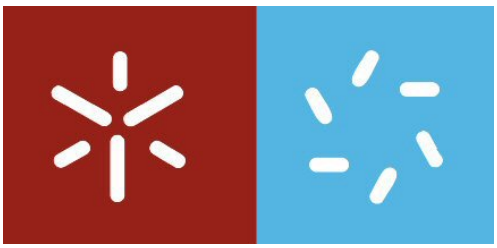


Universidade do Minho

Licenciatura em Ciências da Computação

Projeto - ComputationalMind
Desafios para Desenvolvimento Computacional

Inês Pires Presa (A90355) Ivo Miguel Gomes Lima (A90214)
Tiago André Oliveira Leite (A91693)
Tiago dos Santos Silva Peixoto Carriço (A91695)

29 de junho de 2022

Resumo

Este relatório retrata todo o processo de desenvolvimento da plataforma *"ComputationalMind"* e as diferentes decisões tomadas ao longo do mesmo. O objetivo deste projeto é criar um *Website* educacional para uso em escolas e outras instituições de ensino, que possibilite ao público mais jovem aprender por meio de jogos de escolha múltipla, resposta curta e verdadeiro ou falso, o *mindset* a adotar para ser um excelente *problem solver* e, consequentemente, um excelente programador. O desenvolvimento começou pelo desenho do modelo lógico da base de dados e validação da mesma junto dos proponentes. Para tal foram utilizadas as ferramentas *draw.io* e *MySQL WorkBench*. A implementação física da base de dados foi realizado através da *framework Django* e o sistema de gestão de bases de dados escolhido foi o *MySQL*. O *back-office* foi construido em *Django* com recurso à ferramenta *Django rest framework* que possibilita o desenvolvimento de *Web APIs*. O *front-office* foi feito com *ReactJS* e *tailwindcss* sendo que o seu processo de construção começou com o desenho de *mockups* no *figma.com*. O fluxo dos dados inseridos pelo utilizador é feita do *front-office* para o *back-office* pela *rest API* através de ficheiros *Json*. Os dados enviado pelo *ReactJS* são recebido pelo *Django*, validados e caso estejam no formato correcto, são inseridos na base de dados. O fluxo de dados da BD para o utilizador é realizada de modo análogo. A plataforma pode ser acedida por meio de um navegador *web* no computador ou telemóvel.

Palavras-Chave: Pensamento Computacional, *web site*, aprendizagem, *back-office*, *front-office*, *back-end*, *front-end Rest framework*, *Django*, *ReactJS*, *Base de dados*.

Conteúdo

Capítulo 1

Introdução

Na unidade curricular Projeto da licenciatura de Ciências da Computação, foi-nos proposto o desenvolvimento de uma plataforma que permita inserir problemas/desafios para que os utilizadores possam, em modo jogo, responder aos mesmos. Esta plataforma deve ser constituída por uma interface de *back-office* (BO) e uma interface de *front-office* (FO).

Para auxiliar na criação deste projeto, inspirámo-nos em websites já existentes com um propósito muito semelhante ao nosso, como, por exemplo, o *Kahoot*[1], o *Socrative*[2] e o *Hypatiamat*[3].

Relativamente à estrutura do nosso trabalho, este foi dividido em três etapas distintas:

- A primeira etapa, consiste na caracterização do *web site* a ser desenvolvida, onde é feita a apresentação do plano de desenvolvimento, bem como o estudo das medidas para a satisfação dos objetivos.
- A segunda etapa, consiste em materializar as ideias obtidas na primeira fase, existindo assim, uma descrição detalhada do software que irá ser desenvolvido. Nesta etapa faz-se, também, um planeamento da base de dados a implementar.
- A terceira e última etapa, a Implementação, consiste no desenvolvimento do software bem como a implementação das funcionalidades fundamentais, tendo em conta o planeamento das etapas anteriores. É nesta fase que se faz a validação da aplicação e se apresenta o produto final.

[1] https://kahoot.it
[2] https://socrative.com
[3] https://hypatiamat.com

Capítulo 2

Enunciado do Projeto

No âmbito do treino do Pensamento Computacional, pretende-se criar uma plataforma que permita inserir problemas/desafios para que os utilizadores possam, em modo jogo, responder a esses desafios.

A plataforma deve ter uma interface de *back-office* (BO) para armazenar os problemas/desafios (nomeadamente o seu enunciado, opções de resposta, resposta certa, faixa etária, etc) apresentados na *página da web* do bebras[1]. Esta conterá toda a informação necessária ao sistema, e é responsável pela gestão do *site* e das suas funcionalidades. A interface de *front-office* (FO) oferecerá a interação ao utilizador para que, no modo jogo este possa responder aos desafios selecionados a partir do repositório criado no BO da plataforma, que lhe atribuirá pontos, prémios, ou outras formas de recompensa por forma a motivá-lo e envolvê-lo mais.

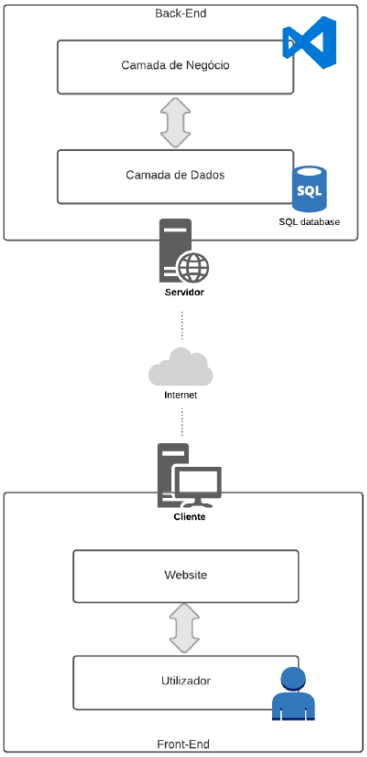


Figura 2.1: Esquema do Sistema

[1] http://bebras.dcc.fc.up.pt/problems/2021/problemas_09_10.pdf

Capítulo 3

Concessão da Solução

Para a elaboração de todo o projeto foi necessário recorrer a diversos recursos que ajudassem na realização do mesmo.

Antes de mais, o principal recurso utilizado foi a equipa de docentes que nos esclareceu e ajudou na tomada de decisões ao longo de todas as fases do projeto.

Inicialmente, com a escolha do tema foi necessário recolher uma série de artigos e questões com o intuito de nos familiarizar com os diferentes tipos de mídia que a plataforma teria de suportar e possibilitar a sua mesclação.

Após essa pesquisa, surgiu a necessidade de ser feita a modelação da Base de Dados, junto com a esquematização da página através da criação de Mockups [1]. Na nossa visão estes passos representam um pouco os pilares e ideias sobre os quais o nosso projeto acenta, daí terem tomado uma parte significativa do nosso tempo.

De seguida, foram selecionadas as *frameworks* para criação tanto da *parte de suporte*, como da *interface frontal*, sendo elas o *Django* [2] e o *Reactjs* [3] junto com o *TailWindCSS* [4]. Nesta parte, destaca-se a fase de teste e de adaptação com as ferramentas em uso. Após esta familiarização, seguiu-se a etapa das correções dos eventuais erros e de ajustes por forma a nos certificarmos que o produto final correspondia às nossas expetativas e às do público-alvo.

