

Relatório do Trabalho Prático

Segurança em Aplicações Web

CTeSP - Desenvolvimento para a WEB e Dispositivos Móveis

Alunos: Célio Ighour de Castro Rodrigues (18180016)
Simona Tocitu (18180008)

16 de Janeiro de 2019

Sumário

1. Objetivo	3
2. Especificação da Aplicação	3
3. Arquitetura da Aplicação	3
4. Tecnologias Associadas	4
5. Demonstração Visual	5
6. Descrição da Aplicação Backend (PHP)	7
6.1. Gestor de Dependências PHP	7
6.2. Arquitetura MVC	7
6.3. Padrão REST	8
6.4. Armazenamento de Arquivos	8
6.5. JSON Web Token (JWT)	8
6.6. Middlewares	9
6.7. Dados de Arranque (seeds)	9
6.8. Sanitização	9
6.9. Validação	10
6.10. Registro	10
6.11. Passwords	10
6.12. Tratamento de Exceções	10
6.13. Modelagem das Entidades e Relacionamentos	11
7. Descrição da Aplicação Frontend (Javascript)	11
7.1. Arquitetura e Padrões	11
7.2. Cliente	11
7.3. Progressive Web App (PWA)	11
7.4. Validação	12
8. Referências	12

1. Objetivo

O objetivo geral do trabalho consiste na elaboração de uma aplicação Web em PHP que permita listar os itens que determinados utilizadores pretendem vender.

Dentre as funcionalidades da aplicação, inclui-se:

- a) Registro de utilizadores;
- b) Recuperação de conta (*Forgot Me*);
- c) Manter autenticação local (*Remember Me*).
- d) Login de usuário;
- e) Área de acesso público, onde serão listados todos os itens vendidos;
- f) Área de acesso a usuários autenticados:
 - i) Perfil do utilizador, com a possibilidade de alterar seus dados;
 - ii) Lista de itens vendidos do utilizador, com a possibilidade de adicionar, editar ou remover itens;
- g) Área de acesso a usuários administrativos:
 - i) Lista de usuários existentes;
- h) Possibilidade de vinculação de imagem a um utilizador e a um item vendido;
- i) Dados de arranque para teste da aplicação (*seed*), com informações verossímeis.

Tal aplicação deverá seguir diretrizes de segurança para o seu funcionamento, tais como sanitização e validação de *inputs*, *hashing* de *passwords*, tratamento de exceções, dentre outras.

Outrossim, deverão ser acrescentadas outras funcionalidades pertinentes e a especificação dos objetivos com um tema específico.

