



Biblioteca Educacional de Códigos de Correção de Erros com Interface Visual

Leandro Jesus Silva

leandro.silva@ua.pt

Departamento de Física & Departamento de Matemática

Universidade de Aveiro

Licenciatura em Engenharia Computacional

Orientador: Maria Raquel Rocha Pinto

Co-Orientador: Zita Abreu

julho 2025

Resumo

Este projeto apresenta o desenvolvimento de uma aplicação web educacional interativa dedicada ao ensino de códigos de correção de erros. A ferramenta foi concebida para transformar conceitos abstratos, frequentemente inacessíveis a alunos do ensino básico e secundário, em experiências visuais, intuitivas e envolventes. Através de simuladores, jogos e questionários, os utilizadores podem explorar diferentes tipos de códigos, como os de repetição e de Hamming, compreendendo o seu funcionamento e aplicabilidade em cenários reais. A aplicação inclui ainda funcionalidades de personalização de perfil, registo de estatísticas e rankings online, promovendo a motivação e o acompanhamento da aprendizagem. Este trabalho alia fundamentos teóricos da teoria da informação a uma abordagem pedagógica moderna, demonstrando o potencial das tecnologias web no apoio ao ensino de conteúdos matemáticos e computacionais.

Palavras-chave: Códigos de correção de erros; Educação interativa; Aplicação web; Aprendizagem autónoma; Plataforma pedagógica.

Conteúdo

1	Introdução	1
2	Preliminares	2
2.1	Códigos Lineares	3
2.2	Codificação e Matriz Geradora	4
2.3	Códigos de Repetição	5
2.4	Código de Hamming (7,4)	6
2.5	Matrizes Geradoras Sistemáticas	7
3	Metodologia	7
3.1	Planeamento Inicial e Definição de Objetivos	7
3.2	Levantamento de Requisitos	7
3.3	Escolha de Tecnologias	8
3.4	Processo de Desenvolvimento	9
4	Tecnologias Utilizadas: Descrição e Justificação Técnica	9
4.1	Arquitetura Tecnológica da Aplicação	10
5	Interface da Aplicação: Estrutura e Funcionalidade	11
5.1	Página Inicial e Navegação	12
5.2	Página — “Onde encontramos os códigos de correção?”	13
5.3	Página — “Explorar Códigos de Correção:”	15
5.3.1	Simulador interativo	17
5.4	Página — “Jogos e Desafios”	19
5.4.1	Jogo 1 — Detetive de Erros	21
5.4.2	Quizes — Avaliação com Competição e Feedback Interativo	23
6	Discussão	25
7	Conclusão	25

1 Introdução

A crescente dependência de sistemas digitais em áreas como a comunicação digital, o armazenamento de dados e a computação distribuída exige mecanismos robustos para assegurar a integridade da informação. Um dos principais desafios nesses sistemas é a presença inevitável de erros durante a transmissão ou armazenamento de dados, devido a ruídos, interferências ou falhas de hardware.

Para mitigar os problemas causados por erros em sistemas digitais, foram desenvolvidos **códigos de correção de erros**, que desempenham um papel essencial na garantia da fiabilidade das comunicações. Estes códigos permitem identificar e corrigir automaticamente falhas introduzidas durante a transmissão, assegurando a recuperação da mensagem original. De forma geral, o processo envolve três etapas principais: primeiro, a codificação, onde a mensagem original, representada por uma sequência de bits, é convertida numa palavra-código com informação redundante. Em seguida, ocorre a transmissão, durante a qual essa palavra é enviada por um canal sujeito a ruídos e interferências. Por fim, dá-se a receção, momento em que a mensagem recebida é verificada quanto à sua integridade e, se necessário, corrigida. Esta abordagem é fundamental para garantir a segurança e a robustez da informação em contextos como redes informáticas, armazenamento digital e sistemas de comunicação sem fios [7, 8].

De acordo com Huffman e Pless [4], os códigos de correção de erros são uma das ferramentas mais eficazes para assegurar a fiabilidade em sistemas digitais modernos.

Entre os mais utilizados encontram-se os *códigos de Hamming*, os *códigos cíclicos* e os *códigos de Reed-Solomon*, com aplicações em áreas como sistemas de satélite, CDs, DVDs, QR codes e comunicações sem fios [7, 8].

Este trabalho insere-se na área da Engenharia Computacional e conjuga conceitos de Álgebra Linear, Teoria da Computação e desenvolvimento de software interativo. Aborda tanto os fundamentos matemáticos dos códigos de correção de erros como a sua implementação computacional, recorrendo a tecnologias modernas de programação web.

Apesar da sua importância, a teoria subjacente a estes códigos pode ser abstrata e desafiante para estudantes e profissionais em formação. A dificuldade em visualizar os processos de codificação, deteção e correção de erros torna o seu ensino menos acessível. Assim, destaca-se a utilidade de ferramentas didáticas que aliem **interatividade, visualização em tempo real e simulação prática** para apoiar a aprendizagem [10, 11].

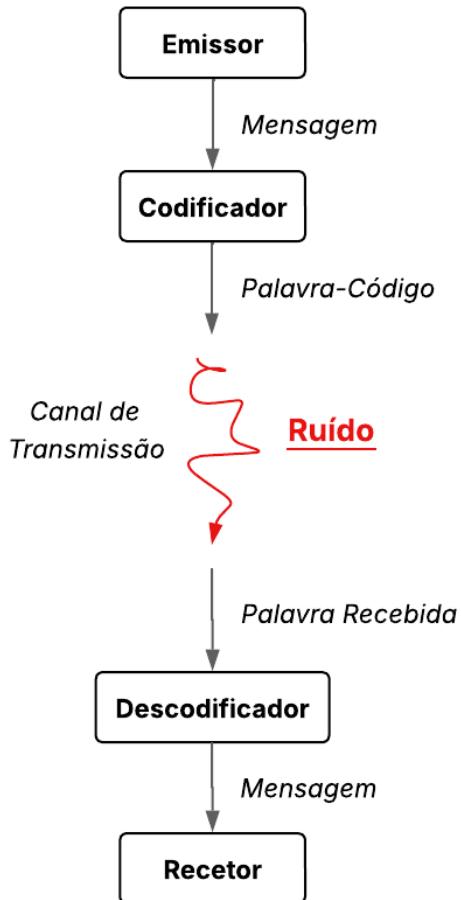
Neste sentido, propõe-se o desenvolvimento de uma **ferramenta educativa baseada na web** para ser utilizada por alunos do ensino básico e secundário. Essa ferramenta permite explorar e compreender os princípios dos códigos de correção de erros através de representações visuais dos processos de codificação e descodificação, simulação de erros e observação das etapas de correção.

Este projeto distingue-se pelo seu caráter inovador, ao combinar rigor matemático com uma abordagem visual e interativa orientada para o ensino. A conceção da ferramenta teve como preocupação central a acessibilidade, procurando adaptar os conceitos abstratos da Álgebra Linear e da Teoria dos Códigos para uma linguagem clara, intuitiva e adequada a estudantes com pouca ou nenhuma formação prévia nestas áreas. Esta tradução conceptual, tanto ao nível da interface como da terminologia utilizada, visa facilitar a compreensão gradual dos princípios fundamentais dos códigos de correção de erros, promovendo uma aprendizagem autónoma, motivadora e inclusiva. Para além disso, houve também o cuidado de utilizar um vocabulário menos formal e mais direto, com expressões mais próximas da linguagem corrente dos estudantes, num tom descontraído e acessível, quase como se estivéssemos a “falar a mesma língua”. O projeto assume assim não só um papel pedagógico, mas também social, ao tornar uma área tradicionalmente técnica mais acolhedora e acessível a um público mais vasto.

Adicionalmente, a acessibilidade e o caráter visual da ferramenta poderão tornar os conteúdos mais atraentes a um público mais diversificado.

A implementação foi realizada com HTML, CSS, JavaScript e Firebase, permitindo simulações visuais interativas no navegador, sem necessidade de instalação.

Hoje em dia, todos os sistemas de comunicação que utilizam dados digitais requerem a utilização de códigos de correção de erros porque todos os canais reais são ruidosos. Para enquadrar de forma intuitiva o funcionamento dos códigos de correção de erros no contexto das comunicações digitais, apresenta-se de seguida um esquema representativo de um sistema de codificação sujeito a ruído. A compreensão deste fluxo é essencial para perceber onde e como os códigos atuam na deteção e correção de erros, permitindo garantir a integridade da informação transmitida.



Componentes do sistema:

- **Emissor** — envia a mensagem original.
- **Codificador** — converte a mensagem numa palavra-código.
- **Canal de Transmissão** — meio por onde passa a informação.
- **Ruído** — interferência que pode afetar a transmissão.
- **Descodificador** — corrige os erros introduzidos na palavra-código durante a transmissão e converte-a de volta em mensagem.
- **Recetor** — recebe a mensagem final.

Figura 1: Esquema do sistema de codificação com canal ruidoso.

Antes de apresentar a implementação da ferramenta, é fundamental compreender os conceitos matemáticos subjacentes à Teoria dos Códigos. Na próxima secção esses conceitos são estudados.

2 Preliminares

Nesta secção vão ser introduzidas noções de Álgebra Linear necessárias à contextualização matemática deste trabalho.