[1] https://figma.com
[2] https://djangoproject.com
[3] https://reactjs.org
[4] https://tailwindcss.com

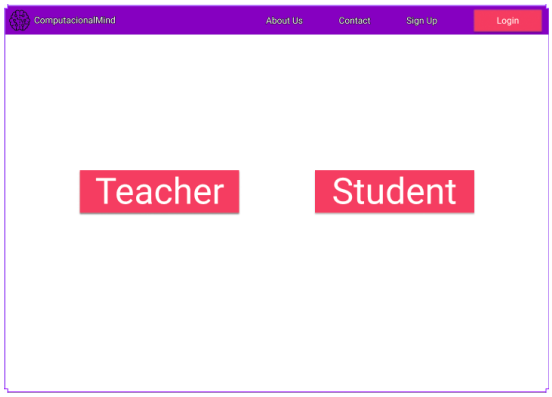
Capítulo 4

Mockups

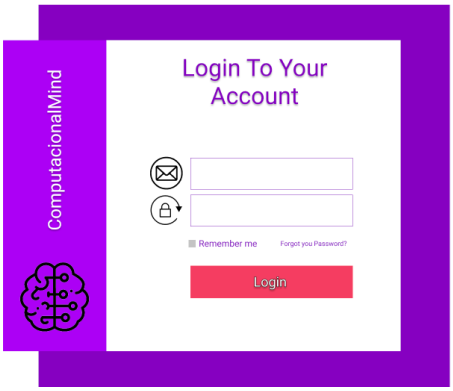
Durante o processo de criação do site propuseram-nos a criação de uma maquete (*mockup*), que serviria como uma protótipo visual da forma como a *página da web* poderia ser apresentada, permitindo-nos testar como os vários elementos visuais funcionam em conjunto.

Como o *mockup* é um design estático, este não tem as funcionalidades oferecidas por um site vivo, por exemplo, não abrirá um *pop-up* quando se clica no botão *Login*, entre outros elementos. Fizemo-lo com o intuito de apresentar muitos dos elementos finais, da maneira como queríamos o *design*, mas como em todas as maquetes algumas ideias foram modificas e eliminadas, devido a sugestões propostas pelos docentes da UC e para facilitar o processo de implementação.

Em última análise, percebemos que o *mockup*, aparece no ponto intermédio do processo de criação de *web design*. A maquete de uma página deve ser criada com um propósito específico em mente, e esta possibilitou-nos visualizar como poderíamos alcançar o objetivo final. Nomeadamente, como poderíamos concretizá-la através da utilização de padrões de marca e criatividade visual.

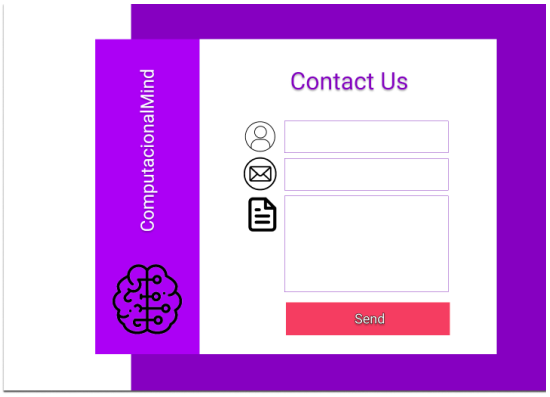


(a) Página Principal

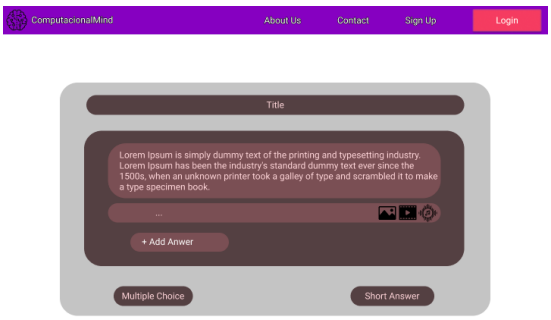


(b) Página de Login

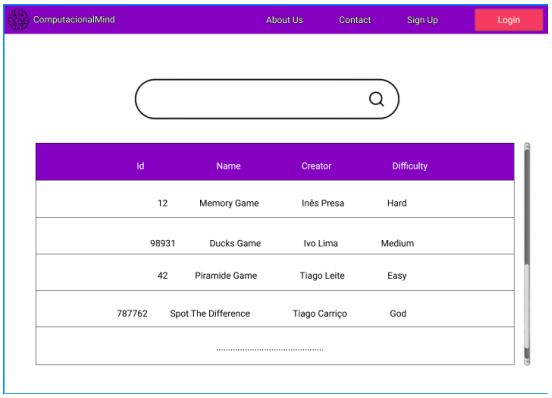
(a) Página de Criação de Conta



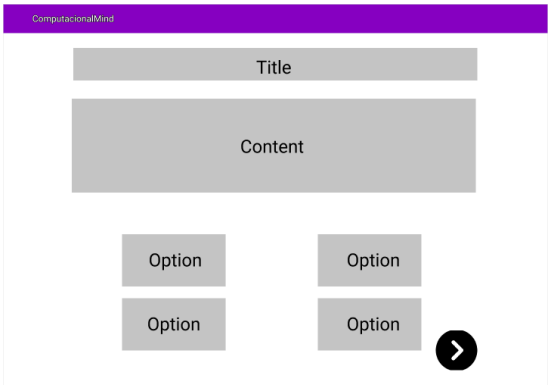
(b) Página de Contacto



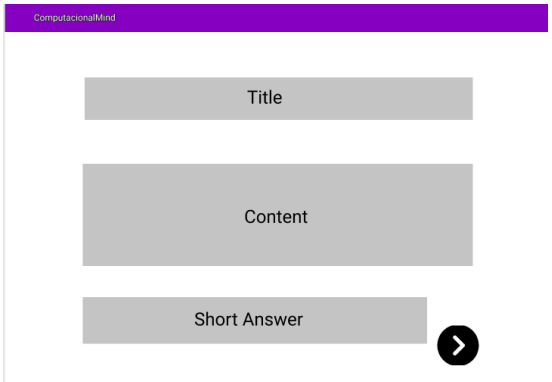
(c) Menu de Inserção de Jogo



(d) Menu de Seleção de Jogo

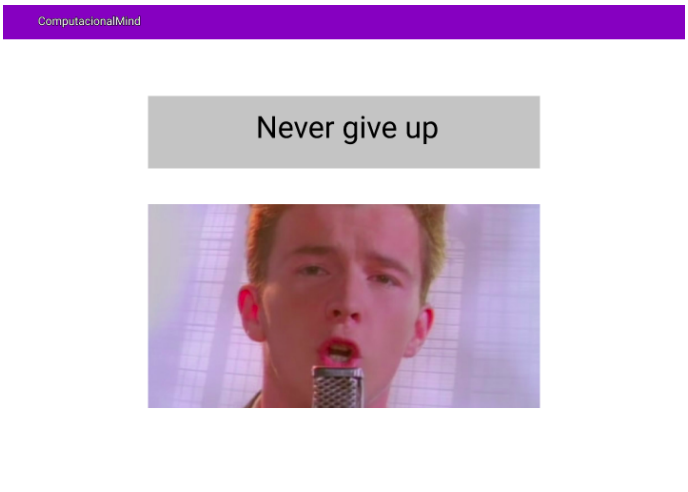


(e) Jogo de Escolha Múltipla



(f) Jogo de Resposta Curta

(a) Resposta certa



(b) Resposta errada

Capítulo 5

Base de Dados

Para este projeto decidimos implementar uma Base de Dados Relacional que permita analisar e relacionar os dados do *Computational Mind* com vista a melhor o serviço, uma vez que esta deverá possibilitar a manipulação e consulta dos dados de uma forma ágil e segura. Este será um processo trabalhoso, mas que trará grandes frutos no futuro, uma vez que será substancialmente mais eficiente, especialmente na atualização, consulta e tratamento dos dados, o que permitirá alcançar todos os objetivos propostos pela equipa docente. Para além disso, será também um sistema mais fiável, visto que garantirá uma uniformização dos dados, garantindo que qualquer interveniente que pretenda consultar ou alterar os dados o fará de uma forma mais segura e controlada.