2. Especificação da Aplicação

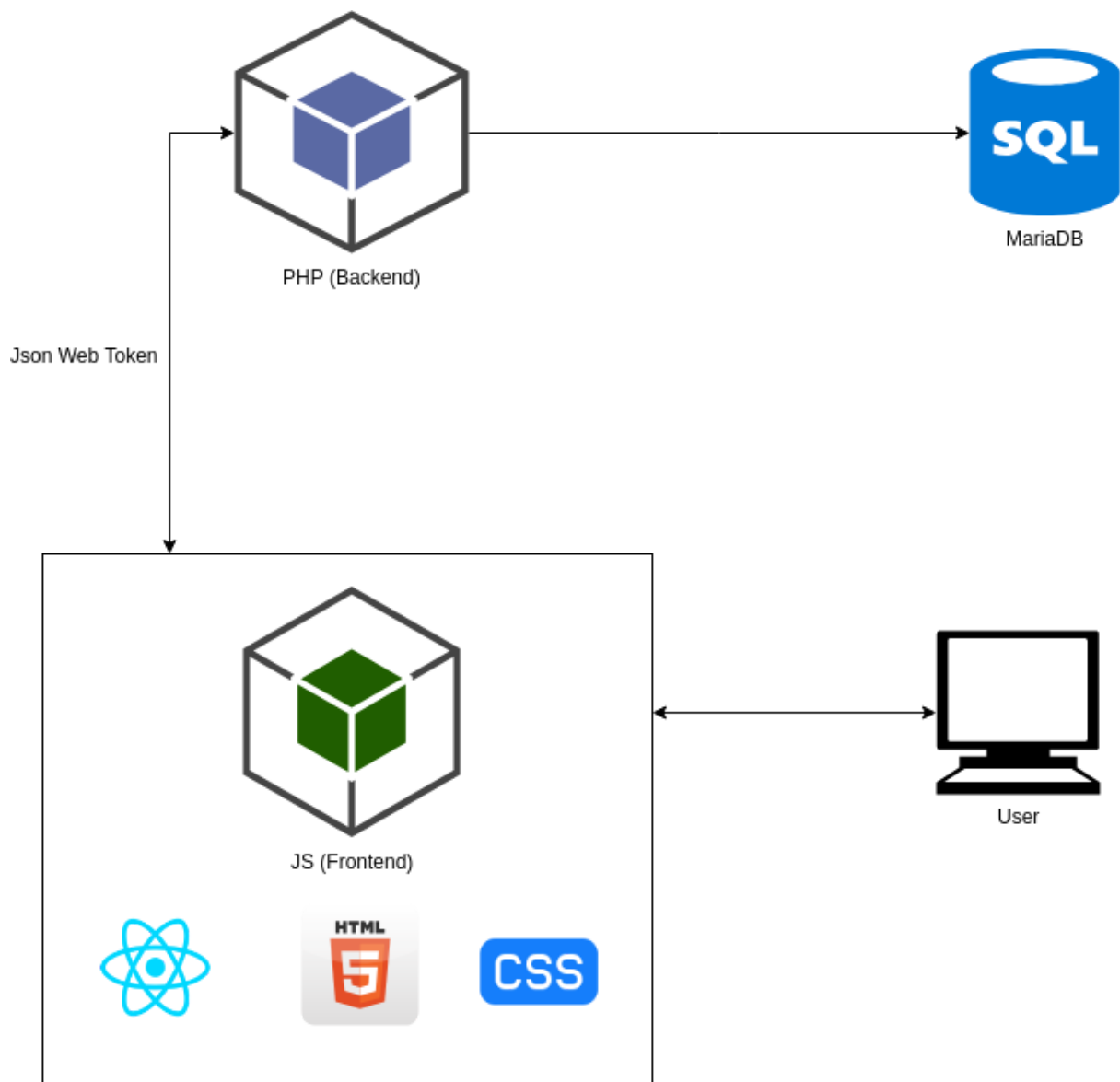
In casu, a aplicação a ser desenvolvida abordará o tema de jogos para *videogames*, consubstanciando em uma *Games Store*.

Portanto, os usuários da aplicação irão gerenciar jogos de *videogames* a serem disponibilizados na plataforma como itens para venda, com a possibilidade de os gerir (incluindo com uma imagem do jogo em questão).

3. Arquitetura da Aplicação

A arquitetura base da aplicação vai dividi-la em duas aplicações distintas:

- a) Aplicação *web service* construída em PHP, servindo como API (*backend*);
- b) Aplicação cliente (*frontend*) construída em JavaScript (ReactJS), consumindo os serviços do *web service*. Funciona como um *Progressive Web App*, permitindo o seu uso como Desktop, Web ou Mobile.



Dessa forma, o usuário final primeiramente obterá a aplicação *frontend*, por meio do qual irá realizar *requests* seguindo o protocolo HTTP para a aplicação *backend*, obtendo-se as devidas respostas e renderizando tais resultados para o usuário.

4. Tecnologias Associadas

A aplicação como um todo é desenvolvida com o uso de:

- [Docker](#): cria contêineres para cada aplicação integrante (*backend*, *frontend*) e outros serviços utilizados (MariaDB e phpMyAdmin), integrando-os e conectando-os;
- [MariaDB](#): SQL *database* para armazenar os dados utilizados pela aplicação;
- [phpMyAdmin](#): Interface web intuitiva para acesso à base de dados.

No caso da aplicação responsável pelo *backend*, as tecnologias associadas são:

- [Composer](#): gestor de dependências em PHP;

- b) [Dotenv](#): componente do *Symfony 4* para o uso de variáveis de ambiente;
- c) [PHP](#): linguagem de programação utilizada;
- d) [PHPMailer](#): *library* para o envio de emails com PHP;
- e) [Php Jwt](#): *library* para encriptar e descriptar JSON Web Tokens em PHP.

