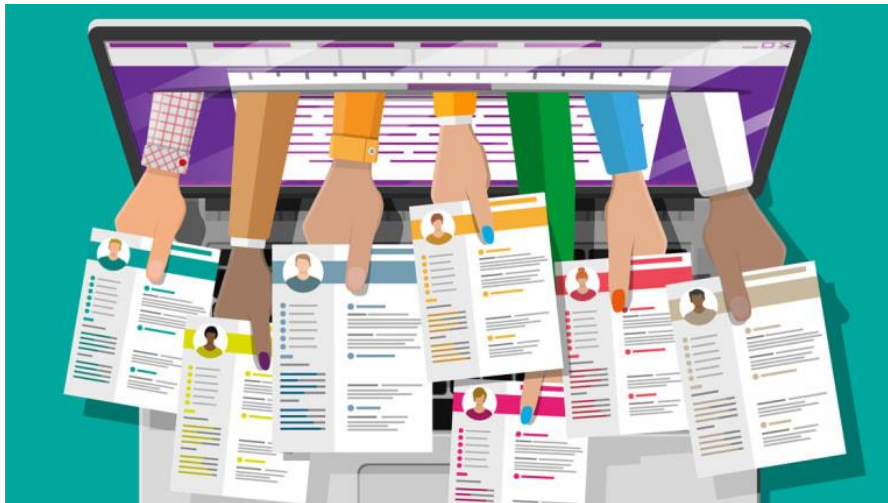


Programação Web 2022

Construção de rede social de portefólios



Projeto Final
Época Normal e de Recurso

Índice

1	Introdução.....	3
2	Tecnologia e metodologia a utilizar	3
2.1	Tecnologias	4
3	As ofertas de emprego.....	5
4	A rede de portefólios	5
5	Comparador de ofertas (Extra)	8
6	Sugestões	8
7	Regras no desenvolvimento do projeto.....	9
7.1	Regras de implementação e codificação	9
7.2	Constituição de grupos	9
7.3	Submissões do projeto.....	10
7.4	Estrutura de ficheiros e pastas do projeto.....	11
8	Avaliação	12
8.1	Regras de Avaliação	12
8.2	Critérios de Avaliação	12
9	Anexos.....	13
9.1	Sistema de Gestão da Base de Dados (SGBD).....	13
9.1.1	MySQL	13
9.2	Web Service	13
9.2.1	Abordagem RESTful.....	13
9.2.2	Comunicação entre a SPA e o serviço RESTful	14

1 Introdução

O principal objetivo deste projeto é desenvolver, utilizando HTML, CSS e JavaScript, uma solução Web que permita a criação de **um website**, ao estilo de uma rede social, com informação relativa ao portefólio dos seus utilizadores. **O website** deverá compreender as técnicas e seguir as boas práticas do desenvolvimento de código faladas nas aulas e deverá permitir aos utilizadores preencherem e visualizarem informações sobre o currículo profissional, assim como ofertas de emprego, com diferentes permissões de visualização.

Existirá também a possibilidade de ser desenvolvido um extra para o projecto que consiste na criação de um comparador de ofertas de emprego. Este extra deverá também recorrer ao uso das tecnologias já mencionadas (HTML, CSS e *JavaScript*) para cumprir o seu propósito.

A decisão pela escolha desta temática justifica-se pela crescente importância que as actuais redes sociais deste âmbito já têm no recrutamento e contratação de profissionais nas mais diversas áreas. A possibilidade de os utilizadores terem uma plataforma que lhes permita divulgar o seu portefólio e currículo entre a comunidade, é uma mais-valia que em muito auxilia o seu percurso profissional.

2 Tecnologia e metodologia a utilizar

Num primeiro momento, o principal foco do projeto deve ser o desenvolvimento de uma interface HTML/CSS que possibilite uma interação limpa e simples com as principais funcionalidades.

O número de páginas do website fica à escolha dos alunos, dependendo da idealização que fizerem para o problema na **fase de mockups**. Na criação do site devem seguir o material fornecido nas primeiras aulas para que as páginas sejam responsivas, tenham boa usabilidade e o aspeto visual seja claro e agradável.