5.1 Identificação e caracterização das entidades

5.1.1 Author

Um autor é identificado por um **código**, devendo ter também uma referência do seu **nome**, do ***e-mail*** e da ***password***.

5.1.2 Player

Um jogador é identificado por um **código**, devendo ter também uma referência do seu **nome**, da **data de aniversário**, do ***e-mail*** e da ***password***.

5.1.3 Question

Uma questão é identificado por um **código**, devendo ter também uma referência ao **autor**, um **título**, um identificador do **tipo** (resposta curta, escolha múltipla ou verdadeiro e falso), uma **classificação**, a **dificuldade** e a **idade mínima**.

5.1.4 Content

Um conteúdo é identificado por uma referência à **questão**, uma **ordem** pela qual as questões aparecem, o **tipo** do conteúdo e os **links/imagens/vídeos** associados.

5.1.5 Option

Uma opção é identificada por um **código**, uma referência à **questão**, a **resposta** para serem feitas as verificações e a **opção correta**.

5.1.6 History

Um histórico é identificado por **código**, uma referência ao **jogador**, uma referência à **questão**, a **data** que o jogador respondeu à questão, as suas **respostas** e quais **acertou**.

5.2 Identificação e caracterização dos relacionamentos

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Question	1..N	tem	1	Author
Option	1..N	tem	1	Question
History	1..N	tem	1	Question
History	1..N	tem	1	Player
Content	1..N	tem	1	Question

5.3 Modelo lógico

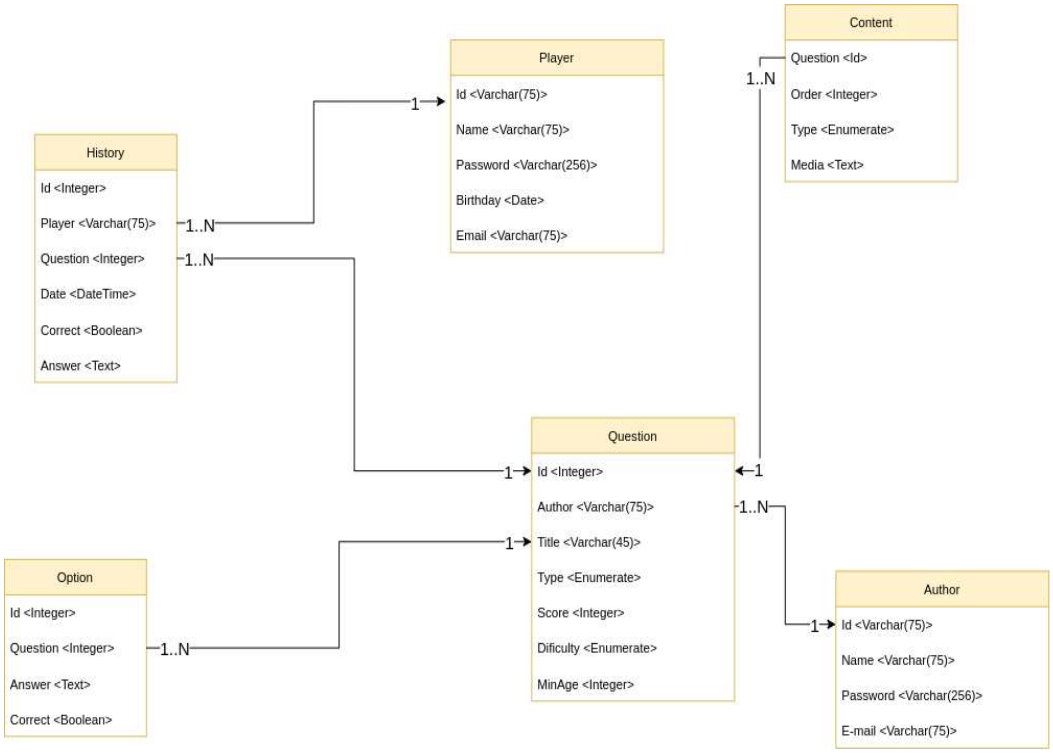


Figura 5.1: Esquema lógico

5.4 Alterações

Por forma a acrescentar mais funcionalidade e também devido as restrições no desenvolvimento do *back-end*, foram realizadas algumas alterações no modelo inicial da Base de Dados.

5.4.1 User

As entidades *Author* e *Player* passam a ser a entidade *User* que tem um atributo *type* para fazer a distinção entre ambos.

5.4.2 Quiz

Um quiz é constituído por várias questões. É identificado através de um **código**, devendo ter ainda uma referência ao **autor** e um **título**.

5.4.3 Identificação e caracterização dos relacionamentos

Entidade	Multiplicidade	Relacionamento	Multiplicidade	Entidade
Quiz	1..N	tem	1	User
Question	1..N	tem	1	User
Question	1..N	tem	1	Quiz
Option	1..N	tem	1	Question
History	1..N	tem	1	Question
History	1..N	tem	1	User
Content	1..N	tem	1	Question

5.4.4 Modelo lógico final

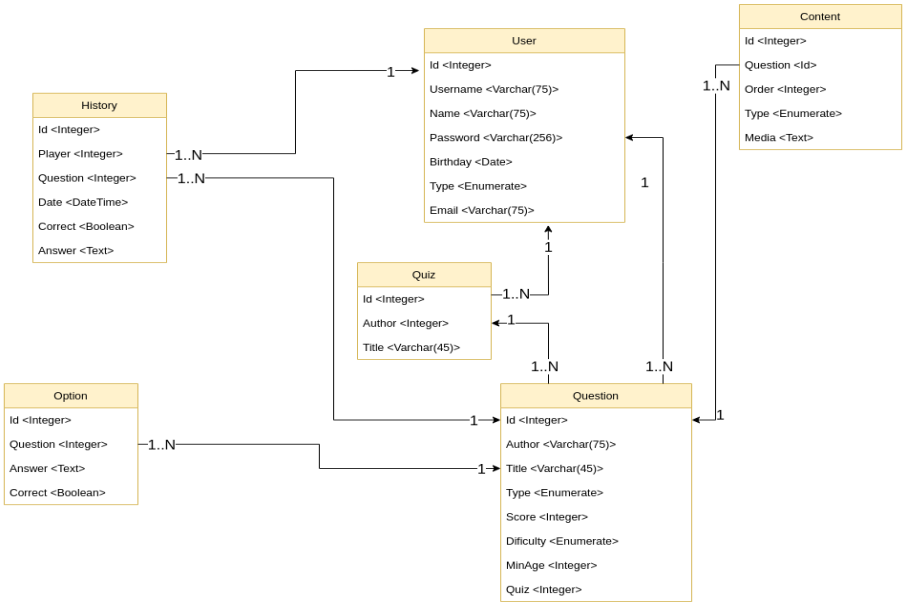


Figura 5.2: Esquema lógico final

Capítulo 6

Back-end

6.1 Ferramentas

Para o desenvolvimento da *parte secundária* utilizou-se *Django* e *Django REST Framework*.

6.2 Django

Django é uma *framework* gratuita e de código aberto, com o intuito de facilitar e acelerar o desenvolvimento *web*, escrito em Python, que utiliza o padrão *model-template-view* (MTV) e o princípio *don't repeat yourself* (DRY) através do reaproveitamento de código já feito. O *web server*, pode ajudar os *developers* a produzir de forma eficiente um FO ou BO rico em recursos, seguro e escalável.

O *Django REST Framework* é um conjunto de ferramentas utilizadas para construir *APIs* (Interfaces de Programação de Aplicações) para *Web*.

Apesar do *Django* possibilitar o desenvolvimento do *front-end* e do *back-end*, a utilização de uma *REST API* permite criar uma *parte de suporte* mais genérica, e torna possível a construção de uma *parte frontal* para diversos dispositivos no futuro, tornando a nossa aplicação mais escalável.

