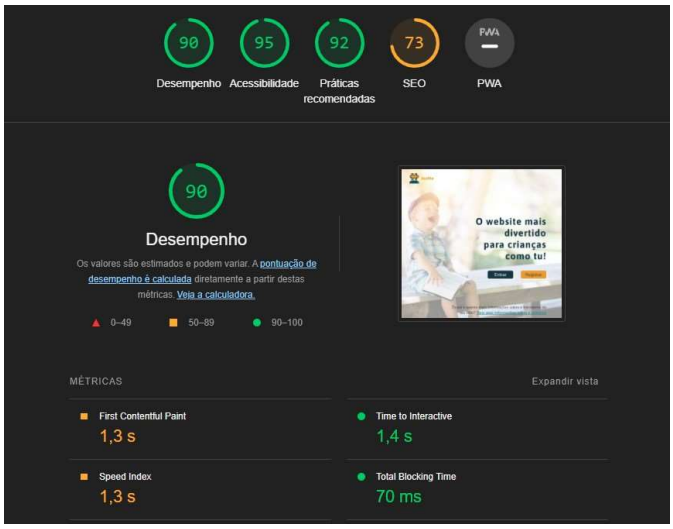
Utilizamos a técnica de carregar os ficheiros do tipo *Javascript* no fim da página, pois quando o navegador encontrava uma invocação ao ficheiro *javascript*, ele interrompia todas as outras operações enquanto analisa o código fazendo assim com que o carregamento da página seja mais lento. Esta técnica faz com que melhore o desempenho da página, pois o conteúdo pode ser visto mesmo antes que seja feita uma tentativa de processar *Javascript*.

Ainda relativamente à invocação do ficheiro *JS*, para garantir que o mesmo seja carregado sem bloquear o processo de renderização, utilizamos o atributo "defer" que faz com que todos os ficheiros do tipo *JS* adiados, sejam executados na ordem em que são mencionados na página.

Já quanto aos ficheiros css e javascript, tivemos de os minificar, ou seja remover os comentários, espaços em branco e carateres, removemos também o css que já não era necessário de modo a reduzir o tamanho dos ficheiros.

Para tornarmos a nossa aplicação web numa "PWA" (Progressive Web App) que obedece aos seguintes parâmetros, responsividade, interativa tal como uma aplicação de raiz, sempre atualizada e segura, tivemos de utilizar "Service Workers" (Permite que sejam aproveitadas as solicitações de rede e construir melhores experiências na web) para intercetar e armazenar em cache quaisquer solicitações de rede.

De seguida, só nos faltava fazer novamente os testes à nossa aplicação web, e podemos perceber através da ferramenta "PageSpeed Insights" claramente que a performance da mesma melhorou através das alterações que fizemos os valores das métricas dados foram positivos tal como a rápida resposta da página. As imagens abaixo representadas refletem a evolução da página principal da aplicação, antes e depois (respetivamente) da implementação da melhoria da performance.



2. Plano de testes a implementar

Relativamente ao plano de testes que implementamos, este consistia em definir os objetivos do teste e a abordagem para cumprir esses mesmo objetivos dentro de todas as restrições do projeto. Esta fase passa por decidir as técnicas de teste que precisam de ser executadas. Os objetivos deste plano de testes são:

- Avaliar produtos de trabalho como requisitos, histórias de utilizadores, design e código;
- Verificar se todos os requisitos especificados foram atendidos;
- Validar se o objeto de teste está completo e funciona conforme o esperado;
- Construir confiança no nível de qualidade do objeto de teste;
- Reduzir o nível de risco de qualidade de software inadequado;
- Encontrar falhas e defeitos;
- Prevenir defeitos através da identificação de defeitos nas especificações dos requisitos, que são removidos antes que causem problemas nas especificações do projeto.

2.1 Requisitos a serem testados

Através destes testes serão testados os vários requisitos que definimos para a nossa aplicação, entre os quais o desempenho, a segurança, a interface, os jogos das crianças, as emoções, ou seja, todas as principais funcionalidades da nossa aplicação.