Em termos de conteúdo o website deve ter pelo menos a seguinte informação:

- um menu que deve estar acessível em todas as páginas. No caso de sites com páginas longas é importante ter atenção para que não seja demasiado complicado para o utilizador aceder ao menu
- uma descrição inicial que descreve o objetivo e funcionalidades do site. Podem ser visitados outros websites para ter ideias sobre como organizar esta secção.
- uma secção que descreve a equipa (os alunos). A descrição deve ser feita com um formato profissional, e não como um projeto académico. Considere que o seu grupo criou uma startup com um projeto inovador e que nesta secção descreve a sua equipa (o grupo).
 - Poderá visitar alguns websites para inspiração. É importante ter em atenção que websites de empresas maiores podem não descrever a sua equipa, mas em websites de empresas mais pequenas encontrará secções como esta.
- uma secção com ofertas de emprego. A informação das ofertas de emprego deve ser criada dinamicamente à semelhança do que foi feito no Laboratório 5. Na secção 3 descrevemos com maior detalhe esta secção.

- Caso opte por fazer a funcionalidade extra irá também ter uma secção que compara ofertas de emprego que será acessível a partir desta secção. Na secção 5 descrevemos com maior detalhe esta funcionalidade de comparação de ofertas.
- um link para o portefólio de currículos. Nesta secção a informação será fornecida pelo servidor Node/Express através de uma API REST, sendo obtida a partir de uma base de dados MySQL^[1] que os alunos deverão também desenvolver. A interface desta secção pode ter algumas diferenças, devido à secção ter um propósito mais concreto, mas deve manter alguma consistência com o resto do website. Na secção 4 serão fornecidas mais informações sobre as funcionalidades desta secção.

Em relação ao Layout, o sistema deve seguir as diretrizes indicadas nas aulas teóricas e laboratoriais que permitam ter uma página *Responsive*. Em termos de interfaces gráficas, **cada grupo deverá optar pela opção gráfica que achar mais adequada, desde que se mantenha fiel aos objetivos e requisitos de base propostos para este projeto, sendo esta alvo do processo de avaliação.**

No que toca a documentação, os alunos **devem definir a estrutura das páginas em conjunto e criar os mockups antes de iniciarem o desenvolvimento**. O projeto tem **duas fases** de desenvolvimento, mas os mockups para ambas, bem como o modelo de dados, terão de ser entregues no final da primeira fase (**mais informações sobre cada fase na secção 7.3**).

^[1] Sugere-se o *software* MySQL Workbench como interface gráfica para se trabalhar com o repositório de dados.

2.1 Tecnologias

Com exceção do portefólio de currículos, o restante website deve ser desenvolvido usando um servidor Node + Express a servir páginas estáticas (pasta public) de HTML, CSS e Javascript.

No portefólio de currículos o backend (servidor que fornece os dados) deve ser uma API REST desenvolvida também em Node + Express com ligação a uma base de dados MySQL.

A parte de frontend do portefólio de currículos pode ser feita usando HTML, CSS e Javascript, com chamadas AJAX ao backend para obter os dados dinâmicos que serão colocados nas partes estáticas das páginas.

O motor de templates do Express pode ser usado, mas só para auxiliar na criação da parte estática da página (ex: evitar repetir o cabeçalho/rodapé várias vezes se for o mesmo nas várias páginas). Não pode ser usado na criação do conteúdo dinâmico para que a API REST seja usada.

Em alternativa pode também ser usado React para desenvolver o frontend, usando a API REST. Neste caso ficarão com dois servidores, um para a API REST e outro para o frontend em React. **Nota importante:** React será uma matéria tratada no final do semestre (caso haja tempo para tal), pelo que a sua utilização no projeto é **opcional** e **dependerá em grande medida da pesquisa feita por parte dos alunos**.

Não é obrigatória a utilização de outras frameworks ou bibliotecas, mas será sempre considerada na nota e **deve ser referenciada na documentação**.

3 As ofertas de emprego

Nesta secção são apresentados os requisitos base para a implementação da secção de ofertas de emprego no website.

A informação das ofertas de emprego disponíveis deve ser guardada numa **lista de objectos em javascript**. Esta lista pode estar diretamente no código associado à página onde são apresentadas as ofertas.