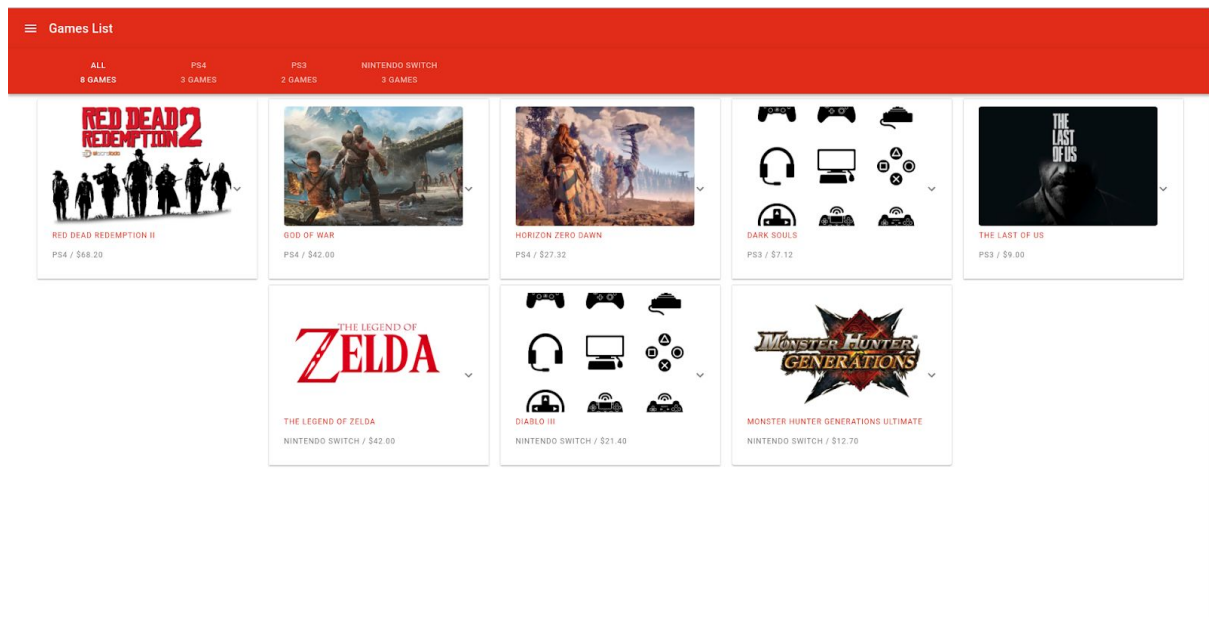
Outrossim, a aplicação *frontend* utiliza as seguintes tecnologias associadas:

- a) [Axios](#): cliente HTTP baseado em *Promises* em Javascript;
- b) [Javascript](#): linguagem de programação;
- c) [Jwt Decode](#): descripta JSON Web Tokens;
- d) [Material-UI](#): componentes ReactJS que implementam o *Material Design* da Google;
- e) [ReactJS](#): *library* em Javascript para a construção de interfaces do usuário;
- f) [React Router](#): roteador declarativo para o ReactJS.

5. Demonstração Visual

Após o desenvolvimento da aplicação, o aspecto visual da interação com usuário resultou nas imagens demonstrativas a seguir.

5.1. Visualização *web* da Lista de Jogos



5.2. Visualização *mobile* da Lista de Jogos

Games List

ALL


8 GAMES

PS4

3 GAMES


PS3

2 GAMES









RED DEAD REDEMPTION II

PS4 / \$68.20



5.3. Visualização *mobile* da Lista de Usuários

Admin: Users		
	ADMIN ADMIN@GMAIL.COM	▼
	USER1 USER1@GMAIL.COM	▼
	USER2 USER2@GMAIL.COM	▼
	USER3 USER3@GMAIL.COM	▼
	USER4 USER4@GMAIL.COM	▼
	USER5 USER5@GMAIL.COM	▼

6. Descrição da Aplicação *Backend* (PHP)

6.1. Gestor de Dependências PHP

Inicialmente, foi necessária a adoção de um gestor dependências (Composer) para facilitar a utilização das classes e funções PHP de autoria própria ou de terceiros.