2.2 Estratégias e ferramentas de teste

Os testes que irão ser executados através deste plano de testes são os testes de integração, os testes de componente(unitários) e os testes de usabilidade. Futuramente descreveremos aqui as ferramentas que utilizamos para executar os referidos testes.

2.3 Equipa e infraestrutura

Test manager: Luís Gomes

Tester: Pedro Silva

2.4 Cronograma de atividades

- Testes de usabilidade
  - ✓ Fazer de 16-05-2022 a 23-05-2022
- Testes de componente (unitários)
  - ✓ Fazer de 23-05-2022 a 30-05-2022
- Testes de integração
  - ✓ Fazer de 30-05-2022 a 06-06-2022

2.5 Critérios de entrada:

- Disponibilidade de requisitos testáveis

2.6 Critérios de saída:

- 40 testes

3. Descrição da implementação do plano de testes

Após termos definido o plano de implementação dos testes, seja de usabilidade, seja de integração, quer sejam testes unitários, passamos para a fase de implementação dos testes relativos ao nosso projeto. Através dos testes que implementamos percebemos que havia alterações cruciais a fazer a certas funções da "API" e do "VUE", principalmente nos jogos.

3.1 Testes unitários

Os testes unitários são testes automatizados que verificam uma pequena porção de código, que são executados de uma forma rápida e de forma isolada.

De modo a testarmos funções isoladas que foram utilizadas no front-end da aplicação web, procedemos à resolução dos testes unitários às mesmas. O objetivo da realização destes testes passa por permitir o crescimento sustentável do projeto.

As funções testadas são as seguintes:

- *Valid password*
  - ✓ Confirma se as palavra-passes inseridas correspondem uma à outra.
- *Verify lifes*
  - ✓ Verifica o número de vidas do jogo "CompletaMe", para devolver a classificação da criança no jogo.
- *Check*
  - ✓ Verifica o número de respostas certas do jogo, para devolver a classificação do mesmo, verifica se é maior e menor que dois e verifica também caso haja erro.
- Check Question
  - ✓ Verifica se a resposta selecionada do jogo está correta, pode devolver resposta correta ou incorreta, caso contrário devolve um erro.

3.2 Testes de integração à API

Os testes de integração desempenham um papel importante no seu conjunto de testes, estes tipos de testes quase sempre verificam como o seu sistema trabalha em integração com dependências fora do processo. Esses testes que cobrem o quadrante de controladores podem às vezes ser testes de unidade. Se todas as dependências fora de processo forem substituídas por simulações, não haverá dependências compartilhadas entre testes;

As funções testadas são as seguintes:

- *Create account*
  - ✓ Cria a conta e verifica se o estado da resposta da API é verdadeiro.

- *Login*
  - ✓ Verifica se todos os parâmetros da resposta de login são os correspondentes e guarda o token da para aceder a outras rotas da aplicação.

- *Login – Error 400*
  - ✓ Verifica se a resposta devolveu o erro 400, devido a não enviar a password, devolve um erro.

- *Get user profile*
  - ✓ Verifica se os dados enviados são os correspondentes ao utilizador que foi criado no primeiro teste.

- *Get user profile – Error 403*
  - ✓ Devolve o erro de que não tem permissão para aceder à rota pois o username não corresponde ao token que foi enviado.

User not authenticated

- ✓ Devolve o erro que o utilizador não tem permissões para aceder à referida rota pois não é enviado o token correspondente.

Update password

- ✓ Faz o teste de atualização da password do utilizador que é passado como parâmetro.

Create diary

- ✓ Cria um diário que é passado como parâmetro, guarda o id desse diário para ser possível apagá-lo.

Delete diary

- ✓ Apaga o diário criado pelo teste anterior, verifica se os dados da resposta e a mensagem estão corretos.