Cada oferta deve ter, no mínimo, a seguinte informação:

- Nome da empresa
- Descrição da oferta
- Área (programação, base de dados, administração de sistemas). Utilize o conceito de enumeração aprendido nas aulas.
- Duração da oferta em meses
- Valor total da oferta
- Data de validade da oferta

A informação deve ser apresentada na página **através da geração dinâmica do HTML** tendo como base a lista de objectos acima referida, tal como ensinado nas aulas.

Deve ser também ser possível fazer as seguintes filtrações de e ordenações:

- Filtrar por duração, por área ou ambas.
- Ordenar por valor base ou por data de validade (crescente ou decrescente)

A filtragem e ordenação não devem ser só feitas na página, mas também na lista de dados, sem que estes sejam perdidos. Esta funcionalidade é semelhante ao que foi pedido nas aulas de laboratório, mas a UI pode ser melhorada com o que aprendeu nessas aulas.

Em relação ao Layout, este deve manter a consistência com o layout da restante página, mantendo a responsividade, usabilidade e clareza.

4 A rede de portefólios

Apresentam-se aqui, de forma resumida, as especificações e requisitos básicos que deverão ser consideradas pelos alunos de forma a implementarem uma plataforma de gestão de portefólios.

A plataforma terá de ter a sua própria navegação, páginas e em algumas das opções será necessária a criação de algum conteúdo através do DOM. Contudo, **não é necessário que todo o conteúdo seja construído desta forma**, podendo existir algumas partes onde se poderá optar pela escrita do código HTML na página e gerir as visibilidades desses conteúdos através de *JavaScript*. Um exemplo simples dessa gestão de visibilidades poderá ocorrer com os formulários para inserção dos dados das diversas entidades.

É importante que exista uma zona inicial com descrição do que é o portefólio de currículos, a lista de empresas e as funcionalidades de login e registo.

Terá também de ser **criada a base de dados** que suporta a plataforma. Deve primeiro identificar quais os dados que irá necessitar de guardar e as suas relações e modelar a base de dados em conformidade. Tem também de introduzir uma quantidade de dados na BD que seja suficiente para testar as funcionalidades que implementou na sua aplicação, incluindo situações excecionais (ex: não existirem dados para alguma funcionalidade).

Existem pelo menos 3 tipos de utilizadores: **empresas, administradores e profissionais.**

Empresas

As empresas têm a seguinte informação:

- Nome
- Descrição
- URL do site da empresa
- Email
- Password
- URL do logo da empresa.

Quando uma empresa se regista, o seu registo só fica válido quando um administrador confirma o mesmo. Até essa altura a empresa não aparece na lista de empresas e se tentar fazer login deve aparecer uma mensagem a indicar que aguarda confirmação de um administrador. Caso o pedido tenha sido rejeitado a informação da empresa será apagada (considera-se que a empresa será avisada por outra via, por exemplo por email).

Em termos de funcionalidade as empresas devem conseguir:

- Fazer login usando o email e password
- Ver a lista de portefólios dos profissionais que escolheram ser vistos por empresas.
- Filtrar a lista de portefólios por localidade e por idade.

Administradores

Não existirá registo de administradores (os dados dos mesmos, password e email, já devem estar presentes na BD desde o início).

Ao autenticarem-se os administradores devem ter acesso às seguintes funcionalidades:

- Ver os pedidos de registo de empresas e aceitar ou rejeitar os mesmos.
 - Quando aceite a empresa passa a poder autenticar-se e a sua informação passa a aparecer na lista.
 - Se rejeitado a informação da empresa é apagada (poderá enviar um email automático para o email registado, mas essa funcionalidade não é um requisito, será um extra a ser avaliado juntamente com todos os outros elementos extra).

- Ver todos os portefólios de profissionais, independentemente de ter sido escolhida a opção de ser visto por empresas ou não.
- Filtrar por localidade ou por idade também devem estar disponíveis.

Profissionais

Os profissionais quando se registam têm de preencher a seguinte informação:

- Nome
- Data nascimento
- Género
- Descrição
- Email
- Password
- Localidade
- Indicar se querem ser vistos por empresas

Após registarem-se, os profissionais ficam imediatamente com a possibilidade de fazerem autenticação, não necessitando de confirmação dos administradores.