6.3 Objetivo

O *back-end* do projeto tem como objetivo fazer a ligação entre a Base de Dados e a *interface frontal*, tendo sido desenvolvido ainda uma *REST API* que será "consumida"pela mesma.

6.4 Desenvolvimento

O processo de desenvolvimento começou pela construção da base de dados através do *Django*, com base no esquema lógico definido. Para tal, foram construídos todos os modelos necessários para representar as várias entidades e os seus relacionamentos. A construção da base de dados foi feita com recurso à funcionalidade *migrate* disponibilizada pelo *Django*. O sistema de gestão de base de dados utilizado foi o *MySQL*. Devido a algumas restrições imposta pelo *Django*, foi necessário realizar algumas alterações no esquema lógico, inicialmente, definido.

A principal função do *Django* é servir de intermediário da comunicação entre a *parte frontal* e a base de dados. Essa comunicação é realizada através da *REST API* com o envio de ficheiros em formato *JSON*. Para tal, o *Django* possui funções específicas para serializar os dados.

A comunicação entre os constituintes desta plataforma ocorre da seguinte forma:

1. O *ReactJs* faz uma pedido através da *REST API*.
2. Este pedido é analisado *Django* e, caso esteja correto, os dados requeridos são extraídos da base de dados, convertidos e enviados para a *interface frontal* no formato *JSON*.

É de destacar que a comunicação inversa é também possível e processa-se de modo análogo.

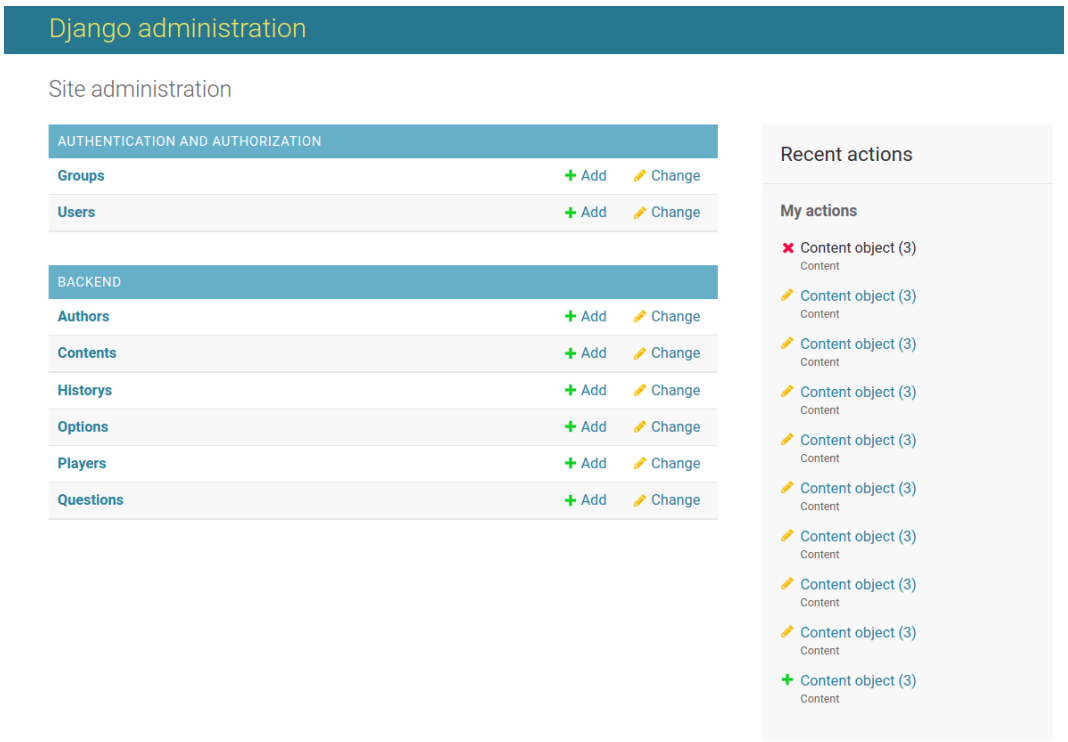


Figura 6.1: REST API

6.5 Manual da API

6.5.1 /api/profile/

tipo: GET.
função: obter dados do *user* que está autenticado.
requisitos: autenticação.
resposta:

```
{
    "id": 9,
    "username": "username",
    "name": "name",
    "birthday": null,
    "email": "email@email.com",
    "type": "A"
}
```

status correto: 200.
status errado: 401.

6.5.2 /api/profile/password/

tipo: POST.
função: alterar a *password* do *user* que está autenticado.
requisitos: autenticação.
pedido:

```
{
    "current":"0123",
    "new":"56789"
}
```

status correto: 200.
status errado: 403.

6.5.3 /api/users/

tipo: POST.
função: alterar a *password* do *user* que está autenticado.
requisitos: autenticação.
pedido:

```
{
      "username": "username",
      "name": "name",
      "password": "password",
      "email": "email@email.com",
      "birthday": null,
      "type": "A"
 }
```

status correto: 200.
status errado: 403.

6.5.4 /api/users/<str:username>

tipo: GET.
função: obter a informação do *user* que esta autenticado e com *usernanme* = <str:username>
requisitos: autenticação.
resposta:

```
{
      "id": 1,
      "username": "username",
      "name": "name",
      "birthday": null,
      "email": "email@email.com",
      "type": "A"
 }
```

status correto: 200.
status errado: 404.

6.5.5 /api/questions/

tipo: GET.
função: obter todas as questões.
requisitos: autenticação.
resposta:


```
[
    {
        "id": 1,
        "author": "username",
        "title": "Teste",
        "type": "SA",
        "score": 10,
        "dificulty": "E",
        "minage": 5,
        "options": [
            {
                "id": 1,
                "answer": "Ola Mundo",
                "question": 1,
                "correct": true
            },
            {
                "id": 2,
                "answer": "Mundo",
                "question": 1,
                "correct": false
            }
        ],
        "contents": [
            {
                "id": 1,
                "question": 1,
                "order": 1,
                "type": "T",
                "text": "Escreva Ola Mundo",
                "media": null
            }
        ],
        "quiz": 1
    }
]
```


status correto: 200.
status errado: 401.
Nota: Se o *user* que faz o pedido for do tipo *P* o campo *correct* nas *options* é removido.

6.5.6 /api/questions/<int:id>

tipo: GET.
função: obter a questão com *id* = <int:id>.
requisitos: autenticação.
resposta:


```
{
    "id": 1,
    "author": "username",
    "title": "Teste",
    "type": "SA",
    "score": 10,
    "dificulty": "E",
    "minage": 5,
    "options": [
        {
            "id": 1,
            "answer": "Ola Mundo",
            "question": 1,
            "correct": true
        },
        {
            "id": 2,
            "answer": "Mundo",
            "question": 1,
            "correct": false
        }
    ],
    "contents": [
        {
            "id": 1,
            "question": 1,
            "order": 1,
            "type": "T",
            "text": "Escreve Ola Mundo",
            "media": null
        }
    ],
    "quiz": 1
}
```


status correto: 200.
status errado: 401.
Nota: Se o *user* que faz o pedido for do tipo *P* o campo *correct* nas *options* é removido.

6.5.7 /api/questions/user/

tipo: GET.
função: obter todas as questões do autor autenticado.
requisitos: autenticação.
resposta:


```
[
    {
        "id": 1,
        "author": "username",
        "title": "Teste",
        "type": "SA",
        "score": 10,
        "dificulty": "E",
        "minage": 5,
        "options": [
            {
                "id": 1,
                "answer": "Ola Mundo",
                "question": 1,
                "correct": true
            },
            {
                "id": 2,
                "answer": "Mundo",
                "question": 1,
                "correct": false
            }
        ],
        "contents": [
            {
                "id": 1,
                "question": 1,
                "order": 1,
                "type": "T",
                "text": "Escreve Ola Mundo",
                "media": null
            }
        ],
    }"quiz": 1
]
```


status correto: 200.
status errado: 401.