6.2. Arquitetura MVC

Em seguida, implementou-se uma arquitetura *MVC* (*Model-View-Controller*) para facilitar a gestão dos recursos da aplicação. Dessa forma, a conversão de um *request* em um *response* passa por um processo que engloba:

- O servidor recebe o request para qualquer diretório e direciona-os para um arquivo de entrada (*index.php*), de acesso público.

- b) O *index.php* comunica-se com o restante da aplicação (de acesso não público), por meio de um ficheiro inicializador (*bootstrap.php*) e posteriormente inicializando o controlador principal (*FrontController.php*).
- c) O ficheiro *bootstrap.php* inicializa algumas definições do PHP, declara algumas constantes globais e carrega as variáveis de ambiente (*.env*).
- d) O controlador principal recebe a URL do *request* e detecta para qual controlador (e método) deve enviá-lo (com base em um número de rotas predefinidas), incluindo todos os parâmetros e arquivos anexos.
- e) Os controladores acessados comunicam-se com outras classes e modelos da aplicação, construindo uma resposta e devolvendo-a conforme o protocolo HTTP e no formato JSON.

A arquitetura MVC facilita a proibição de acesso a outros ficheiros da aplicação (senão o *index.php* ou outros ficheiros permitidos no diretório *public*), garantindo uma melhor segurança contra acesso não autorizado de arquivos da aplicação.

6.3. Padrão REST

Ademais o acesso da aplicação (como *web service*) utiliza-se do padrão REST (Representational State Transfer), para acesso dos recursos no servidor. Dessa forma, os recursos são acessados de forma *stateless* com uso de URLs padronizadas, por exemplo:

- a) Lista de usuários: GET */users*
- b) Dados do usuário de ID 1: GET */users/1*
- c) Alteração do usuário de ID 1: PUT */users/1/edit*
- d) Criação de um usuário: POST */users*
- e) Deleção de um usuário: DELETE */users/1*

O acesso à base de dados é realizado por meio de um *Data Access Object*, implementado com o PDO (*PHP Data Object*), protegendo o acesso à base de injeções SQL.

6.4. Armazenamento de Arquivos

Os arquivos (imagens) enviadas e recebidas são armazenados em diretório próprio em que apenas a aplicação pode escrever, mas todos podem ler (para possibilitar a visualização dessas imagens pelos usuários).

6.5. JSON Web Token (JWT)

A autenticação e autorização de acesso a determinadas rotas e recursos é possível por meio da utilização de um JSON Web Token (JWT). Após o *login* do usuário, é retornado uma mensagem de sucesso contendo o JWT. Caso a aplicação cliente (*frontend*) queira continuar suas requisições como tal usuário, é necessário enviar em seu *header Authorization* a informação do JWT, possibilitando o servidor a

detectar qual o usuário que faz a requisição (tendo em vista que é um serviço *stateless*).

O JWT inclui algumas informações importantes, tais como o ID do usuário, a data em que o token foi emitido e o *role* do usuário, sendo o token encriptado posteriormente com uma chave privada da aplicação.

Dessa forma, qualquer alteração ao token ou tentativa de criar um token fora da aplicação, resultará na invalidade do mesmo.

Outrossim, ao realizar o *logout*, o token é listado como bloqueado, não permitindo mais o seu uso.

E no caso de o usuário autenticar-se em outro ambiente (gerando um novo token), o token antigo é tornado inválido, pois ao realizar as requisições é checada a data de emissão do token recebido com a data do último token emitido para tal usuário (o único válido).

6.6. Middlewares

Corre na aplicação um mini-serviço para facilitar a realização de checagem da autenticação e autorização do usuário, a ser chamado em qualquer momento por um controlador.

Portanto, cada método do controlador pode definir se é necessário autenticação para o acesso do mesmo ou até mesmo que tipo de autorização o usuário precisa (e.g. administrador).

Da mesma forma, antes de redirecionar para qualquer controlador, é utilizado um mini-serviço de checagem de CORS (cross-origin resource sharing), para detectar se a origem pode acessar o servidor.