Em termos de funcionalidade os profissionais devem conseguir:

- Autenticar-se com o email e password que introduziram
- Editar os dados que preencheram no registo
- Adicionar e apagar informação sobre os locais onde trabalharam. Cada local deve ter:
 - Nome do local
 - URL do logo
 - Data de início e de fim para o período em que trabalhou nesse local
 - Descrição das funções desempenhadas
- Adicionar e apagar informação sobre cursos académicos e outras formações. Cada um deve ter:
 - Estabelecimento de ensino
 - Nome do curso
 - Tipo de curso (ex: Licenciatura, Mestrado, curso de formação, etc)
 - Média de notas (podendo não existir se a formação não for avaliada)
- Gerir a lista de amigos. A gestão de lista de amigos tem as seguintes funcionalidades:
 - Fazer pedido de amizade
 - Aceitar pedido de amizade (só após aceite é que ambos ficam na lista de amigos um do outro)
 - Ver lista de amigos.
 - Remover amigos
- Ver os portefólios dos amigos. **Sugestão:** Integre esta funcionalidade na visualização da lista de amigos (ex: um botão que abre o portefólio desse amigo).

A ter em atenção

Todas estas funcionalidades vão gerir dados que vêm da BD e o seu acesso deve ser feito através da API REST.

A plataforma deve impedir o acesso a informação por utilizadores que não tenham permissões para verem essa informação. Essas verificações devem ser **feitas do lado do servidor**.

Tal como referido anteriormente em relação ao Layout, este deve manter a consistência com o layout das outras secções do website, mantendo a responsividade, usabilidade e clareza.

5 Comparador de ofertas (Extra)

O comparador de ofertas deverá ser integrado na secção de visualização de ofertas e deve cumprir os seguintes requisitos:

- Deve ser possível escolher ofertas a partir da lista de ofertas
- As ofertas que já foram escolhidas para comparar devem ser mostradas ao lado, uma descrição curta, e deve ser possível remover ofertas desse conjunto. Este conjunto de ofertas para comparar deve estar sempre visível para o utilizador poder decidir o que fazer.
- Deve existir um número máximo de ofertas para comparar, inicialmente será 3 (deve, no entanto, ser simples a alteração deste valor no código para que possam ser feitos testes). Caso o utilizador escolha mais que 3 (ou o mais do que o valor posteriormente alterado) deve ser mostrado uma mensagem de erro e a oferta não será adicionada.
- No conjunto de ofertas escolhidas para comparar deve também aparecer um botão para comparação. Ao carregar neste botão deve aparecer numa janela modal as ofertas a comparar com toda a informação das mesmas e mais o valor da oferta por mês.
 - Exemplo de como fazer uma janela modal (pode ser feito de outra forma): https://www.w3schools.com/howto/howto_css_modals.asp.
 - Todos os campos presentes na criação de uma oferta de emprego devem ser utilizados e mostrados na comparação (Nome, Descrição, Área, etc) e deve ser adicionado o campo de valor por mês
 - Deve ser possível fechar a janela modal em qualquer altura
 - Deve destacar a melhor oferta (valor por mês mais elevado)

A componente visual desta parte da aplicação ficará a cargo do aluno, mas será importante que não haja uma diferença estética e de layout muito acentuada em relação ao restante website.

6 Sugestões

As interfaces de gestão das diversas entidades podem seguir uma metodologia como a utilizada nos laboratórios dedicados a *JavaScript*. Conseguindo ter um código mais versátil, e tendo em consideração as propriedades existentes em cada tipo de objeto, é possível minimizar o trabalho para a gestão dessas entidades.

Outro aspeto a ter em atenção é que sempre que se pretende criar uma entidade, os campos do formulário devem estar limpos para que não perdurem dados inseridos numa criação/edição anterior (caso se opte por mostrar/esconder formulários pré-existent no HTML).

Terá de ser utilizada a solução de *web framework* Javascript Express (em junção com Node.js) para a vertente servidor da aplicação e, caso haja disponibilidade e conhecimento para tal, os alunos poderão recorrer à utilização de React para a componente do lado do cliente (nomeadamente na área dos Portefólios).