Definição 2.1 ([9]) Um conjunto \mathbb{K} com as operações $+$ e \cdot é um **corpo** se:

- 1) para todos $a, b \in \mathbb{K}$, $a + b = b + a$
- 2) para todos $a, b, c \in \mathbb{K}$, $(a + b) + c = a + (b + c)$
- 3) existe um elemento $u_0 \in \mathbb{K}$ tal que para todo $a \in \mathbb{K}$, $a + u_0 = a$
- 4) para todo $a \in \mathbb{K}$, existe $a' \in \mathbb{K}$ tal que $a + a' = u_0$

- 5) para todos $a, b \in \mathbb{K}$, $a \cdot b = b \cdot a$
- 6) para todos $a, b, c \in \mathbb{K}$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- 7) existe um elemento $u_1 \in \mathbb{K} \setminus \{u_0\}$ tal que para todo $a \in \mathbb{K}$, $a \cdot u_1 = a$
- 8) para todo $a \in \mathbb{K} \setminus \{u_0\}$ existe $\bar{a} \in \mathbb{K} \setminus \{u_0\}$ tal que $a \cdot \bar{a} = u_1$
- 9) para todos $a, b, c \in \mathbb{K}$, $a \cdot (b + c) = a \cdot b + a \cdot c$

Os elementos u_0 e u_1 são normalmente representados por 0 e 1, respectivamente.

As operações sobre corpos finitos são essenciais para a construção de códigos de correção de erros. As propriedades fundamentais destes corpos, encontram-se amplamente descritas na literatura especializada [3, 9]. Neste trabalho vamos considerar apenas o corpo binário \mathbb{F}_2 (ou \mathbb{Z}_2).

Definição 2.2 ([3]) O corpo \mathbb{F}_2 é composto pelos elementos 0 e 1 e pelas operações de soma e multiplicação definidas como se segue:

Soma em \mathbb{F}_2	Multiplicação em \mathbb{F}_2
$0 + 0 = 0$	$0 \times 0 = 0$
$0 + 1 = 1$	$0 \times 1 = 0$
$1 + 0 = 1$	$1 \times 0 = 0$
$1 + 1 = 0$	$1 \times 1 = 1$

Definição 2.3 ([12], [9]) Sejam V um conjunto e \mathbb{K} um corpo. Uma operação interna em V é uma operação “+”, denominada adição, tal que para todos $u, v \in V$ existe um e um só $w \in V$ tal que $u + v = w$. Uma operação externa em V é uma operação “.”, denominada multiplicação escalar, tal que para todo $u \in V$ e $\alpha \in \mathbb{K}$ existe um e um só $w \in V$ tal que $\alpha \cdot u = w$.

O conjunto V munido das operações “+” e “.” é um espaço vetorial sobre \mathbb{K} se:

- 1) para todos $u, v \in V$, $u + v = v + u$
- 2) para todos $u, v, w \in V$, $(u + v) + w = u + (v + w)$
- 3) existe um único elemento $0_V \in V$ tal que para todo $u \in V$, $u + 0_V = u$
- 4) para todo $u \in V$ existe $u' \in V$ tal que $u + u' = 0_V$
- 5) para todos $u, v, w \in V$, $u \cdot (v + w) = u \cdot v + u \cdot w$
- 6) para todos $u, v, w \in V$, $(u + v) \cdot w = u \cdot w + v \cdot w$
- 7) para todos $\alpha, \beta \in \mathbb{K}$ e $u \in V$, $(\alpha\beta) \cdot u = \alpha \cdot (\beta \cdot u)$
- 8) o elemento neutro da multiplicação em \mathbb{K} , $1_{\mathbb{K}}$, é tal que para $u \in V$, $1_{\mathbb{K}} \cdot u = u$

O conjunto \mathbb{F}_2^k é um espaço vetorial sobre o corpo \mathbb{F}_2 porque satisfaz todas as propriedades acima. Este conjunto consiste em todas as sequências de comprimento k cujos elementos pertencem ao corpo \mathbb{F}_2 . A adição de vetores e multiplicação escalar são realizadas componente a componente.

2.1 Códigos Lineares

As transformações lineares são utilizadas neste contexto porque permitem uma descrição eficiente do processo de codificação [7, 8].

Seja $\varphi : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ uma transformação linear. Diz-se que φ é injectiva quando vetores distintos em \mathbb{F}_2^n são transformados em vetores distintos em \mathbb{F}_2^m .

Neste contexto, um código C linear (n, k) pode ser entendido como a imagem de uma transformação linear $\varphi : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ injetiva, isto é,

$$C = \text{Im}(\varphi) \subseteq \mathbb{F}_2^n,$$

ou seja C é um subespaço de \mathbb{F}_2^n .

A transformação linear $\varphi : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ obedece ao Teorema das Dimensões:

$$\dim(\mathbb{F}_2^k) = \dim(\ker(\varphi)) + \dim(\text{Im}(\varphi)),$$

onde

$$\ker(\varphi) = \{x \in \mathbb{F}_2^k : \varphi(x) = 0\}$$

é o núcleo de φ .

Este resultado fundamental encontra-se amplamente estabelecido na literatura sobre Álgebra Linear [13].

Note-se que uma transformação linear φ é injetiva se e só se $\ker(\varphi) = \{0\}$ [13], logo, pelo Teorema das Dimensões, um código linear (n, k) tem dimensão $\dim(\mathbb{F}_2^k) = k$. Obtem-se assim a seguinte definição:

Definição 2.4 ([12]) Um código linear (n, k) é um subespaço de \mathbb{F}_2^n , de dimensão k e diz-se que o código tem taxa de transmissão $\frac{k}{n}$.

2.2 Codificação e Matriz Geradora

Uma matriz geradora de um código linear (n, k) C é uma matriz $G \in \mathbb{F}_2^{k \times n}$ cujas linhas formam uma base de C . Todos os elementos de C são combinação linear das linhas de G e são denominados **palavras-código**.

A codificação de uma mensagem $u \in \mathbb{F}_2^k$ numa palavra-código $v \in \mathbb{F}_2^n$ pode ser realizada por meio de uma matriz geradora $G \in \mathbb{F}_2^{k \times n}$, de seguinte forma:

$$v = u \cdot G.$$

A construção e as propriedades da matriz geradora são amplamente abordadas na literatura sobre códigos lineares [3–6].

Exemplo 2.5 Considere $k = 3$, $n = 4$ e a seguinte matriz geradora:

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}_{3 \times 4}$$

A codificação de uma mensagem $u = [u_1 \ u_2 \ u_3] \in \mathbb{F}_2^3$ é dada por:

$$v = u \cdot G = u_1 \cdot G_1 + u_2 \cdot G_2 + u_3 \cdot G_3,$$

onde G_i representa a i -ésima linha da matriz G . Esta operação gera a palavra-código

$$v = [u_1 \ u_2 \ u_1 + u_2 + u_3 \ u_1 + u_3] \in \mathbb{F}_2^4.$$

Por exemplo, para a mensagem $u = [1 \ 0 \ 1]$ obtém-se a palavra-código

$$\begin{aligned} v &= [1 \ 0 \ 1] \cdot G \\ &= [1 \ 0 \ 1] \cdot \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \\ &= [1 \ 0 \ 0 \ 0]. \end{aligned}$$

Um código C linear (n, k) pode admitir múltiplas matrizes geradoras.

Definição 2.6 Duas matrizes geradoras de um código linear C , dizem-se **matrizes geradoras equivalentes**.

Duas matrizes $G_1, G_2 \in \mathbb{F}^{k \times n}$ são matrizes geradoras equivalentes se e só se

$$G_1 = UG_2,$$

para alguma matriz invertível $U \in \mathbb{F}^{k \times k}$ [3–6].

Dois exemplos de códigos lineares são:

- **Código de Repetição** — cada bit é repetido múltiplas vezes para facilitar a deteção e correção de erros simples.
- **Código de Hamming (7,4)** — código eficiente que permite a correção de 1 erro e deteção de até 2 erros, com mínima redundância. Este tipo de códigos são amplamente utilizados em sistemas de memória (RAM) e comunicações sem fios, pela sua capacidade de correção simples e implementação eficiente [7,8].

2.3 Códigos de Repetição

Um dos códigos mais simples e intuitivos é o **código de repetição**. Este tipo de código baseia-se na repetição de cada bit da mensagem original um número fixo de vezes, com o objetivo de introduzir redundância suficiente para permitir a deteção e, em alguns casos, a correção de erros.

O seguinte exemplo ilustra um código de repetição.

Exemplo 2.7 Consideremos um código de repetição de ordem 3. Cada bit da mensagem é repetido três vezes e a mensagem

$$(1, 1, 0, 1)$$

é transformada na palavra-código

$$(111, 111, 000, 111).$$

Se ocorrer um erro durante a transmissão por exemplo, se um dos bits “0” for recebido como “1” é ainda possível recuperar corretamente a mensagem inicial, aplicando a **regra da maioria** em cada bloco de três bits. Esta regra atribui ao bloco o valor do bit que surge com maior frequência:

$$\text{Recebido: } (111, 111, 001, 111) \Rightarrow \text{Mensagem recuperada: } (1, 1, 0, 1).$$

Observe que o código de repetição de ordem 3 permite corrigir até 1 erro por bloco de três bits. No entanto, esta abordagem implica uma elevada redundância.

Uma **vantagem** deste código foi claramente a sua simplicidade de codificação e descodificação. Contudo, uma **limitação** é o facto de a taxa de transmissão ser muito baixa, neste caso, apenas $\frac{1}{3}$ dos bits transportam informação relevante.

Nos códigos de repetição, a quantidade de redundância necessária para permitir a correção de erros torna-se rapidamente excessiva, reduzindo drasticamente a eficiência da transmissão. Apesar desta limitação, os códigos de repetição são frequentemente utilizados como primeiro exemplo introdutório em Teoria de Códigos, dada a sua clareza conceptual e facilidade de visualização.

Os códigos de repetição são particularmente eficazes quando se trata de canais com erros esporádicos e isolados. A sua capacidade de deteção e correção está limitada a erros muito simples, sendo ineficaz perante erros múltiplos ou sequenciais. Ou seja, este tipo de códigos são pouco eficientes em situações onde a capacidade do canal é limitado ou onde os erros ocorrem em sequências.

Surge então uma questão natural: *será possível construir códigos mais eficientes, que garantam maior fiabilidade na deteção e correção de erros, mas com menos redundância?* A resposta é afirmativa, e um dos exemplos mais clássicos e eficazes nesse sentido é o dos **códigos de Hamming** que serão abordados na próxima secção.