6.5.8 /api/questions/insert/

tipo: POST.
função: inserir uma questão
requisitos: autenticação e ser um *user* do tipo *author*
pedido:

```
{
      "title": "title",
      "type": "SA",
      "score": 10,
      "dificulty": "E",
      "minage": 5,
      "options": [
          {
              "answer": "Ola Mundo",
              "correct": true
          }
      ],
      "contents": [
          {
              "order": 1,
              "type": "T",
              "text": "Escreva Ola Mundo"
          }
      ]
 }
```

resposta:

```
{
    "id": 1,
    "author": 1,
    "title": "title",
    "type": "SA",
    "score": 10,
    "dificulty": "E",
    "minage": 5,
    "options": [
      {
        "id": 1,
        "answer": "Ola Mundo",
        "question": 1,
        "correct": true
      }
    ],
    "contents": [
      {
        "id": 1,
        "question": 1,
        "order": 1,
        "type": "T",
        "text": "Escreva Ola Mundo",
        "media": null
      }
    ],
    "quiz": null
```

}

status correto: 201.
status errado: 401 ou 403.

6.5.9 /api/questions/update/<int:id>

tipo: POST.
função: alterar a questão com *id* = <int:id>
requisitos: autenticação e ser o autor da questão.
pedido:

```
{
    "title": "title",
    "type": "SA",
    "score": 10,
    "dificulty": "E",
    "minage": 5,
    "options": [
        {
            "id": 1,
            "answer": "Ola",
            "correct": true
        }
    ],
    "contents": [
        {
            "id":1,
            "order": 1,
            "type": "T",
            "text": "Escreva Ola"
        }
    ]
}
```

resposta:

```
{
    "id":
    "title": "title",
    "type": "SA",
    "score": 10,
    "dificulty": "E",
    "minage": 5,
    "options": [
        {
            "id": 1,
            "answer": "Ola",
            "correct": true
        }
    ],
    "contents": [
        {
            "id":1,
```

```
            "order": 1,
            "type": "T",
            "text": "Escreva Ola",
            "media" null
        }
    ],
    "quiz": null
}
```

Status correto: 200.

Status errado: 403 ou 404.

nota: os campos que não forem enviado serão mantidos. Só serão mantidas as *options* e questions para as quais seja enviado o seu *id*. Se não for enviado o *id*, serão criadas novas instâncias.

6.5.10 /api/questions/delete/<int:id>

tipo: POST.

função: apagar a questão com *id* = <int:id>

requisitos: autenticação e ser o autor da questão.

status correto: 200.

status errado: 403 ou 404.

6.5.11 /api/quizzes/

tipo: GET.

função: obter todos os quizzes.

requisitos: autenticação.

resposta:

```
[{
"id": 1,
"questions": [
  {
            "id":
            "title": "title",
            "type": "SA",
            "score": 10,
            "dificulty": "E",
            "minage": 5,
    "options": [
        {
            "id": 1,
            "answer": "Ola",
            "correct": true
        }
    ],
    "contents": [
        {
            "id":1,
            "order": 1,
            "type": "T",
            "text": "Escreva Ola",
```

```
            "media" null
        }
    ],
    "quiz": 1
    }],
  "author": 1,
  "title": "quiz"
}]
```


status correto: 200.
status errado: 401.
Nota: Se o *user* que faz o pedido for do tipo *P* o campo *correct* nas *options* de cada questão é removido.

6.5.12 /api/quizzes/<int:id>

tipo: GET.
função: obter o quiz com *id* = <int:id>.
requisitos: autenticação.
resposta:


```
{
"id": 1,
"questions": [
  {
            "id":
            "title": "title",
            "type": "SA",
            "score": 10,
            "dificulty": "E",
            "minage": 5,
    "options": [
        {
            "id": 1,
            "answer": "Ola",
            "correct": true
        }
    ],
    "contents": [
        {
            "id":1,
            "order": 1,
            "type": "T",
            "text": "Escreva Ola",
            "media" null
        }
    ],
    "quiz": 1
    }],
  "author": 1,
  "title": "quiz"
}
```


status correto: 200.

status errado: 401 ou 404.

Nota: Se o *user* que faz o pedido for do tipo *P* o campo *correct* nas *options* de cada questão é removido.

6.5.13 /api/quizzes/user/

tipo: GET.

função: obter todos os quizzes do autor autenticado.

requisitos: autenticação.

resposta:


```
[{
"id": 1,
"questions": [
  {
            "id":
            "title": "title",
            "type": "SA",
            "score": 10,
            "dificulty": "E",
            "minage": 5,
    "options": [
        {
            "id": 1,
            "answer": "Ola",
            "correct": true
        }
    ],
    "contents": [
        {
            "id":1,
            "order": 1,
            "type": "T",
            "text": "Escreva Ola",
            "media" null
        }
    ],
    "quiz": 1
    }],
  "author": 1,
  "title": "quiz"
}]
```


status correto: 200.

status errado: 401.

6.5.14 /api/quizzes/insert/

tipo: POST.
função: inserir um quiz
requisitos: autenticação e ser um *user* do tipo *author*
pedido:

```
{
"questions": [
  {
            "title": "title",
            "type": "SA",
            "score": 10,
            "dificulty": "E",
            "minage": 5,
    "options": [
        {
            "answer": "Ola Mundo",
            "correct": true
        }
],
    "contents": [
        {
            "order": 1,
            "type": "T",
            "text": "Escreva Ola Mundo"
        }
    ]
    }],
    "title": "quiz"
}
```

resposta:

```
{
  "id": 1,
  "questions": [
    {
      "id": 1,
      "author": 1,
      "title": "title",
      "type": "SA",
      "score": 10,
      "dificulty": "E",
      "minage": 5,
      "options": [
        {
          "id": 1,
          "answer": "Ola Mundo",
          "question": 134,
          "correct": true
        }
      ],
      "contents": [
        {
          "id": 1,
          "question": 1,
```

```
            "order": 1,
            "type": "T",
            "text": "Escreva Ola Mundo",
            "media": null
          }
        ],
        "quiz": 1
      }
    ],
    "author": 1,
    "title": "quiz"
}
```

status correto: 201.
status errado: 401 ou 403.

6.5.15 /api/quizzes/update/<int:id>

tipo: POST.
função: alterar o quiz com *id* = <int:id>
requisitos: autenticação e ser o autor do quiz.
pedido:

```
{
  "questions": [
    {
      "id":1,
      "author": 1,
      "title": "title",
      "type": "SA",
      "score": 10,
      "dificulty": "E",
      "minage": 5,
      "options": [
        {
          "id":1,
          "answer": "Ola",
          "correct": true
        }
      ],
      "contents": [
        {
          "id":1,
          "order": 1,
          "type": "T",
          "text": "Escreva Ola"

        }
      ]

    }
  ]
}
```

resposta:


```
{
  "id": 1,
  "questions": [
    {
      "id": 1,
      "author": 1,
      "title": "title",
      "type": "SA",
      "score": 10,
      "dificulty": "E",
      "minage": 5,
      "options": [
        {
          "id": 223,
          "answer": "Ola",
          "question": 211,
          "correct": true
        }
      ],
      "contents": [
        {
          "id": 1,
          "question": 1,
          "order": 1,
          "type": "T",
          "text": "Escreva Ola",
          "media": null
        }
      ],
      "quiz": 1
    }
  ],
  "author": 1,
  "title": "quiz"
}
```


Status correto: 200.
Status errado: 403 ou 404.

6.5.16 /api/quizzes/delete/<int:id>

tipo: POST.
função: apagar o quiz com *id* = <int:id>
requisitos: autenticação e ser o autor do quiz.
status correto: 200.
status errado: 403 ou 404.