O módulo MySQL para o Node.js terá de ser instalado com o intuito de criar conexões à base de dados MySQL previamente instalada. Aqui será necessário ter-se os dados de acesso ao servidor de base de dados.

Poder-se-á sentir a necessidade de usar extensões do Visual Studio Code para ajudar com JSON, SQL, etc. Os alunos poderão usar o que acharem conveniente para melhorar a sua produtividade e a qualidade do trabalho final.

7 Regras no desenvolvimento do projeto

7.1 Regras de implementação e codificação

O projeto deverá ser desenvolvido segundo as seguintes etapas e tecnologias:

- Etapa 1 - Criação de mockups e definição de requisitos do projecto
- Etapa 2 - HTML, CSS para definição da IU;
- Etapa 3 - JavaScript para interatividade e gerir entidades com geração de objetos por omissão;
- Etapa 4 - Construção de uma base de dados em SQL e de um Webservice RESTful em Node.js, que irá interagir com o site através da tecnologia AJAX;
- Etapa Extra – Desenvolvimento do comparador recorrendo a HTML, CSS e *Javascript*;

Os alunos deverão colocar em prática os conceitos fundamentais da programação para a Web, base de dados e do paradigma de programação orientada por objetos que aprenderam nas aulas teóricas.

É necessário que o projeto cumpra o que é pedido no seu enunciado, **sendo deixado ao critério dos alunos qualquer aspeto de implementação que não seja referido no mesmo, devendo ser apresentado ao respetivo docente de laboratório e devidamente documentado.**

Obviamente que se pretende que a aplicação utilize um modelo suficientemente genérico que possibilite a sua utilização com os diferentes propósitos, conforme seja intenção do utilizador final. No entanto **bastará que o grupo mostre que a aplicação consegue lidar com a publicação de portefólios pessoais, a sua visibilidade e pesquisa pelas empresas.**

7.2 Constituição de grupos

Cada projeto deverá ser elaborado por um grupo de dois alunos do mesmo docente de laboratório, podendo eventualmente ser elaborado individualmente (devendo ter aprovação prévia do respetivo docente de laboratório). **Excepções a estas regras têm obrigatoriamente de ser avaliadas pelos docentes.**

7.3 Submissões do projeto

O projeto deverá ser entregue até à data-limite especificada por via exclusivamente eletrónica, utilizando a área dos trabalhos do respetivo docente de laboratório no Moodle. Existem dois momentos de entrega:

- **Fase 1:** até às **23:55 do dia 05 de Dezembro de 2022**.
 - Ficheiro Zip com a implementação da parte estática do website (**com exceção da gestão de portefólios e o extra**). Alguns pontos importantes:
 - Menu do site (o link para a gestão de portefólios neste momento pode dar para uma página estática a dizer que está em construção)
 - Secção de descrição
 - Secção sobre a equipa de desenvolvimento
 - Secção de oferta de emprego descrita na secção 3 deste enunciado
 - Um documento resumo que apresente:
 - Uma descrição do projeto (incluindo tecnologias de implementação) e constituição do grupo;
 - os requisitos funcionais de base a implementar para todo o website (incluí a gestão de portefólios);
 - mockups correspondentes de todo o website;
 - modelo relacional;
 - lista de **possíveis** frameworks/APIs a usar e com que objetivo (**não** é obrigatório o uso de qualquer framework extra);
- **Fase 2:** até às **23:55 do dia 16 de Janeiro de 2023**. Esta será a entrega final e deverá incluir o projeto completo, englobando todas as etapas (e se pretenderem a etapa extra). O que foi entregue na fase 1 poderá ter sido retificado para esta fase final. **Nota:** em **Época de Recurso** o projecto terá de ser entregue até às **23:55 do dia X de Fevereiro de 2023**.