2.4 Código de Hamming (7,4)

Um dos exemplos mais relevantes de códigos lineares são os **códigos de Hamming (7,4)**. De seguida introduzimos algumas definições para conseguirmos estudar com rigor estes códigos.

Distância de Hamming e Peso de Hamming

A noção de **distância de Hamming** formaliza o conceito de “proximidade” entre duas palavras-código e é definida como o número de posições em que dois vetores diferem [5, 7].

Definição 2.8 ([5]) Sejam $x, y \in \mathbb{F}_2^n$. A **distância de Hamming** entre x e y , denotada por $h(x, y)$, é o número de componentes em que x e y diferem:

$$h(x, y) = \#\{i \in \{1, \dots, n\} \mid x_i \neq y_i\}.$$

Esta distância é uma métrica válida, pois satisfaz as seguintes propriedades:

- $h(x, y) \geq 0$ e $h(x, y) = 0 \Leftrightarrow x = y$;
- $h(x, y) = h(y, x)$;
- $h(x, z) \leq h(x, y) + h(y, z)$, para todos $x, y, z \in \mathbb{F}_2^n$.

Esta distância relaciona-se com o **peso de Hamming**, definido como o número de elementos não nulos num vetor $u = (u_1, u_2, \dots, u_n) \in \mathbb{F}_2^n$:

$$\text{wt}(u) = \#\{i \in \{1, \dots, n\} \mid u_i \neq 0\}.$$

Neste contexto, verifica-se que:

$$h(x, y) = \text{wt}(x - y),$$

para todos $x, y \in \mathbb{F}_2^n$.

Distância Mínima e Detecção e Correção de Erros

A **distância mínima** d de um código $C \subseteq \mathbb{F}_2^n$ é definida como a menor distância de Hamming entre quaisquer duas palavra-código distintas:

$$\text{dist}(C) = \min\{h(x, y) \mid x, y \in C, x \neq y\}.$$

Esta distância é fundamental para determinar a capacidade de deteção e correção de erros dum código. O seguinte resultado clássico formaliza essa relação:

Teorema 2.9 ([7,8]) Se um código C tem distância mínima d , então:

- Pode **detectar até $d - 1$ erros**;
- Pode **corrigir até $\lfloor \frac{d-1}{2} \rfloor$ erros**.

Na prática, a descodificação consiste em identificar a palavra-código mais **próxima** da palavra recebida. Em 1950, Richard Hamming introduziu o código de Hamming (7,4). Este código tem matriz geradora

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}_{4 \times 7}$$

e distância mínima $d = 3$, sendo assim capaz de corrigir até 1 erro e detetar até 2 erros.

Como já indicado, estes códigos destacam-se pela sua estrutura simples e capacidade de correção eficiente, sendo aplicados nos contextos tecnológicos mencionados anteriormente. O código de repetição tem também distância mínima 3 e como tal também é capaz de corrigir 1 erro e detetar 2 erros. No entanto tem taxa de transmissão $1/3$ enquanto o código de Hamming (7,4) tem taxa de transmissão $4/7$.

2.5 Matrizes Geradoras Sistemáticas

Todo o código linear (n, k) admite uma matriz geradora sistemática. A forma típica da matriz geradora sistemática é:

$$G = \begin{bmatrix} I_k & \tilde{G} \end{bmatrix},$$

para alguma matriz $\tilde{G} \in \mathbb{F}^{k \times (n-k)}$, a menos de uma permutação de colunas, onde I_k é a matriz identidade de dimensão $k \times k$.

Esta estrutura facilita o processo de codificação e descodificação, bem como a verificação da integridade dos dados, já que cada palavra-código $v \in \mathbb{F}_2^n$ pode ser escrita como

$$v = \begin{bmatrix} u & \tilde{v} \end{bmatrix},$$

onde $u \in \mathbb{F}_2^k$ é a mensagem original e $\tilde{v} \in \mathbb{F}_2^{n-k}$ são os bits de paridade.

Exemplo 2.10 A matriz geradora do código de Hamming (7,4) acima é uma matriz geradora sistemática.

3 Metodologia

Como foi mencionado o desenvolvimento deste projeto seguiu uma abordagem centrada na criação de uma ferramenta interativa com aplicabilidade didática no contexto do ensino de códigos de correção de erros. A metodologia adotada procurou articular uma base matemática sólida com soluções tecnológicas acessíveis, sempre com foco na clareza pedagógica e numa linguagem acessível.

3.1 Planeamento Inicial e Definição de Objetivos

O projeto teve início com a identificação de uma necessidade pedagógica centrada na valorização do papel fundamental da matemática, em particular da Álgebra Linear, no quotidiano e nas tecnologias modernas. Observou-se que muitos estudantes, sobretudo em idades mais jovens, não têm consciência da importância da matemática e da sua aplicação na vida real. Esta ausência de noção sobre a aplicabilidade da matemática pode limitar o interesse e a motivação para a sua aprendizagem, tornando fundamental promover o gosto pela matemática.

O objetivo central consistiu, portanto, no desenvolvimento de uma aplicação web educativa que, através da simulação visual, evidenciasse a presença e a importância da matemática (nomeadamente da Álgebra Linear) nos fundamentos da Teoria de Códigos. Este objetivo foi delineado com base em princípios de aprendizagem ativa, proporcionando ao utilizador experiências de exploração e construção do conhecimento por meio da interação direta com os conteúdos.

Pretende-se, assim, contribuir para uma maior valorização do conhecimento matemático e para o reconhecimento do seu impacto prático em contextos tecnológicos atuais, tornando estes conceitos acessíveis e apelativos a um público jovem e diversificado.

3.2 Levantamento de Requisitos

Foram definidos os seguintes requisitos orientadores para o desenvolvimento da aplicação:

- **Exploração contextual e motivacional:**
 - Explicar de forma apelativa a relevância dos códigos de correção de erros no dia a dia digital, recorrendo a exemplos reais, curiosidades históricas e analogias práticas.
 - Promover a valorização da matemática, em particular da Álgebra Linear.
- **Demonstração e experimentação interativa:**
 - Proporcionar simulações visuais dos principais conceitos, como transmissão de mensagens, introdução de erros e mecanismos de deteção/correção.
 - Permitir ao utilizador experimentar e observar, de modo autónomo, o funcionamento de diferentes tipos de códigos (ex.: códigos de repetição), com feedback imediato.
- **Componentes lúdicas e desafios:**
 - Integrar jogos, quizzes e desafios progressivos para reforçar a aprendizagem e promover o envolvimento ativo dos utilizadores.
 - Incentivar a exploração e a superação de obstáculos através de conquistas e rankings.
- **Acessibilidade e usabilidade:**
 - Garantir uma interface clara, intuitiva e apelativa, adaptada a diferentes faixas etárias e níveis de conhecimento.
 - Assegurar compatibilidade com vários dispositivos (computadores, tablets, smartphones) e navegadores, sem necessidade de instalação.

3.3 Escolha de Tecnologias

A escolha das tecnologias teve como principal objetivo a aprendizagem de novas ferramentas, já que este projeto foi iniciado sem conhecimento prévio em algumas dessas tecnologias, nomeadamente HTML, CSS e JavaScript. Enfrentar esse desafio desde o zero permitiu uma evolução significativa ao longo do seu desenvolvimento.

Além disso, foram considerados critérios como acessibilidade, simplicidade e integração web. Foram utilizadas exclusivamente tecnologias do lado do cliente:

-  **HTML** — para estruturar o conteúdo e organizar a interface.
-  **CSS** — para a definição do estilo visual da aplicação.
-  **JavaScript** — para implementar a lógica interativa da simulação.
-  **JSON** — para armazenar conteúdos dinâmicos (como perguntas e respostas).
-  **Firebase** — como backend em tempo real, permitindo guardar dados remotamente.

Estas escolhas permitiram criar uma aplicação leve, adaptável e funcional em qualquer navegador moderno, sem necessidade de configuração local. A compatibilidade com dispositivos móveis foi considerada desde as primeiras fases do desenvolvimento.

3.4 Processo de Desenvolvimento

O desenvolvimento da aplicação decorreu de forma incremental e modular, tendo cada página ou secção sido implementada individualmente, de modo sequencial. Ou seja, para cada página, foram concluídas as etapas de estruturação (HTML), estilização (CSS) e implementação da lógica interativa (JavaScript) antes de avançar para a seguinte. Este método permitiu testar e validar funcionalidades à medida que eram desenvolvidas, facilitando a identificação e correção de eventuais problemas em cada componente.

O processo global foi dividido em fases principais:

1. **Fase 1:** Estruturação e estilização de cada página da interface com HTML e CSS, incluindo menus, secções de simulação e áreas de interação, desenvolvidas e testadas uma a uma.
2. **Fase 2:** Implementação da lógica interativa em JavaScript para cada página, nomeadamente os processos de codificação, introdução de erros e análise da mensagem recebida, bem como as funcionalidades de correção de erros (por exemplo, com o código de Hamming).
3. **Fase 3:** Organização dos conteúdos pedagógicos (exercícios, perguntas) em ficheiros JSON, com carregamento dinâmico via JavaScript.
4. **Fase 4:** Integração com o Firebase para armazenamento de resultados, estatísticas e perfis de utilizador (incluindo rankings), recorrendo a autenticação simplificada e base de dados em tempo real.

Durante todo o processo, foi utilizado o **Visual Studio Code** como ambiente de desenvolvimento, recorrendo a extensões de pré-visualização e formatação automática de código. Para testar e visualizar a aplicação localmente, utilizei tanto servidores *localhost* como as próprias extensões do editor, garantindo uma verificação rápida e eficiente do funcionamento do site antes da publicação. O controlo de versões e backup do projeto foi assegurado com recurso ao Git e GitHub.