Assim, é possível definir explicitamente de onde podem vir as requisições para o *web service* (e.g. o endereço web do *frontend*).

6.7. Dados de Arranque (seeds)

Para facilitar o teste da aplicação, foram acrescentados dados de arranque da aplicação (*seeds*), com informação verossímil.

6.8. Sanitização

Após o redirecionamento para um controller, antes de qualquer coisa, é realizada a sanitização de todos os parâmetros do *request*, impedindo o seguimento de dados impróprios e, assim, reduzindo as chances de um *sql injection* ou *code injection*.

Outrossim, antes da inserção na base de dados, os parâmetros são sanitizados novamente pelo PDO.

6.9. Validação

Os parâmetros sanitizados são, posteriormente, validados. Dessa forma, é checado se o parâmetro recebido era do tipo e formato esperado pela aplicação, reduzindo as chances de armazenamento de dados inválidos.

6.10. Registro

No intuito de evitar a criação indevida de usuários, é exigido a confirmação de email após o registro, em que o usuário recebe um *link* de acesso à aplicação *frontend*, contendo um token gerado aleatoriamente.

Após a verificação do token, caso correto, o usuário é liberado para o *login* na aplicação.

6.11. Passwords

As senhas dos usuários exigem um grau de segurança médio na sua criação (quantidade de caracteres).

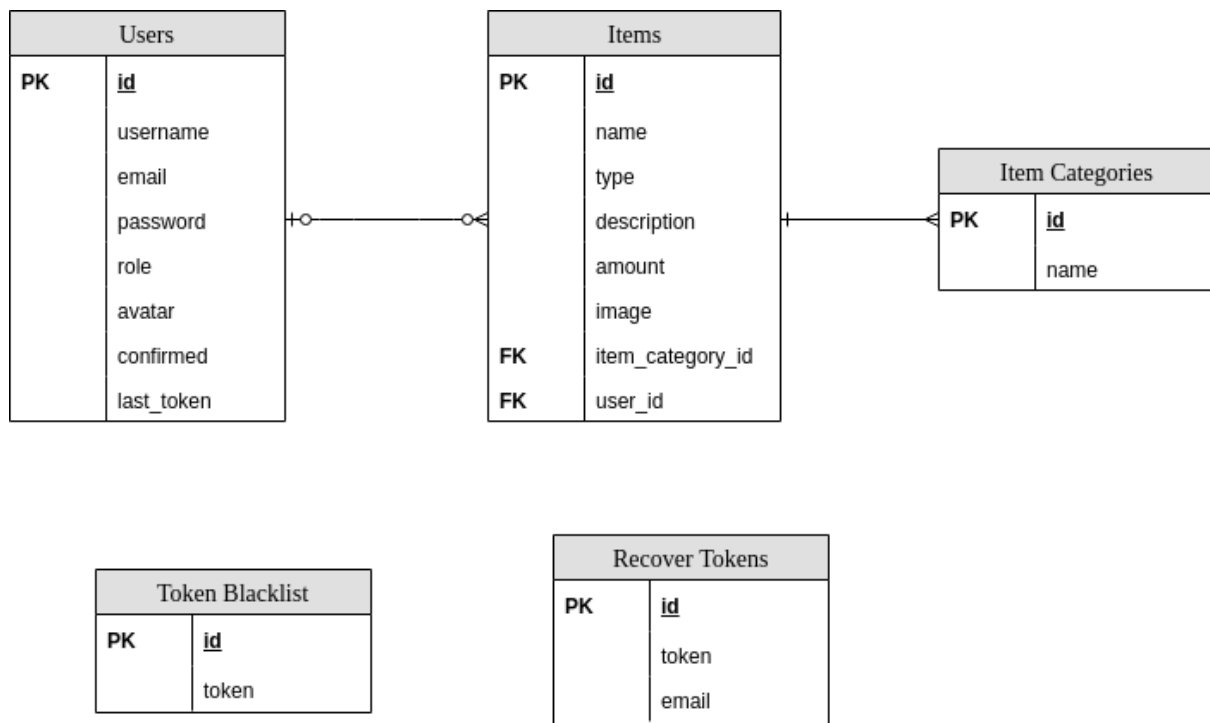
Outrossim, são encriptadas antes de serem inseridas na base de dados, com a utilização de um *salt* para reduzir o sucesso de *crackings*.