Os materiais do projeto deverão incluir:

- Pasta do projeto que deverá ter na raiz:
 - Ficheiro `package.json` com a descrição do projeto, em particular as suas dependências em termos de módulos, permitindo o seu carregamento através de `"npm install"`.
 - Ficheiro `app.js` com a implementação do *web server*;
 - Pasta `www` com os ficheiros de implementação do programa: ficheiros `*.html`, pasta `images` para conter as imagens/ícones, pasta `scripts` para conter os ficheiros JavaScript e pasta `styles` para conter os ficheiros CSS.
 - Ficheiro, na linguagem SQL, com todo o código necessário para criação de todas as tabelas necessárias à solução, modelo entidade-relação. **Deverá existir também a inserção de linhas com dados de exemplo que permitam o teste da aplicação.**
- **Os ficheiros JavaScript deverão ser documentados através de JSDoc.**
- Devem também incluir um ficheiro pdf com a documentação da API REST desenvolvida, indicando para cada cada endpoint:
 - Título e descrição curta do que o endpoint faz
 - URL e método

- Parâmetros e dados recebidos pelo endpoint (se existirem)
- Formato do resultado devolvido pelo endpoint caso tudo corra bem (status 200)
- Erros que podem ser devolvidos (status HTTP e formato de resultado para cada erro)

Neste site podem ver uma explicação de como fazer este tipo de documentação:

<https://bocoup.com/blog/documenting-your-api>

Todos os ficheiros que compõem o projeto deverão estar guardados num único ficheiro compactado (formato ZIP ou RAR), cujo nome deverá seguir a seguinte estrutura: `<numAluno1>_<numAluno2>.zip|rar`. **Nota: o não cumprimento desta regra implica a dedução de 0.5 valores na nota final do projecto.**

7.4 Estrutura de ficheiros e pastas do projeto

O projecto deve seguir as boas práticas e exemplos das mesmas mostradas nas aulas. Devem organizar o vosso código de forma modular em que exista separação de responsabilidades. Devem pelo menos ter a separação normalmente designada pelo padrão arquitetural MVC (model-view-controller):

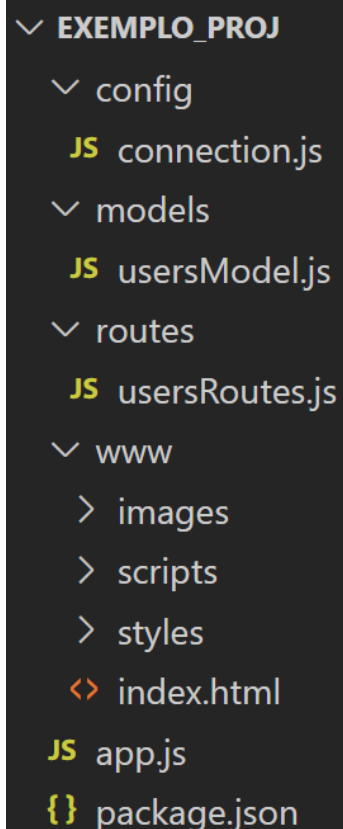
- Models: Uma pasta “models” com os ficheiros que definem, para cada recurso REST, as funcionalidades que manipulam os dados, sejam cálculos ou interações com a BD (também chamada de lógica de negócio)
- Controllers: Uma pasta “routes” com os ficheiros que definem, para cada recurso REST, o roteamento dos pedidos do utilizador: definição de endpoints e receber/enviar dados. Cada rota poderá chamar uma ou mais funcionalidades dos models
- Views: Uma pasta “www” com a estrutura que já foi falada em aulas para ter o CSS, HTML, Javascript e demais ficheiros que irão correr no lado do cliente.

Devem também ter uma pasta “config” que irá ter as configurações de ligação à BD ou outras bibliotecas que queiram utilizar.

Para além disso devem ter os dois ficheiros que normalmente contêm as configurações da aplicação, o **package.json** e o **app.js**.

Ao lado é mostrada uma imagem que mostra o conjunto de pastas que devem usar e um exemplo de alguns ficheiros. Atenção que é suposto criar diversos ficheiros de **Models** e **Routes** caso tenha várias funcionalidades que manipulam recursos REST diferentes.