4 Tecnologias Utilizadas: Descrição e Justificação Técnica

O desenvolvimento deste projeto assentou na utilização de tecnologias Web, selecionadas com base na sua simplicidade, eficiência e adequação aos objetivos pedagógicos propostos. A aplicação foi concebida para ser executada diretamente no navegador, dispensando qualquer processo de instalação e garantindo acessibilidade a partir de múltiplos dispositivos.

A estrutura do conteúdo foi construída com recurso a **HTML (HyperText Markup Language)**, tecnologia padrão para a criação de páginas Web. O HTML possibilitou uma organização clara e modular dos diversos componentes da aplicação, como menus, botões, áreas de simulação, perguntas e resultados.

O **CSS (Cascading Style Sheets)** foi utilizado para definir o estilo visual da aplicação, assegurando uma apresentação limpa e coerente em todas as páginas. O CSS permitiu a personalização de cores, tipografia, alinhamentos e layout gráfico, contribuindo para uma interface alusiva e intuitiva, adequada a um público escolar.

A lógica da aplicação foi implementada em **JavaScript**, linguagem de programação executada no lado do cliente. O JavaScript serviu para dinamizar a aplicação, permitindo, por exemplo, simular erros de transmissão, reagir às interacções do utilizador e gerar conteúdos interativos. A utilização desta linguagem contribuiu para uma resposta rápida e fluida, sem necessidade de recarregar a página.

Para a gestão dos conteúdos dinâmicos, nomeadamente das perguntas utilizadas nos jogos, recorreu-se a ficheiros em formato **JSON (JavaScript Object Notation)**. Este formato leve, estruturado e compatível com JavaScript permitiu armazenar listas de perguntas, opções de resposta e outros dados estruturados, facilitando a leitura, edição e reutilização dos conteúdos.

Adicionalmente, foi integrada a plataforma **Firebase**, fornecida pela Google, para assegurar funcionalidades de *backend* como o armazenamento remoto de dados. Através do Firebase, tornou-se possível guardar

resultados dos utilizadores ou estatísticas de utilização de forma persistente e em tempo real, sem necessidade de gerir um servidor externo. Esta integração trouxe vantagens ao nível da escalabilidade, segurança e simplicidade de implementação.

A escolha destas tecnologias permitiu atingir os principais objectivos do projeto: criar uma aplicação leve, modular, de fácil manutenção e que proporcione uma experiência interativa e acessível. A separação clara entre estrutura (HTML), estilo (CSS), lógica (JavaScript) e dados (JSON/Firebase) resultou numa solução pedagógica eficaz, adequada para utilização autónoma.

Durante o desenvolvimento, foram ainda utilizadas ferramentas de apoio como o **Visual Studio Code**, com extensões de pré-visualização e formatação automática de código, que contribuíram para a organização e eficiência do processo.

A escolha por tecnologias baseadas em navegador, sem dependência de instalação local, visa também reduzir a barreira de entrada para estudantes, permitindo o uso em sala de aula, laboratórios escolares ou até em dispositivos móveis. Além disso, a interatividade fornecida pelo JavaScript e a separação de dados em JSON facilitam a personalização do conteúdo para diferentes públicos ou níveis de dificuldade.

O **design adaptável** da aplicação permite a sua utilização tanto em ecrãs de computador como em dispositivos móveis, promovendo a acessibilidade do conteúdo. A combinação de HTML, CSS e JavaScript segue os princípios das boas práticas de desenvolvimento Web modernos.

As tecnologias selecionadas contam ainda com ampla documentação oficial e suporte comunitário, o que facilitou a integração de funcionalidades e a resolução de problemas durante o desenvolvimento.

4.1 Arquitetura Tecnológica da Aplicação

O seguinte diagrama ilustra a interação entre as principais tecnologias utilizadas: o **HTML** fornece a estrutura da interface; o **CSS** estiliza a aparência; o **JavaScript** gera a lógica de interação com o utilizador, recorrendo a dados em **JSON** e comunicando com o backend (**Firebase**) para armazenamento remoto. Toda a aplicação é executada no navegador do utilizador, sem necessidade de instalação local.

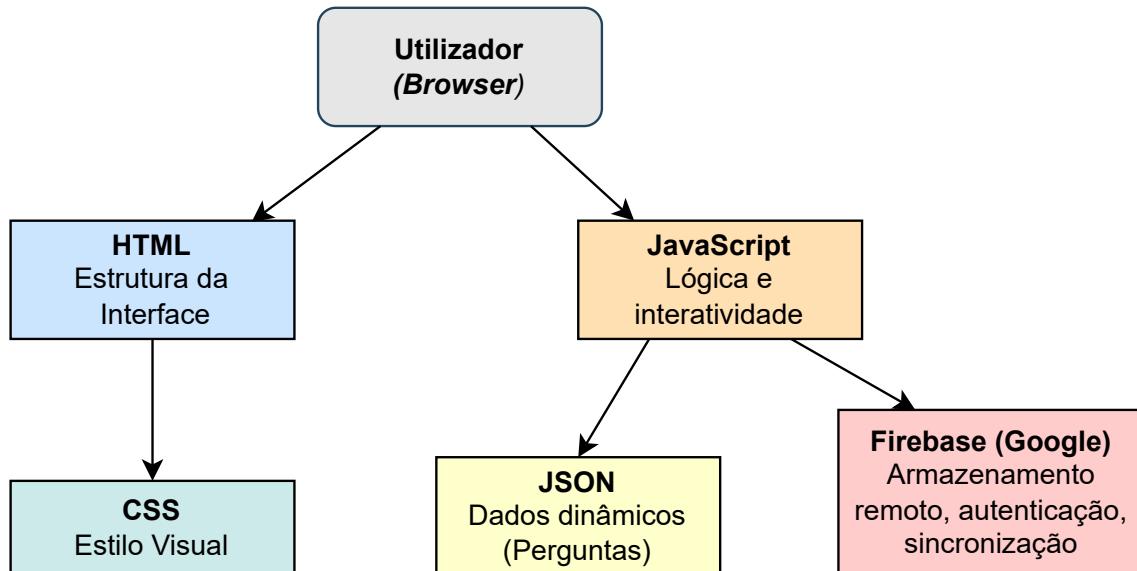


Figura 2: Arquitetura tecnológica da aplicação Web educativa

5 Interface da Aplicação: Estrutura e Funcionalidade

A organização da aplicação foi pensada de forma modular, permitindo uma separação clara entre estrutura, estilo, lógica, conteúdo dinâmico e recursos multimédia. A estrutura de diretórios principal do projeto está representada na Figura 3.

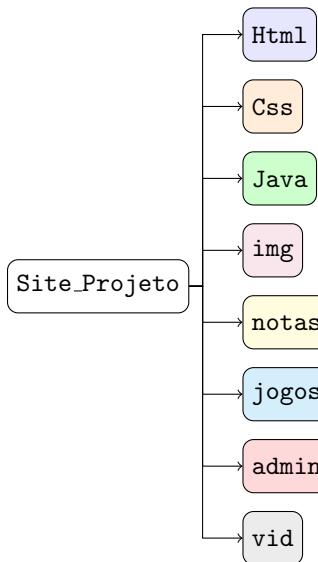


Figura 3: Estrutura de diretórios da aplicação Web desenvolvida

Legenda de cores:

- `Html` — estrutura das páginas web;
- `Css` — estilos visuais da interface;
- `Java` — lógica funcional em JavaScript;
- `img` — recursos gráficos e ícones;
- `notas` — apontamentos e registos;
- `jogos` — módulos interativos e lúdicos;
- `admin` — área de gestão da aplicação;
- `vid` — video representativo duma correção de códigos.

Nota sobre a organização do código

É relevante referir que o desenvolvimento inicial da interface foi realizado com a separação tradicional de ficheiros, utilizando um ficheiro `.html` para a estrutura da página, um ficheiro `.css` para a estilização, e um ficheiro `.js` para a lógica e interatividade. Esta abordagem modular facilitou a organização do código nas fases iniciais de prototipagem e testes individuais de funcionalidades.

Contudo, numa fase mais avançada do projeto, optou-se por consolidar os diferentes componentes num único ficheiro `.html`, incorporando o código CSS e JavaScript diretamente através das tags `<style>` e `<script>`, respetivamente. Esta decisão teve como objetivo simplificar o processo de integração e facilitar a gestão dos conteúdos ao longo do desenvolvimento.

5.1 Página Inicial e Navegação

A página inicial da aplicação, ilustrada na Figura 4, funciona como ponto de entrada para os principais conteúdos interativos. Apresenta um design colorido, apelativo e acessível, pensado para captar a atenção de estudantes do ensino básico e secundário e facilitar a navegação intuitiva entre os vários módulos do projeto.



Figura 4: Página inicial da aplicação Web educativa — menu principal

A estrutura visual da página assenta num cabeçalho com ícones temáticos e título central (*Códigos de Correção de Erros*), seguido por um conjunto de três blocos interativos:

- **Onde encontramos os códigos de correção?** — Secção introdutória que apresenta exemplos do quotidiano onde estes códigos são aplicados.
- **Explorar Códigos de Correção** — Dá acesso a explicações visuais e intuitivas sobre códigos lineares, entre eles os de repetição e de hamming, com exemplos e simulações interativas.
- **Jogos e Desafios** — Permite ao utilizador aplicar os conhecimentos adquiridos através de atividades lúdicas e quizzes com feedback imediato.

Fundo Interativo e Elementos Visuais

Para tornar a página inicial mais dinâmica e apelativa, foi desenvolvido um fundo interativo com base em JavaScript, que combina partículas flutuantes, formas geométricas, linhas de ligação e ondas animadas. Este fundo não serve apenas como elemento decorativo: transmite um ambiente digital, reforçando o enquadramento computacional da aplicação.

Para ilustrar de forma clara a composição visual deste fundo, apresenta-se de seguida um esquema representativo com os seus principais elementos.