6.5.17 /api/history/

tipo: POST.
função: inserir a resposta a uma questão.
requisitos: autenticação e o *user* ser do tipo *Player*.
pedido:

```
{
    "question": "1",
    "answer": "Ola"
}
```

resposta:

```
{
    "id": 1,
    "player": 2,
    "question": 1,
    "date": "2022-06-27T01:09:23Z",
    "correct": true,
    "answer": "Ola"
}
```

status correto: 201.
status errado: 401 ou 404.

6.5.18 /api/history/user/

tipo: GET.
função: obter o historio de respostas do utilizador que está autenticado.
requisitos: autenticação e o *user* ser do tipo *Player*.
pedido:

```
{
    "question": "1",
    "answer": "ola"
}
```

resposta:

```
[
    {
        "id": 1,
        "player": "2",
        "question": 1,
        "date": "2022-06-27T01:09:23Z",
        "correct": true,
        "answer": "Ola"
    },
    {
        "id": 2,
        "player": "2",
        "question": 2,
        "date": "2022-06-26T22:02:10Z",
        "correct": true,
        "answer": "mundo"
    }
]
```

status correto: 200.
status errado: 401.

6.5.19 /api/history/question/<int:id>

tipo: GET.
função: obter o historio de respostas da questão com *id* = <int:id>.
requisitos: autenticação e ser o autor da questão.
resposta:


```
[
    {
        "id": 1,
        "player": "2",
        "question": 1,
        "date": "2022-06-27T01:09:23Z",
        "correct": true,
        "answer": "Ola"
    }
]
```


status correto: 200.
status errado: 401 ou 404.

6.5.20 /api/login/

tipo: POST.
função: autenticar um *user*.
pedido:


```
{
    "username": "username",
    "password":"password"
}
```


resposta:


```
{
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTY1NjM3
    ODUzMiwiaWF0IjoxNjU2MjkyMTMyLCJqdGkiOiIyMjFjZjAz
    YWUxODE0Nzc5ODI3NjNmOWMxOGYwYjY1MyIsInVzZXJfaWQiOjl9.
    8Qlwm7xgbjZFEH0ndK7CiNS1P-D4FCoRnPb4jLf7-Io",

    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjU2Mjk1Nz
    MyLCJpYXQiOjE2NTYyOTIxMzIsImp0aSI6IjdlMTM3OTE5OWM1
    OTQ3NWViODFmMGQ2Y2MyMWU1OTFkIiwidXNlcl9pZCI6OX0.
    TDjrl61WosT_hMp1UMCOCI-iG3DZnU93QKuu3aIAVho"
}
```


status correto: 200.
status errado: 401.

6.5.21 /api/login/refresh/

tipo: POST.
função: atualizar o *token* quando expirado.
pedido:


```
{
    "refresh":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTY1NjM3O
    TY4OSwiaWF0IjoxNjU2MjkzMjg5LCJqdGkiOiI2ZDcwMjQ5N2
    VjYTE0ZmQ0YjQzMjM5YzMzNDQ5NjNiMiIsInVzZXJfaWQiOjl9.
    0lB2EG0DjDjIiZt2OqBuUjy1Zw8c71uCOHf-UMTHY2I"
}
```


resposta:


```
{
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjU2Mjk3N
    DQxLCJpYXQiOjE2NTYyOTMyODksImp0aSI6ImY1YjNhOWY2ZDY
    zYzRjOWY4Yjc5ZWFkODRkNTJiMzEyIiwidXNlcl9pZCI6OX0.
    SPNTGHdXwGwVod-1VZLSJvRIRo5mIECk6brbTqwOE34"
}
```


status correto: 200.
status errado: 401.

6.5.22 /api/login/verify/

tipo: POST.
função: verificar se o *token* é válido.
pedido:


```
{
    "token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNjU2Mjk2
    ODg5LCJpYXQiOjE2NTYyOTMyODksImp0aSI6Ijg1YzM3YWJi
    OTZjNTQ0Y2U4M2M2ZmY5MzBiMTgwMTdjIiwidXNlcl9pZCI6OX0.
    -C5f3KhAUgN_udTNjgQtTEIYqc80zFmafsjb19nHshc"
}
```


resposta:
status correto: 200.
status errado: 401.

Capítulo 7

Front-End

7.1 Ferramentas

Para o desenvolvimento do *front-end* utilizou-se *React* e *TailWindCSS*.

7.2 React e TailWindCSS

React.js é uma biblioteca *JavaScript* de código aberto usada para construir o *front-end* de aplicativos e de páginas *web* utilizando a lógica de página única. Sendo usada para lidar com a camada de visualização para as aplicações da *Web* e de telemóvel.

Para além disso, esta também nos permite criar componentes para a interface do usuário que sejam reutilizáveis para que os *developers* criem grandes aplicações, por forma a facilitar a alteração dos dados sem que seja necessário o recarregar a página. O seu principal objetivo é ser rápido, escalável e simples. Este funciona apenas em interfaces de usuário no aplicativo, o que corresponde à exibição do modelo *Model-View-Controller* (MVC), existindo a possibilidade de ser usado em conjunto com outras bibliotecas ou *frameworks* de *JavaScript*, como *Angular JS* em MVC.

Tailwind CSS corresponde, basicamente, a uma *framework* de *CSS* fundamental para a construção rápida de interfaces de utilizador personalizadas. Sendo que uma estrutura *CSS* é altamente personalizável e de baixo nível que fornece todos os blocos de construção necessários para criar *designs* sob medida, sem que seja necessária a escrita de *CSS* como na abordagem tradicional.

7.3 Objetivo

O *front-end* é responsável por possibilitar a interação do utilizador com a página dentro de uma aplicação *web* e cobrindo as questões da política de privacidade.

7.4 Desenvolvimento

O desenvolvimento do *front-end* começou com o desenho dos *Mockups* no *Figma*, que serviram como base inicial. No entanto, ao longo deste desenvolvimento, o aspeto do *web site* foi divergindo do, inicialmente, desenhado.

A construção, propriamente dita, do *front-end* realizou-se em *ReactJS*, uma vez que alguns dos elementos do grupo já possuíam conhecimentos na utilização desta *framework*. Relativamente à estilização, utilizou-se a *framework TailWindCSS*.

O *front-end* é a componente da plataforma com o qual o utilizador interage, sendo da sua responsabilidade receber os pedidos e comunicá-los ao *back-end*. Como a esmagadora maioria dos utilizadores não apresenta conhecimentos ao nível informático, o *front-end* adquire especial importância, dado que se o aspeto da plataforma não for apelativo, por melhor que seja a sua implementação ao nível lógico, esta não será valorizada pelo utilizador, e o site terá pouca adesão.

Capítulo 8

ComputationalMind

Ao longo deste semestre desenvolvemos a página do *ComputationalMind* até que o produto final fosse semelhante ao que foi idealizado e apresentado nos capítulos anteriores. Em seguida, será demonstrado o funcionamento e onde se podem encontrar todas as funcionalidades implementadas.

8.1 Página inicial

Na página inicial consideramos que o utilizador pode ser um *Player* que apenas usufrui dos jogos disponibilizados na plataforma pelo *Author* que é a entidade que criou o jogo.