É obrigatório que todos os projetos tenham as pastas com os nomes apresentados, bem como o ficheiro `app.js` e `package.json`. Podem depois ter mais pastas e ficheiros, dependendo das bibliotecas que decidir usar ou de funcionalidades que considere relevante separar.



```

EXEMPLO_PROJ
├── config
│   ├── connection.js
├── models
│   ├── usersModel.js
├── routes
│   ├── usersRoutes.js
├── www
│   ├── images
│   ├── scripts
│   ├── styles
│   ├── index.html
├── app.js
└── package.json

```

8 Avaliação

8.1 Regras de Avaliação

- A classificação do programa terá em conta a qualidade da programação (fatores de qualidade do *software*), a estrutura do código criado segundo os princípios da programação orientada por objetos, tendo em conta conceitos como a coesão de classes e métodos, o grau de acoplamento entre classes e o desenho de classes orientado pela responsabilidade, e a utilização/conhecimento das linguagens envolvidas.
- Serão premiadas a facilidade de utilização, a apresentação, a imaginação e a criatividade.
- Após a entrega haverá uma discussão oral para validação da nota final da componente Projeto.
- Os alunos que não comparecerem à discussão serão classificados com zero. Nesta discussão com os alunos poderá ser apurada a capacidade do aluno de produzir o código apresentado. A nota atribuída será zero nos casos em que essa capacidade não for demonstrada.
- Após a discussão, a nota do Projeto será atribuída individualmente a cada um dos elementos do grupo.
- A avaliação oral será realizada pelo respetivo docente de laboratório e irá ser feita uma marcação prévia para cada grupo de trabalho.
- A apresentação de relatórios ou implementações plagiadas leva à imediata atribuição de nota zero a todos os trabalhos nessas condições, quer tenham sido o original ou a cópia.
- No rosto do relatório e nos ficheiros de implementação deverá constar o número, nome e turma dos autores e o nome do docente a que se destina.

8.2 Critérios de Avaliação

1. Funcionalidades e operações essenciais	40%
2. Implementação técnica (utilização correta dos elementos HTML; definição e utilização de CSS; desenvolvimento do servidor, etc.)	40%
4. Documentação (bom estilo (identificadores, comentários, indentação); JSDoc, relatório; etc.)	10%
4. Extras e Bónus (utilização de bibliotecas, ferramentas ou tecnologia não lecionada e que faça sentido adicionando real valor ao projeto <u>desde que devidamente justificada pelo aluno</u> ; novos requisitos/funcionalidades, realização da etapa extra; etc.)	10%

9 Anexos

9.1 Sistema de Gestão da Base de Dados (SGBD)

Apesar de não ser o principal foco da Unidade Curricular de Programação para a Internet, o sistema de gestão de base de dados irá ser uma componente da máxima importância no projeto e, como tal, terá o respetivo peso na nota final da segunda fase. E até porque em termos profissionais, a maior parte dos projetos de engenharia informática recorre naturalmente a um armazenamento de dados.

9.1.1 MySQL

Trata-se de um SGBD relacional que usa SQL (Structured Query Language), sendo o seu código fonte aberto aos utilizadores. Neste momento é dos sistemas mais usados do mercado e para isso contribui uma base de documentação relativamente completa. Outro dos motivos da escolha deste SGBD é o facto de ele ter sido utilizado nas anteriores unidades curriculares da licenciatura dedicadas ao estudo de “Bases de Dados”. Poderá efetuar o download do servidor no seguinte URL: <https://dev.mysql.com/downloads/installer/>.

O procedimento de instalação é simples e intuitivo, contudo tenha atenção ao passo de escolha do “porto” de ligação ao SGBD, do nome do utilizador de acesso (por default costuma ser root) e a palavra-passe definida para esse mesmo utilizador. Esta informação será necessária para interagir com o SGBD.

Existe um aplicativo gráfico opcional na instalação (MySQL Workbench), que poderá ser usado pelos alunos de forma a tornar mais apelativa a criação das tabelas, campos, chaves primárias, chaves estrangeiras, índices, procedimentos e funções. Tendo em conta, os conhecimentos e experiência adquirida pelos alunos na utilização do MySQL Workbench na Unidade Curricular de Bases de Dados, a sua utilização pode revelar-se um meio simples e rápido para o desenvolvimento da base de dados de suporte necessária ao sistema da aplicação Web para gestão de videojogos.

9.2 Web Service

Um *Web Service*¹ é um qualquer serviço que está disponível pela internet, que usa uma linguagem de comunicação standard para troca de mensagens, e não está dependente de nenhum sistema operativo ou linguagem de programação.