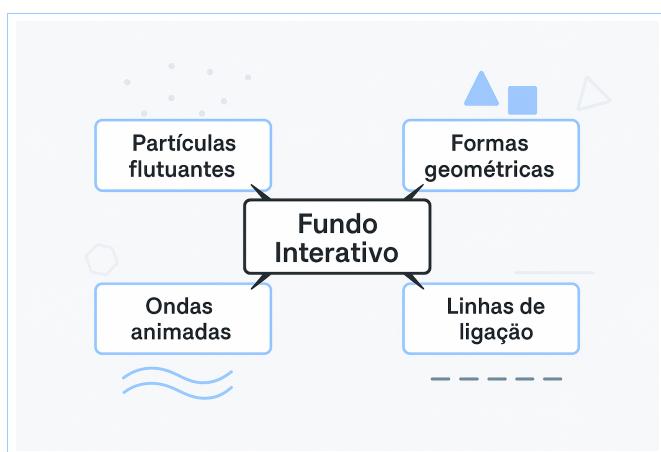


Figura 5: Esquema representativo dos principais elementos visuais que compõem o fundo interativo da aplicação.

A Figura 5 resume visualmente os componentes que contribuem para a identidade gráfica da interface, facilitando a compreensão da estrutura e do efeito visual pretendido.

Estes componentes são criados por funções como `createParticles()`, `createGeometricShapes()` e `createConnectionLines()`, que utilizam o **DOM** para gerar dinamicamente objectos visuais com comportamentos distintos. O resultado é uma interface envolvente, que complementa os conteúdos com uma abordagem gráfica moderna.

No rodapé, encontra-se o nome do autor, tal como a versão atual do site, inclui também 4 botões com ligações ao linkedin, e-mail e github do autor. Também é possível dar o feedback ao site no botão presente no canto inferior direito. Esta disposição reforça a interação com o utilizador e é meramente informativa.

A navegação faz-se através de ligações embutidas em cada secção, proporcionando uma experiência fluida, sem necessidade de recarregar a página, graças à utilização de JavaScript.

Definição 5.1 *O DOM (Document Object Model) é uma interface de programação que representa documentos HTML e XML como uma estrutura em árvore, onde cada nó corresponde a um elemento da página. Esta estrutura permite que linguagens de script, como o JavaScript, acedam, modifiquem ou removam dinamicamente qualquer parte da interface, possibilitando a criação de páginas interativas e reativas.*

5.2 Página — “Onde encontramos os códigos de correção?”

A página “**Onde encontramos os códigos de correção?**” tem como objetivo demonstrar, de forma acessível e cativante, a presença e a importância dos códigos de correção de erros no quotidiano digital. Combinando linguagem informal e envolvente com conteúdo técnico, esta secção foi desenhada para captar a atenção do utilizador desde o primeiro momento.

Mas afinal, porque é que isto interessa? 🤔

Olha, vou ser direto contigo: estes códigos de correção são como os super-heróis invisíveis da tua vida digital. Não os vês, mas estão sempre a trabalhar para ti!

Imagina que estás a ver o teu filme favorito na Netflix e, de repente, a imagem fica toda pixelizada ou o som corta. Chato, não é? Pois é, sem estes códigos, isso aconteceria constantemente. Eles garantem que quando carregas no play, o filme corre lisinho como seda.

E quando pagas com o cartão ou com o MB Way? Estes códigos certificam-se de que os teus dados chegam certinhos ao banco. Senão, imagina o pandemónio: dinheiro a desaparecer, transações a falharem... seria um caos total!

Até quando tiras uma selfie e a guardas no telemóvel, estes códigos estão lá para garantir que não se perde nem um pixel da tua cara bonita. E quando fazes backup das fotos para a nuvem? Lá estão eles outra vez, a certificarem-se de que tudo chega ao destino sem problemas.

A verdade nua e crua é esta: sem códigos de correção, a nossa vida digital seria uma autêntica lotaria. Às vezes funcionava, às vezes não. Mas graças a estes pequenos génios matemáticos, podemos confiar na tecnologia e usá-la sem stress.

Por isso, da próxima vez que vires um código QR meio riscado mas que ainda funciona, ou quando o teu computador não se planta mesmo com aquele disco rígido velhinho, lembra-te: há uns códigos especiais a trabalhar nos bastidores para que tudo corra pelo melhor!

Figura 6: Primeira caixa de texto da página - “Onde encontramos os códigos de correção?”

Exemplos Reais de Aplicação e Curiosidades

A presente página integra diversos elementos, entre os quais se destacam quatro caixas temáticas que apresentam aplicações práticas dos códigos de correção de erros, organizadas por diferentes domínios, conforme ilustrado na Figura 7a. Para além destas quatro caixas, e com o intuito de tornar o conteúdo mais apelativo e captar a atenção dos alunos, foram igualmente integrados balões interativos de curiosidades, com um estilo informal e lúdico. Estes balões apresentam factos históricos e curiosidades acerca da origem e da utilização dos códigos de correção de erros, estando este elemento representado na Figura 7b.

Comunicação

- Satélites & Espaço**
Transmissão sem falhas entre satélites e Terra, mesmo com ruído e distâncias enormes.
- Redes de Telecomunicações**
Chamadas e internet claras e estáveis graças à correção de erros em cada mensagem.
- Internet & Wi-Fi**
Protegem os teus dados enquanto navegas ou fazes downloads, evitando ficheiros corrompidos.

Armazenamento

- CD, DVD & Blu-ray**
Mesmo com riscos, os teus filmes e músicas tocam sem falhas graças à correção de erros.
- Discos SSD & HDD**
Os teus ficheiros ficam seguros mesmo após anos, porque cada bloco de dados é verificado e corrigido.
- Pen Drives & Cartões de Memória**
Evita fotos corrompidas e documentos perdidos em cada gravação.

Multimédia

- Streaming de Vídeo & Música**
Netflix, Spotify, YouTube... imagem e som perfeitos mesmo com internet instável.
- Códigos QR & Barras**
Mesmo danificados, continuam legíveis graças à magia da correção de erros.
- Compressão de Imagem & Áudio**
Fotos e músicas compactas e fáceis ao original, sem perdas inesperadas.

Pagamentos

- Pagamentos Contactless**
O teu cartão ou telemóvel passa rápido e sem erros graças à verificação automática dos dados.
- Códigos de Barras**
Compras no supermercado sem enganos — cada produto é lido corretamente, mesmo com riscos.
- Transações Bancárias**
Transferências, MB Way, pagamentos online... tudo chega certinho ao destino, sem erros nem perdas.

(a) Exemplos reais de aplicação dos códigos de correção.

(b) Balões interativos de curiosidades.

Figura 7: Elementos visuais da página “Onde encontramos os códigos de correção?”

Experiência Interativa com QR Code

Para reforçar a componente prática e visual, foi desenvolvida uma experiência interativa centrada na resistência dos QR Codes. O utilizador pode gerar um código personalizado e simular danos aplicando uma “cortina verde” sobre o mesmo. À medida que cobre o código, é possível observar até que ponto este continua integral.

QR Code Interativo
Testa a magia da correção de erros!

Arrasta a cortina verde
e vê até onde o QR resiste!

Texto para o QR Code:

https://leandrosilva2107.github.io/Site_Projeto/

Gerar QR Code



Arrasta daqui!

Abrir Cortina

Testar QR

Como é que isto funciona?

Olha só que fixe! Podes arrastar a cortina verde para simular um “estrago” (como se fosse um arranhão, sujidade, ou parte rasgada), mas mesmo assim o QR continua a funcionar!

Isto acontece porque:

- **Códigos Reed-Solomon:** Cada QR tem dados extra escondidos que permitem recuperar informação perdida (essa informação é codificada usando um tipo de códigos chamados Reed-Solomon).
- **Diferentes níveis de proteção:** Podes escolher quanta “armadura” o teu QR vai ter.
- **Padrões de localização:** Aqueles quadrados nos cantos ajudam o leitor a orientar-se mesmo estando danificados.
- **Redundância inteligente:** A informação importante está espalhada por todo o código.

Na prática: Isto é super útil quando o QR-Code está numa etiqueta que pode sujar, num poster na rua, ou até tatuado (sim, há pessoal que tatua QR codes! 😱)

Figura 8: Interface interativa de demonstração da robustez de um QR Code com correção de erros.

14

Apesar de parte do código QR ser coberto, este continua integral, algo que só é possível graças à utilização de **códigos de Reed-Solomon**, um tipo de códigos de erros que distribuem de forma inteligente a informação e permitem recuperar dados mesmo com perdas parciais.

Esta atividade proporciona ao utilizador uma compreensão intuitiva dos conceitos matemáticos subjacentes à correção de erros, promovendo ao mesmo tempo o interesse e a experimentação autónoma.

Notas técnicas sobre a implementação

A página inclui uma **cortina interativa**, implementada com um `<div>` posicionado sobre o QR code. A visibilidade da cortina é controlada via `clip-path` em CSS, sendo ajustada dinamicamente em JavaScript consoante o movimento do rato ou toque do utilizador.

O QR code utilizado leva-nos diretamente à página inicial deste mesmo site. O código é funcional e pode ser lido por qualquer dispositivo compatível. Se o utilizador também quiser usar outro QR code poderá fazê-lo inserido o link do mesmo.

Esta interação permite ao utilizador **testar a tolerância a erros** dos QR codes, tapando progressivamente partes do código e observando até que ponto continua legível, ilustrando visualmente a robustez dos mecanismos de correção.

5.3 Página — “Explorar Códigos de Correção:”

Nesta página temos a tela dividida em duas opções, os Códigos de Repetição e Códigos Lineares de Hamming. Quando passamos o cursor por cima duma destas opções a mesma expande-se para conseguirmos ler a sua descrição retornando este efeito:



Figura 9: Página - “Explorar Códigos de Correção:”

Página: Códigos de Repetição

A página dedicada aos **Códigos de Repetição** introduz, de forma acessível, uma das técnicas mais intuitivas de redundância para a deteção e correção de erros. O ecrã apresenta o fluxo completo: codificação, transmissão ruidosa e descodificação por regra de maioria, permitindo ao utilizador acompanhar, passo a passo, a recuperação da mensagem original.