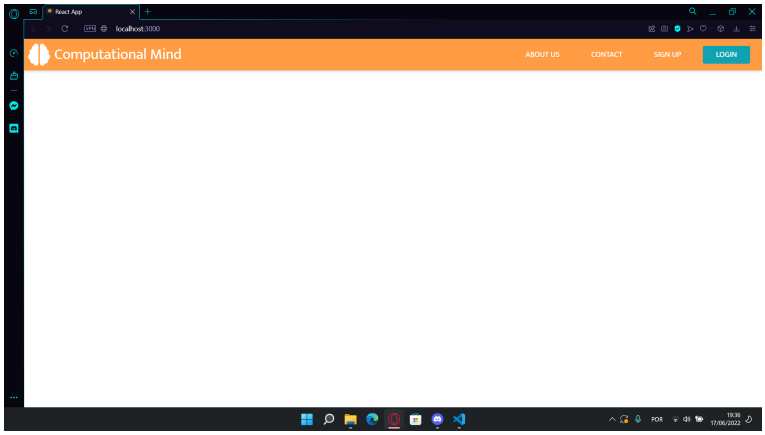


Figura 8.1: Página Principal

8.2 About Us

Nesta secção incorporamos informação sobre o propósito da página.

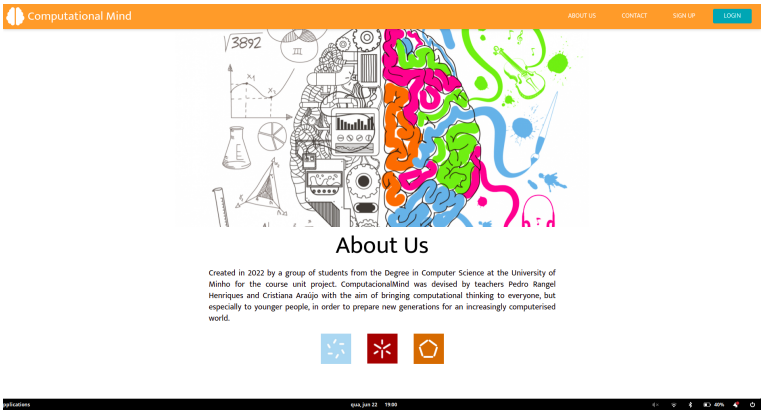


Figura 8.2: About Us

8.3 Contact

Aqui os utilizadores podem informar os *developers* de possíveis *bugs* que tenham encontrado ou sugerir novas funcionalidades que tornariam a plataforma mais apelativa ou cómoda para quem a usa.

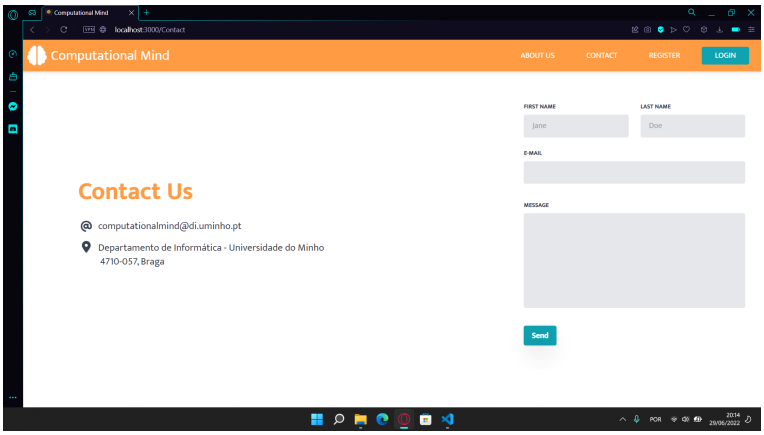


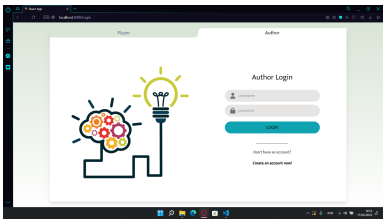
Figura 8.3: Contact

8.4 *Login / Sign Up*

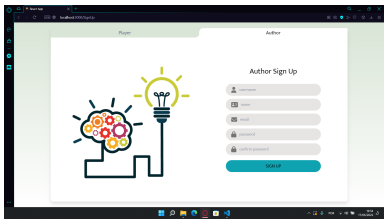
Após a escolha de uma das opções disponibilizadas na página inicial, os utilizadores serão redirecionados para a página de *Login / Sign Up*.

Nesta página é possível um utilizador autenticar-se ou efetuar o seu registo no site.

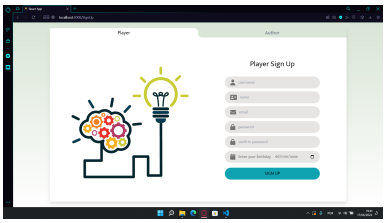
A página de autentificação é igual para ambos os utilizadores, a única diferença na página de *Sign Up* do *Player Sign Up* para a do *Author Sign Up* é que na do *Player* é relevante saber a idade para serem apresentados jogos mais adequados à sua faixa etária enquanto que na do *Author* essa informação não é relevante.



(a) Página de *Login*



(b) Página de *Sign Up Author*



(c) Página de *Sign Up Player*

Figura 8.4: Página de *Login* e Registo

8.5 Menu do *Author*

O *Author* é um utilizador especial, diferindo de um utilizador normal pelo facto de existir uma secção chamada *Creat New Game* onde poderá criar os diferentes tipos de jogos referidos nos capítulos anteriores.

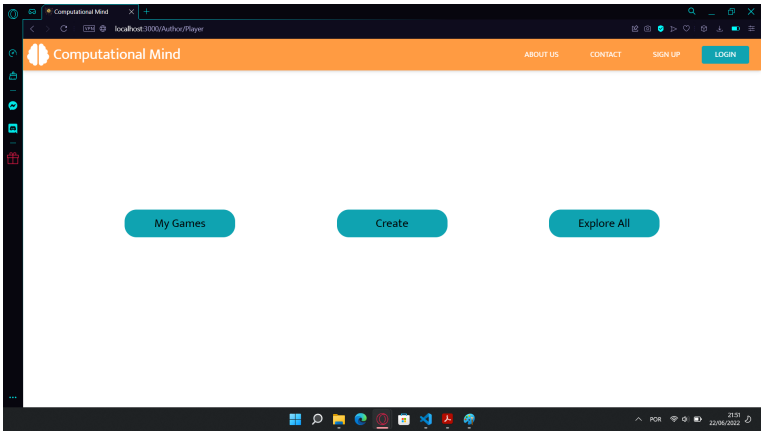


Figura 8.5: Menu do *Author*

8.6 *New Game Menu*

Neste menu oferecemos todas as ferramentas suportadas pela página para que o *Author* consiga criar novo conteúdo para a mesma.

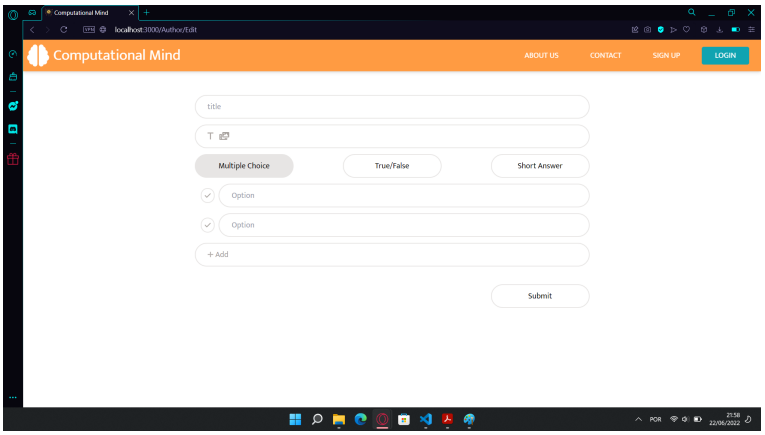


Figura 8.6: Menu para Criação de Jogos pelo *Author*

8.7 Menu de jogos para *Player*

Em seguida, ambos os utilizadores são redirecionados para uma página onde se lista todos os jogos que estão disponíveis para jogar.