Ao realizar o *login*, a senha digitada pelo usuário é encriptada da mesma forma e comparada com o *hash* presente na base de dados, permitindo (ou não) o sucesso da autenticação.

6.12. Tratamento de Exceções

As exceções ocorridas no servidor são tratadas e encapsuladas em mensagens amigáveis em uma resposta JSON ao cliente.

6.13. Modelagem das Entidades e Relacionamentos



7. Descrição da Aplicação *Frontend* (Javascript)

7.1. Arquitetura e Padrões

A aplicação *frontend* foi desenvolvida para ser simplificada, utilizando-se de alguns padrões da comunidade ReactJS, como *Destructuring props*, *Function as children*, *High-order component*, *Controlled inputs* etc.

Outrossim, foi desenvolvida com a aplicação de conceitos da arquitetura *Flux*, realizando um fluxo unidirecional de dados entre os componentes, com o uso de uma *store* para armazenamento deles e *dispatchers* para a sua alteração. Tudo isso com utilização dos *contexts* do ReactJS.

7.2. Cliente

A aplicação é um cliente de um serviço web (*backend* em PHP), consumindo toda a informação necessária a partir do mesmo.

Dessa forma, ela apenas apresenta visualmente as informações recebidas do serviço web e cria funcionalidades para interação com os dados, enviando requisições de recolha, acréscimo, mudança ou retirada de dados no *backend*.

7.3. *Progressive Web App* (PWA)

A aplicação é um PWA, permitindo um uso confiável, rápido e engajador pelo usuário, seja em um ambiente *Desktop*, *Web* ou *Mobile*.

Dessa forma, o usuário poderá se utilizar da aplicação, por exemplo, em computadores desktops ou portáteis como uma aplicação nativa (caso a integração esteja disponível) ou uma aplicação *web*, a ser visualizada em *browsers* ou outros clientes de acesso à *web*.

Da mesma forma, é possível o uso em ambiente *mobile*, seja por meio de *browsers* ou com a possibilidade de instalação similar a uma nativa, quando suportados pelo sistema operacional.

7.4. Validação

Os *inputs* dos formulários são validados previamente na própria aplicação, evitando um *request* ao servidor quando já é sabido que tais dados são inválidos.

Ressalte-se que tal validação não retira a necessidade de validar novamente no servidor, pois a validação no *frontend* tem como intuito melhorar a usabilidade e experiência do usuário, não protegendo a aplicação como um todo do envio de dados inválidos.

8. Referências

Docker. Disponível em: <<https://www.docker.com/>>. Acesso em: 22 dez. 2018.

MariaDB. Disponível em: <<https://mariadb.org/>>. Acesso em: 22 dez. 2018.

PhpMyAdmin. Disponível em: <<https://www.phpmyadmin.net/>>. Acesso em: 22 dez. 2018.

Composer. Disponível em: <<https://getcomposer.org/>>. Acesso em: 22 dez. 2018.

Dotenv. Disponível em: <<https://symfony.com/doc/current/components/dotenv.html>>. Acesso em: 09 jan. 2019.

Php. Disponível em: <<http://php.net/>>. Acesso em: 22 dez. 2018.

Axios. Disponível em: <<https://github.com/axios/axios>>. Acesso em: 22 dez. 2018.

Javascript. Disponível em: <<https://developer.mozilla.org/pt-PT/docs/Web/JavaScript>>. Acesso em: 22 dez. 2018.

Php JWT. Disponível em: <<https://github.com/firebase/php-jwt>>. Acesso em: 09 jan. 2019.

JWT Decode. Disponível em: <<https://github.com/auth0/jwt-decode>>. Acesso em: 09 jan. 2019.

Material UI. Disponível em: <<https://material-ui.com/>>. Acesso em: 22 dez. 2018.

ReactJS. Disponível em: <<https://reactjs.org/>>. Acesso em: 22 dez. 2018.

React Router. Disponível em: <<https://github.com/ReactTraining/react-router>>. Acesso em: 22 dez. 2018.