Aprofundando mais, são apresentados de forma cuidada e progressiva os **códigos de repetição com fator 3 e 5**, através de explicações teóricas (Figura 10) acompanhadas de exemplos práticos (Figura 11). O

objetivo é proporcionar ao utilizador uma compreensão sólida dos mecanismos básicos de redundância e da regra de maioria, aplicada à deteção e correção de erros.

São incluídos em cada etapa exemplos ilustrativos. Esta abordagem reforça os fundamentos do processo de correção, preparando o terreno para a compreensão de técnicas mais avançadas apresentadas noutras secções da aplicação.



Figura 10: *Explicação Teórica (Código de Repetição com Fator 3)*

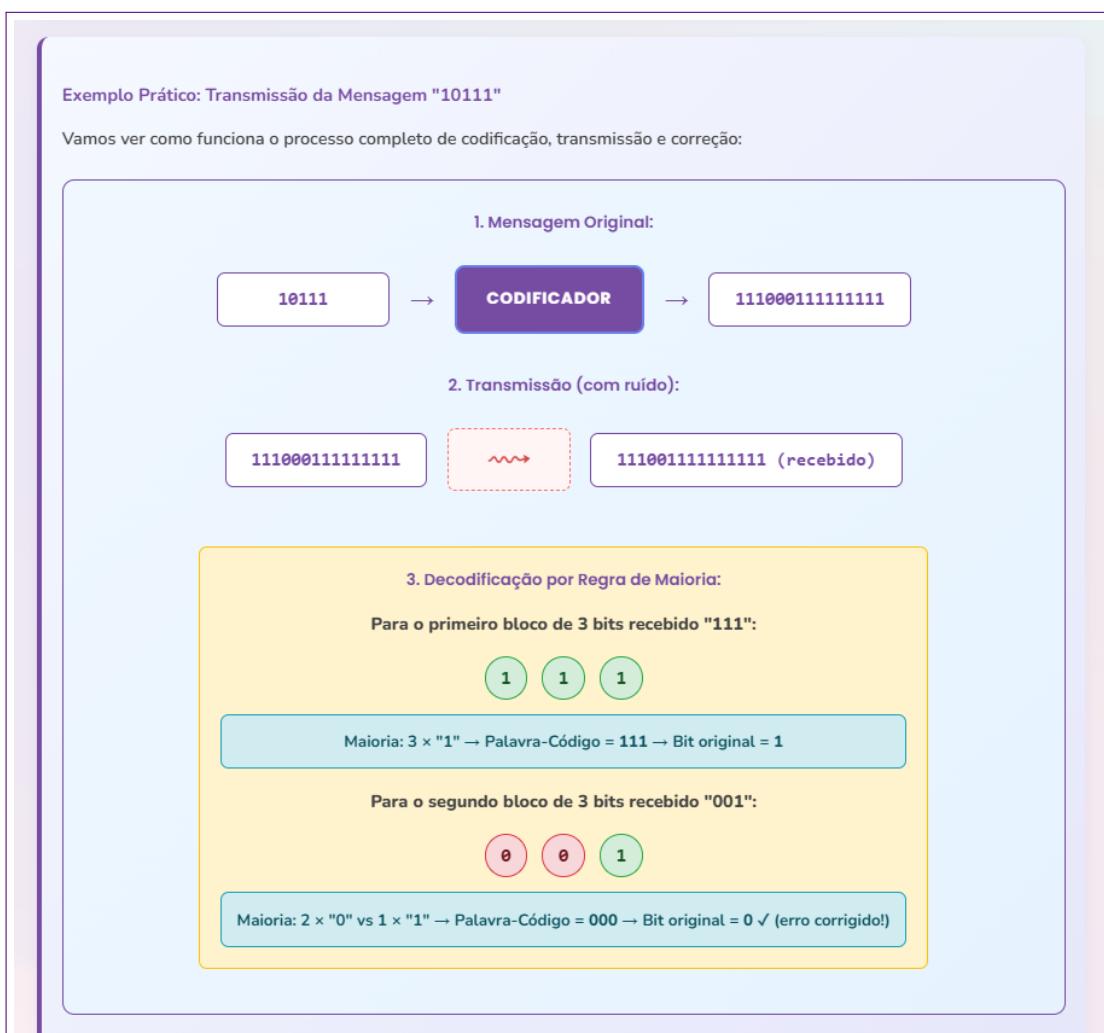


Figura 11: *Exemplo Prático (Código de Repetição com Fator 3)*

5.3.1 Simulador interativo

Nesta página dá-se enfâse ao *Simulador de Códigos de Repetição*, implementado em HTML, CSS e JavaScript (Figura 12).

The screenshot shows the 'Simulador Prático' interface for repetition codes. At the top, there's a header with a logo and the title. Below it, a form for encoding a message:

- Mensagem Original (bits):** 10111
- Fator de Repetição:** 3 (selected via a slider)
- Codificar Mensagem:** Button

Below the form, the process is divided into numbered steps:

- 1 Mensagem Original:** Shows the binary sequence 10111. A note says: "Cada bit será repetido 3 vezes para criar redundância."
- 2 Mensagem Codificada (sequência de palavras-código):** Shows the codified sequence 111 | 000 | 111 | 111 | 111. A note says: "Taxa de transmissão: 5/15 = 33.3%".
- 3 Simulação de Erros de Transmissão:** Contains buttons for "Adicionar Erro Aleatório" (yellow), "Decodificar & Corrigir" (green), and "Reset" (yellow).
- 3 Mensagem Recebida:** Shows the received sequence 111 | 000 | 111 | 111 | 111.
- 4 Resultado da Decodificação:** Shows the decoded message 10111.

At the bottom, a summary box displays:

- Mensagem Original: 10111
- Mensagem Decodificada: 10111
- Status: Correção bem-sucedida!
- Erros Detectados: 0 bloco(s) com erros
- Erros Corrigidos: 0 bloco(s)

Figura 12: Página - *Simulador Códigos de Repetição*

A interface encontra-se organizada em cartões numerados (“Passo 1 — Mensagem Original”, “Passo 2 — Mensagem Codificada”, etc.), orientando o utilizador de forma sequencial.

Numa primeira instância é necessário introduzir a mensagem, depois escolher o fator de repetição e por fim avançar com a codificação. Segue abaixo os detalhes para cada fase.

- **Introdução da mensagem** — campo de texto que aceita apenas dígitos binários, validado através da expressão regular `^ [01] +$`.
- **Escolha do fator de repetição** — controlo deslizante (*slider*) que permite selecionar $n = 3, 5, 7$, com o valor apresentado dinamicamente.
- **Codificação** — a função `encodeMessage()` repete cada bit n vezes e atualiza o painel correspondente.

Para a simulação de erros de transmissão é necessário adicionar os erros (de forma aleatória) e por fim pode-se efetuar a descodificação e/ou correção. Abaixo seguem as especificações destas etapas.

- **Inserção de erros** — o botão Adicionar Erro Aleatório altera um bit numa posição aleatória (`addRandomError()`).

- **Descodificação** — a função `decodeMessage(n)` aplica a regra da maioria, apresenta a mensagem reconstruída, contabiliza os blocos com erros detectados/corrigidos e mostra o resultado com uma mensagem de sucesso ou falha.

O código é modular, facilitando a extensão para fatores de repetição maiores ou para a introdução de erros escolhidos manualmente pelo utilizador.

Observe-se que o simulador reforça a ligação entre a teoria e a prática, permitindo experimentar, em tempo real, os limites de correção dos códigos de repetição. Ao comparar a mensagem original com a descodificada, os estudantes visualizam o efeito da redundância e antecipam a necessidade de códigos mais eficientes, como os de Hamming.

Página: Códigos Lineares de Hamming

Esta secção teve de ser especialmente adaptada a nível técnico para o público-alvo, que normalmente não possui os conhecimentos prévios de Álgebra Linear que seriam necessários para uma introdução dos conceitos da Teoria de Códigos. O conteúdo técnico foi reestruturado com uma linguagem acessível, exemplos visuais e ilustrações que descomplicam o conceito de matriz geradora (Figura 13).

Esta página introduz os fundamentos dos códigos lineares de Hamming. Primeiramente, são apresentados os conceitos de **soma binária (XOR)** (Figura 14) e a ideia central de que a combinação de duas palavras-código através dessa operação resulta sempre numa nova palavra-código.

Ao longo da explicação, é introduzida a **estrutura de um código linear (n, k)**, com ênfase no exemplo do **código de Hamming (7,4)**, sendo explicado o papel dos *bits de paridade* e a forma de os calcular utilizando uma **matriz geradora**. O processo completo de codificação, transmissão com erro e correção, é descrito de forma sequencial e visual.

Matriz Geradora

Como é que vamos formar a palavra-código através dos bits de informação?

Vamos usar um quadro (chamado Matriz Geradora) formado por 0s e 1s, por exemplo:

Matriz Geradora:
$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$

Repara que esta matriz tem 4 linhas e 7 colunas e por isso transforma uma mensagem de 4 bits numa palavra-código formada por 7 bits.

Para calcularmos cada bit da palavra-código vamos considerar a coluna da Matriz Geradora correspondente e somar os bits da mensagem que correspondem a um 1. Por exemplo, vamos calcular a palavra-código associada à mensagem $m = [1011]$.

- Observa a 1º coluna da matriz: [1] [0] [0] [0]

Esta coluna diz-nos que o 1º bit da palavra-código vai ser igual ao 1º bit da mensagem acima [1] [0] [1].

- Observa a 2º coluna da matriz: [0] [1] [0] [0]

Esta coluna diz-nos que o 2º bit da palavra-código vai ser igual ao 2º bit da mensagem acima [1] [0] [1].