Delete diary – Not found

- ✓ Verifica se a mensagem de erro e a resposta estão corretas, para se o utilizador tentar apagar um diário não existente.

3.3 Testes de integração à Base de Dados

Os testes de integração à base de dados consistem nas dependências que são geridas fora do processo de testes, o exemplo mais comum de disso mesmo é uma base de dados que nenhum outro cliente tem acesso. A execução destes testes numa base de dados real oferece proteção à prova de balas contra regressões.

As funções testadas são as seguintes:

- *Get psychologists from database*
  - ✓ Verifica se existem psicólogos registados na base de dados.

- *Create user to database*
  - ✓ Cria um novo utilizador na base de dados e verifica se não existem possíveis erros.

- *Update user*
  - ✓ Atualiza o utilizador criado, e verifica se o resultado foi verdadeiro para eliminar a existência de erros.

- *Delete user*
  - ✓ Apaga um utilizador da base de dados e verifica se foi apagado com sucesso.

- *Get diaries from database*
  - ✓ Verifica se existem diários registados na base de dados.

- *Delete diary*
  - ✓ Apaga um diário da base de dados e verifica se foi apagado com sucesso.

- *Get images from database*
  - ✓ Verifica se existem imagens da tabela “images” registados na base de dados.

Post images to database

- ✓ Cria uma nova imagem na base de dados e verifica se não existem possíveis erros.

Delete images

- ✓ Apaga uma imagem da propriedade “images” da base de dados e verifica se foi apagado com sucesso.

3.5 Testes de usabilidade

Através dos testes de usabilidade podemos ver o que os utilizadores fazem, o que aparece para elas e ver também aquilo que não funciona. Quando o teste de usabilidade faz parte do design e desenvolvimento, o conhecimento que obtemos sobre o a nossa aplicação, a experiência dos utilizadores suporta todos os aspetos de design e desenvolvimento. Para a realização destes testes fizemos uso da ferramenta *selenium* que é uma estrutura de teste automatizada gratuita usada para validar aplicações web em diferentes navegadores e plataformas.

- *Login*
  - ✓ Executa todo o processo do login, envia o username e a password e clica no botão para iniciar sessão.
- *Register*
  - ✓ Envia todos os dados que o utilizador inseriu para o registar na aplicação.
- *Logout*
  - ✓ Após executar todo o processo de login clica no botão de terminar sessão localizado na navbar da aplicação.
- *Achievements*
  - ✓ Clica na opção “Conquistas” localizada na navbar da aplicação, e de seguida vê todas as conquistas existentes.
- *Diary*
  - ✓ Clica na opção de ver os diários e depois insere um novo diário na aplicação.
- *Children*
  - ✓ Clica na opção de ver crianças para ver as crianças que o referido utilizador está conectado, após isso clica em “Ver ficha Técnica”.
- *Completa*
  - ✓ Simula todo o processo do jogo de reconhecimento de emoções.

Profile

- ✓ Consulta os dados do utilizador que está logado, e altera a palavra-passe do mesmo.

PsychologistList

- ✓ Clica na opção de Psicólogos e de seguida simula o processo de marcação de uma consulta para a criança selecionada.

Question

- ✓ Consulta a página de informações existente na aplicação.

5. Links

5.1 Link do repositório do código do *front-end*

https://github.com/luispegasgomes/autme

5.2 Link do repositório do código do *back-end*

https://github.com/luispegasgomes/newAutmeApi

5.3 Link do deploy da api

https://api-autme-new-nodejs.herokuapp.com/

5.4 Link do repositório do código de todos os testes realizados

https://github.com/luispegasgomes/autmeTests

5.5 Link da aplicação web funcional

https://autme-luispegasgomes.vercel.app/

6. Credenciais de acesso à aplicação

Tipo de Utilizador	Username	Password
Criança	child	Esmad_2122
Tutor	tutor	Esmad_2122
Psicólogo	psychologist	Esmad_2122