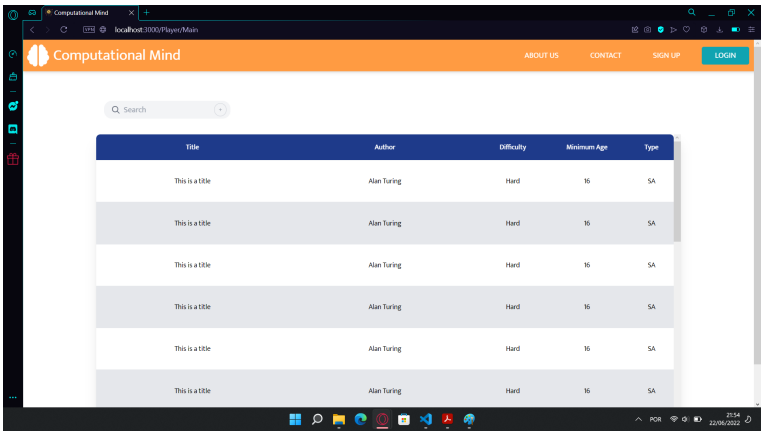


Figura 8.7: Página de Jogos

8.8 Página de Jogo

Nesta página, o *Player* pode fazer a seleção ou a escrita da resposta que pensa ser correta mediante a pergunta enunciada acima.

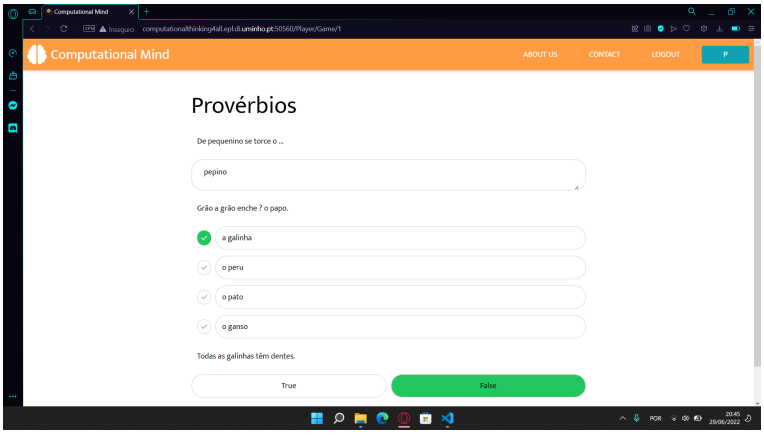


Figura 8.8: Página do Jogo

8.9 Página de *Score*

Após submeter a sua resposta, o *Player* é presenteado com a sua prestação no jogo.

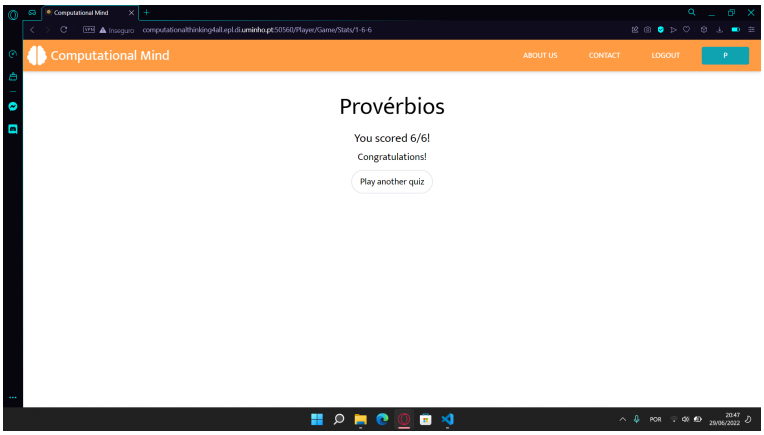


Figura 8.9: Página de *Score*

Capítulo 9

Conclusão

Durante a realização deste trabalho constatamos que a parte fundamental para a implementação de um *web site*, de forma coerente, robusta e segura, assenta em algo mais do que um simples *design* atrativo e um *display* de conteúdos, existindo todo um fluxo e uma rede de comunicação que se dá por trás de cada clique ou de cada palavra digitada e enviada, resultando numa relação causa-efeito.

Relativamente às dificuldades sentidas, o principal desafio foi a utilização de uma linguagem de programação e de ferramentas de desenvolvimento de software completamente novas. Consequentemente, foi necessário um grande investimento da nossa parte com o intuito de nos familiarizarmos com estas e de obtermos um resultado final notável.

Embora tenhamos prestado a devida atenção aos diferentes detalhes, existem, ainda, possíveis melhorias a implementar. Em particular, como o projeto não se encontra totalmente finalizado, consideramos que seria útil a implementação de todas as funcionalidades, que estavam, inicialmente, planeadas e que não estão no produto final. Do mesmo modo, seria igualmente interessante o desenvolvimento de funcionalidades extra que tornassem a página *web* mais apelativa para o utilizador como, por exemplo, a existência de um *ranking* de pontuações adquiridas pelos jogadores nos diferentes quizes, etc.

Em suma, apesar de termos encontrado alguns obstáculos, ao longo deste processo, e de o resultado final ser apenas uma "amostra" daquele que foi inicialmente idealizado, dado que introduzimos apenas as funcionalidades mais importantes, julgamos ter concluído o projeto com relativo sucesso, na medida em que este se tratava de um projeto ambicioso.

Apêndice A

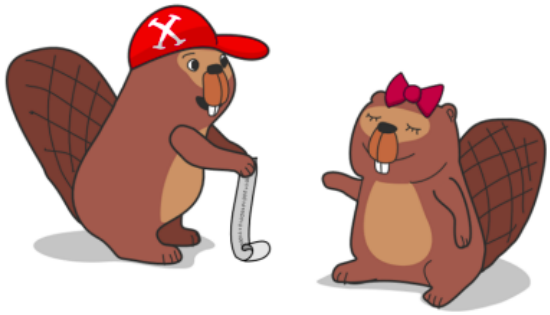
Exemplos

Dificuldade: **difícil** | Origem:

12 – Representação Compacta

O castor Xavier quer representar algumas letras com dígitos binários 0 e 1. Ele repara que as letras T e E são as mais frequentes. Assim, ele decide atribuir-lhes uma representação mais curta e codificar as letras T, E, A, K, C e R como se segue:

Letra	T	E	A	K	C	R
Código	1	00	0010	0110	1010	1110



O Xavier enviou a seguinte mensagem codificada à Ivone:

100100110001010001011 1000

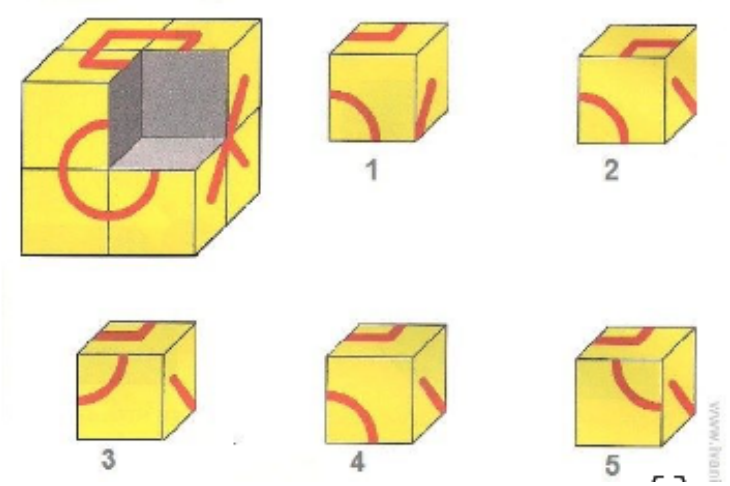
A Ivone já descobriu que a mensagem termina com a letra E.

Pergunta

Em letras (sem espaços a separar), qual é a mensagem completa que o Xavier escreveu?

Resposta

Escreve a tua resposta (uma mensagem).


1
2
3
4
5


1
2
3
4
5
6
7