- Observa a 3º coluna da matriz: [0] [0] [1] [0]

Esta coluna diz-nos que o 3º bit da palavra-código vai ser igual ao 3º bit da mensagem acima [1] [0] [1].

- Observa a 4º coluna da matriz: [0] [0] [0] [1]

Esta coluna diz-nos que o 4º bit da palavra-código vai ser igual ao 4º bit da mensagem acima [1] [0] [1].

- Observa a 5º coluna da matriz: [1] [0] [0] [1]

Esta coluna diz-nos que o 5º bit da palavra-código vai ser igual à soma dos bits da mensagem nos lugares 1, 2 e 4 (porque são os que têm 1s nesta coluna):

1º bit: 1
2º bit: 0
4º bit: 1
Soma: $1 \oplus 0 \oplus 1 = 1 \oplus 1 = 0$

- Observa a 6º coluna da matriz: [1] [1] [1] [1]

Esta coluna diz-nos que o 6º bit da palavra-código vai ser igual à soma dos bits da mensagem nos lugares 1, 3 e 4:

1º bit: 1
3º bit: 1
4º bit: 1
Soma: $1 \oplus 1 \oplus 1 = 0 \oplus 1 = 1$

- Observa a 7º coluna da matriz: [0] [1] [1] [1]

Esta coluna diz-nos que o 7º bit da palavra-código vai ser igual à soma dos bits da mensagem nos lugares 2, 3 e 4:

2º bit: 0
3º bit: 1
4º bit: 1
Soma: $0 \oplus 1 \oplus 1 = 1 \oplus 1 = 0$

Conclusão

A mensagem 1011 é transformada na palavra-código 1011010.

(a) Explicação - Matriz Geradora (Parte 1)

(b) Explicação - Matriz Geradora (Parte 2)

Figura 13: Explicação - Matriz Geradora



Figura 14: Explicação - *Soma binária (XOR)*

Embora tenha sido inicialmente considerado o desenvolvimento de um **simulador interativo para códigos lineares**, concluiu-se que, para o público-alvo pretendido, a sua implementação exigiria um grau de abstração demasiado elevado ou uma adaptação demasiado extensa para garantir uma percepção clara dos conceitos. Assim, optou-se por uma abordagem mais visual e expositiva.

Abordagem gráfica e didática

Para que os alunos pudessem visualizar melhor os passos da codificação e da deteção de erros, foi criada uma **explicação visual em etapas**, com setas, caixas coloridas e mensagens orientadoras. Toda a explicação evita símbolos matemáticos complexos e prioriza o entendimento sequencial da lógica do código.

Esta página foi uma das mais desafiantes do ponto de vista pedagógico, devido à introdução de conceitos abstratos como vetores, operações binárias e matrizes. Foi necessário simplificar e adaptar substancialmente a linguagem, estrutura e exemplos apresentados.

5.4 Página — “Jogos e Desafios”

Com o objetivo de consolidar os conhecimentos adquiridos e promover a aprendizagem ativa, foi desenvolvida uma secção dedicada a jogos e desafios interativos. Esta componente lúdica assume um papel central na aplicação, proporcionando um ambiente motivador e adaptado a diferentes níveis de dificuldade.

A página “**Jogos e Desafios**” apresenta-se com um visual moderno e apelativo, incluindo um sistema de perfis e conquistas que visa aumentar o envolvimento dos utilizadores. Os jogos foram organizados em blocos temáticos, com identificação gráfica, nível de dificuldade e estimativa de duração. A interface adapta-se automaticamente a diferentes resoluções de ecrã, permitindo o acesso a partir de computadores, tablets ou dispositivos móveis.

Os jogos implementados incluem:

- **Detetive de Erros** — Um desafio introdutório onde o utilizador deve identificar erros em sequências de bits transmitidas. Ideal para consolidar a ideia de deteção de erros.
- **Quiz dos Códigos - Facil** — Questionário com perguntas de escolha múltipla, abrangendo diversos conceitos apresentados na aplicação. Apresenta feedback imediato e reforça a aprendizagem.
- **Quiz dos Códigos - Difícil** — Versão avançada do quiz, com perguntas mais complexas, destinado a utilizadores que já dominam os conceitos fundamentais.

A figura seguinte mostra a disposição dos jogos na interface da aplicação:

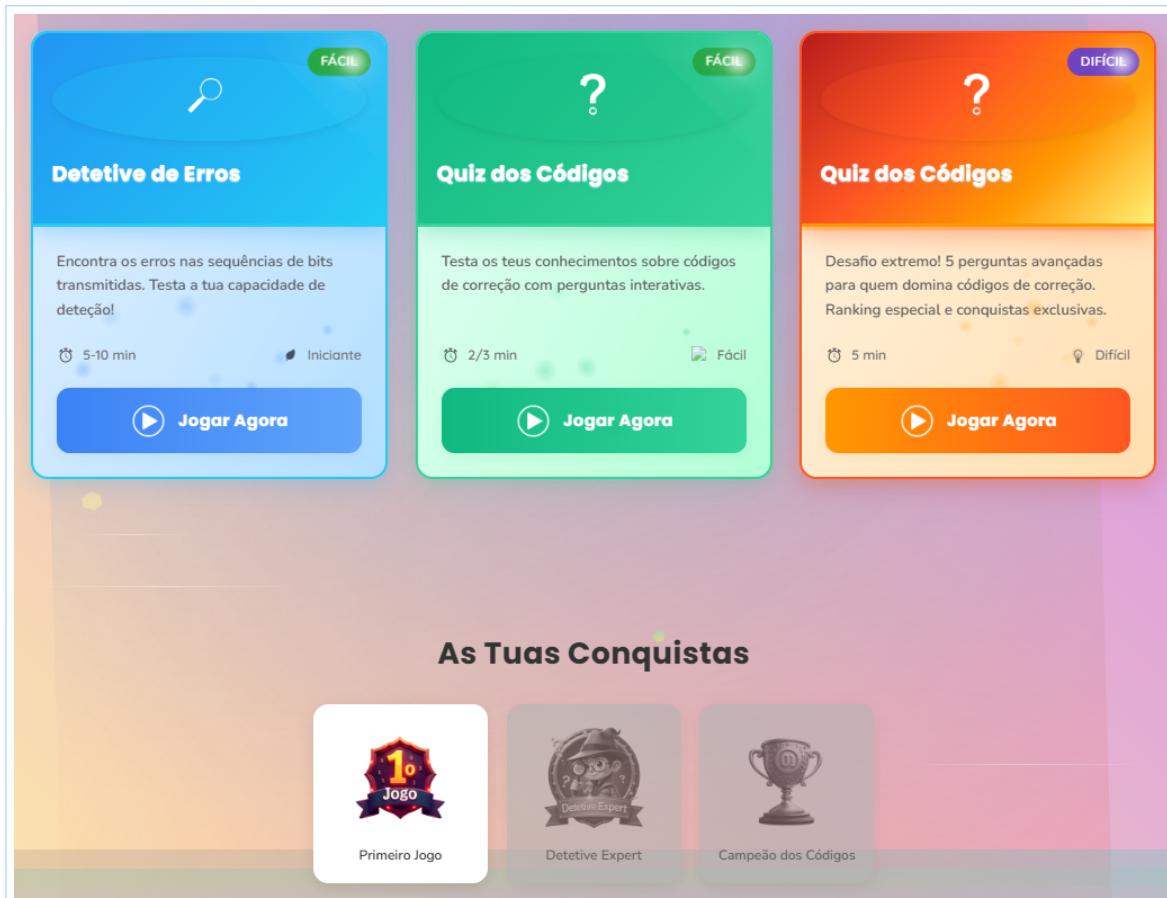


Figura 15: Interface da secção “Jogos e Desafios” com os diferentes módulos interativos

Além dos jogos em si, a aplicação inclui um sistema de perfis personalizáveis, permitindo ao utilizador escolher um nome, avatar e cor temática. Cada perfil guarda estatísticas como número de tentativas, melhor pontuação e conquistas desbloqueadas, promovendo o progresso e a competição saudável.

Estatísticas			
Jogo	Jogadas	Melhor Pontuação	Última Pontuação
Detetive de Erros	0	-	-
Quiz dos Códigos (Fácil)	0	-	-
Quiz dos Códigos (Difícil)	0	-	-

Ainda não há estatísticas para este perfil.

Estatísticas Derivadas

Detetive de Erros: Média: - Taxa de Acerto: -

Quiz dos Códigos (Fácil): Média: - Taxa de Acerto: -

Quiz dos Códigos (Difícil): Média: - Taxa de Acerto: -

Fechar

Estatísticas

Troca Rápida de Perfil

Leandraa0

Teste1

Teste2

Teste3

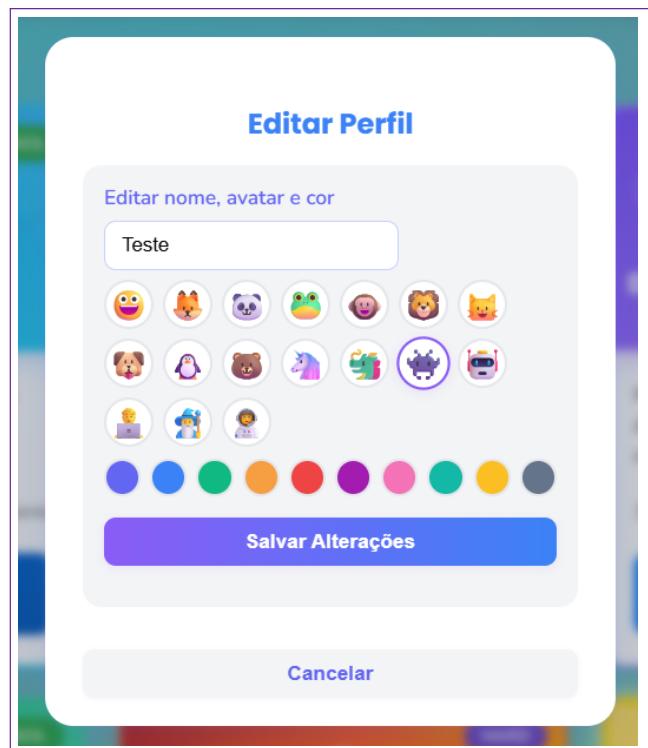
Teste4

Fechar

Troca Rápida do Perfil



Menu do Perfil



Ecrã de Edição de Perfil

Figura 16: Funcionalidades relacionadas com a gestão do perfil do utilizador

Abaixo serão indicadas as funcionalidades dos botões relativos à gestão do perfil.