9.2.1 Abordagem RESTful

Para o desenvolvimento deste projeto deverá recorrer à solução arquitetural RESTful. Apesar de se usar principalmente JSON para a troca de mensagens, é possível recorrer a outros *standards*, como por exemplo XML. Esta abordagem utiliza os métodos (verbos, códigos, recursos) HTTP para efetuar as tradicionais operações de CRUD (ver Tabela 1), não guardando estado entre pedidos.

¹ Para mais informação: https://en.wikipedia.org/wiki/Web_service

OPERAÇÕES	COMANDOS SQL	REST
CREATE	INSERT	POST
READ (ou RETRIEVE)	SELECT	GET
UPDATE	UPDATE	PUT e/ou PATCH
DELETE (ou DESTROY)	DELETE	DELETE

Tabela 1 - Relação entre "Operações CRUD", "Comandos SQL" e "Verbs HTTP".

9.2.2 Comunicação entre a SPA e o serviço RESTful

A arquitetura que se pretende implementar poder ser visualizada na Figura 1.

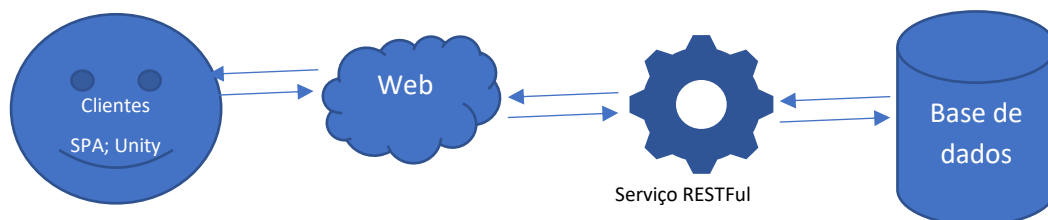


Figura 1 - Arquitetura.

O objetivo será que através do serviço RESTful se consiga a informação que irá iniciar os objetos das classes que já foram desenvolvidas anteriormente.

Caso tenha construído os objetos por omissão para testar o site numa fase anterior, terá de se remover a geração e criar-se as respetivas funções auxiliares recorrendo à tecnologia AJAX (*Asynchronous JavaScript and XML*), que será explicada em aulas futuras. Serão efetuados os pedidos necessários por parte do cliente ao serviço RESTful. Abaixo pode-se observar um exemplo de teste para verificar se é possível carregar o recurso uma determinada entidade.

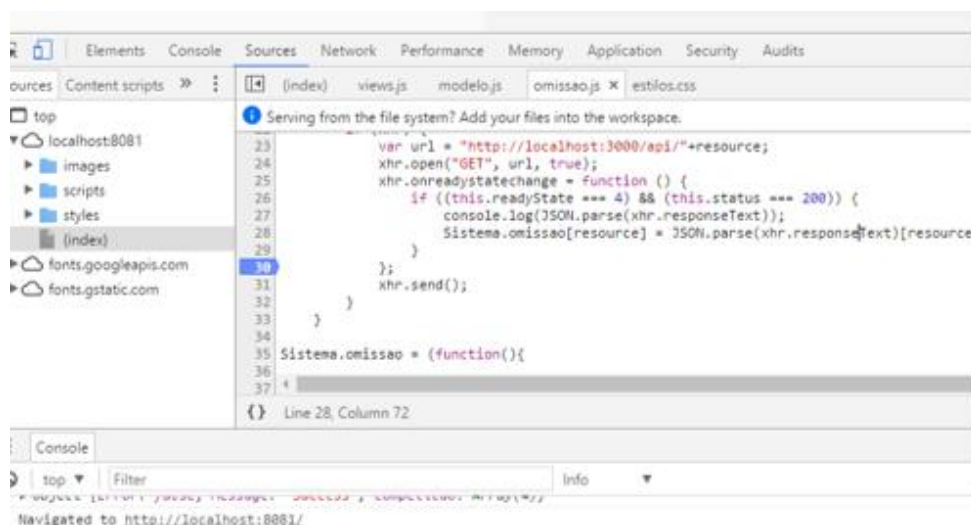


Figura 2 - Exemplo de consumo do serviço RESTful por parte do cliente (ambiente de debug do Chrome).