- “**Ver Perfil**” — Permite observar o perfil criado, assim como outros, e eliminar os mesmos.
- “**Editar Perfil**” — Dá acesso à edição do perfil, conforme ilustrado na imagem da direita (Figura 16).
- “**Estatísticas**” — Apresenta dados como número de tentativas, melhor pontuação, média e percentagem de acerto.
- “**Troca rápida de perfil**” — Permite alternar rapidamente entre perfis criados.
- “**Logout**” — Opção para terminar sessão da conta atual.

Nota sobre o design pedagógico

O uso de mecânicas de jogo e recompensas visuais foi pensado para incentivar o envolvimento dos alunos, em especial os mais jovens. A diversidade de formatos e níveis de complexidade permite adaptar os desafios a diferentes faixas etárias e níveis de conhecimento.

5.4.1 Jogo 1 — *Detetive de Erros*

O primeiro jogo da aplicação é intitulado Detetive de Erros e tem como objetivo principal treinar o reconhecimento de erros em sequências de bits transmitidas. Trata-se de uma atividade lúdica e interativa, que introduz de forma intuitiva o conceito de *ruído em canais de comunicação* e o impacto que este pode ter na integridade dos dados recebidos.

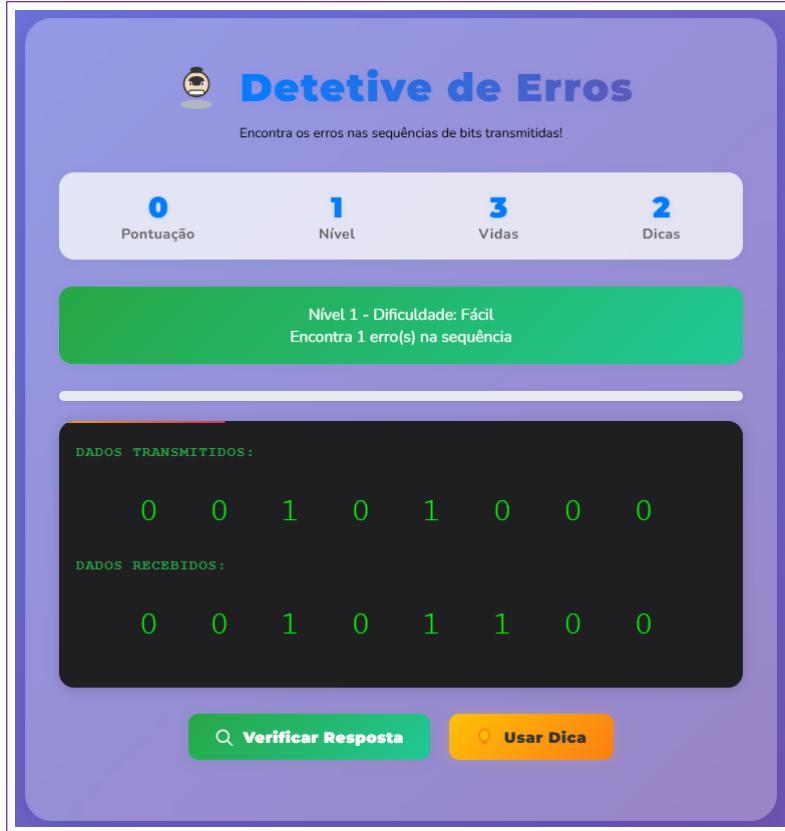


Figura 17: Interface do jogo Detetive de Erros

Neste jogo, o utilizador assume o papel de um detetive digital. A sua missão é identificar, entre duas sequências de bits, a enviada e a recebida, quais foram os bits que foram alterados durante a transmissão. Com uma interface apelativa e dinâmica, o jogo apresenta níveis de dificuldade progressiva e funcionalidades como:

- Indicação de pontuação, nível atual, número de vidas e dicas disponíveis;
- Sequências de bits que variam em comprimento e número de erros;
- Feedback visual imediato para erros encontrados e tentativas incorretas;
- Utilização de dicas para destacar possíveis erros;
- Sistema de progressão com vários níveis até ao término do jogo.

A cada nível, o jogador observa uma sequência transmitida e uma sequência recebida com alguns erros simulados. O utilizador deve selecionar as posições onde acredita que ocorreram erros, e o jogo verifica se a resposta está correta.

Esta atividade reforça conceitos importantes como:

- Noção de erro bit a bit;
- Importância da deteção de erros em comunicações digitais.

No final do jogo, é apresentada uma mensagem personalizada em função da pontuação obtida, promovendo o reforço positivo e incentivando a repetição da atividade.

A pontuação final, bem como as estatísticas de desempenho (tentativas, melhor pontuação, média, etc.), são guardadas no perfil do utilizador sendo armazenadas no localStorage, promovendo um acompanhamento contínuo da evolução.

5.4.2 Quizes — Avaliação com Competição e Feedback Interativo

A aplicação desenvolvida inclui uma secção de **quizes interativos** cujo principal objetivo é avaliar, consolidar e motivar a aprendizagem dos conteúdos relacionados com códigos de correção de erros. Foram desenvolvidos dois quizzes com diferentes níveis de dificuldade (fácil, difícil), no entanto, a estrutura funcional é idêntica em todos, variando apenas o conteúdo das perguntas. Quanto ao aspeto visual cada página de quiz tem cores e elementos diferentes alusivas á dificuldade das mesmas como é possível observar na Figura 18.

Quiz Fácil

Quiz Difícil

Figura 18: Exemplos da interface do quiz nos níveis Fácil e Difícil

Funcionamento Geral O quiz de dificuldade fácil e difícil são compostos por **5 perguntas**, selecionadas aleatoriamente a partir de um ficheiro externo em formato JSON. As perguntas são de escolha múltipla e abrangem várias categorias associadas ao tema. A cada pergunta, o utilizador , na dificuldade fácil, dispõe de **30 segundos para responder**, e na dificuldade difícil 1 minuto. Nestes 2 quizzes é possível ganhar pontos com base em:

- Tempo restante no momento da resposta
- Grau de dificuldade da pergunta
- Número de respostas certas consecutivas (*streak*)

Sistema de Feedback e Rankings Após cada resposta, o utilizador recebe **feedback visual imediato** indicando se a resposta foi correta ou não. No final do quiz, é apresentado um **resumo estatístico** com:

- Pontuação total
- Número de acertos
- Percentagem de precisão
- Posição no ranking diário e geral

Estes rankings são geridos e sincronizados com o **Firebase**, permitindo uma componente de **competição entre utilizadores**. É importante que referir que cada quiz ou melhor cada dificuldade tem o seu próprio ranking.

Perfis e Estatísticas A funcionalidade de perfis permite que cada utilizador tenha o seu próprio histórico e estatísticas. O sistema armazena localmente (via `localStorage`) e também na base de dados remota:

- Melhor pontuação
- Pontuação média
- Número de jogos jogados
- Histórico completo com possibilidade de **revisão detalhada de cada pergunta**

A aplicação recorre ao Firebase para armazenar e sincronizar os dados de desempenho dos utilizadores, garantindo persistência entre sessões e facilitando a construção de rankings dinâmicos. A estrutura abaixo apresenta um exemplo real de um segmento do registo de um utilizador na base de dados, incluindo a sua pontuação, percentagem de acertos, data, e detalhes de cada pergunta respondida:

```
rankings
  └── geral
      ├── OTDEA4JRcCwUpLPxk0
      ├── accuracy: 20
      ├── correctAnswers: 1
      ├── date: 2025-06-25T22:39:25.313Z
      ├── name: Leandras0
      ├── questions: 5
      └── questionsDetails
          ├── 0
          │   ├── correct: 0
          │   ├── options
          │   │   ├── 0: Repete cada bit várias vezes para detectar e corrigir erros
          │   │   ├── 1: Codifica números binários em hexadecimal
          │   │   ├── 2: Aumenta a compressão dos dados
          │   │   ├── 3: Remove bits redundantes
          │   │   ├── question: 0 que faz um código de repetição
          │   │   ├── selected: 0
          │   └── 1
          └── 2
          └── ...
      └── score: 16
```

Aspectos Visuais Para manter o envolvimento, o quiz conta com uma interface moderna e animada: gradientes dinâmicos, partículas em movimento e introdução personalizada com nome do jogador. Tudo isto contribui para uma experiência mais lúdica e acessível. Em cada quiz a interface tem em conta com a dificuldade do nível. No caso do quiz de dificuldade fácil deparamo-nos com uma interface minimalista e simples.

No quiz difícil existem tons mais vermelhos e alaranjados algo como “fogo” e “chamas” para representar algo mais quente, infernal.

Nota

Esta funcionalidade foi pensada de forma a equilibrar simplicidade visual com profundidade técnica adequada ao público-alvo, mas suficientemente robusta para registar dados, calcular estatísticas e integrar rankings online.

6 Discussão

O desenvolvimento deste projeto colocou desafios significativos na conciliação entre a complexidade técnica dos códigos de correção de erros e a necessidade de comunicar estes conceitos de forma acessível a estudantes com pouca ou nenhuma experiência prévia na área. A principal aprendizagem consistiu, assim, em perceber que a eficácia pedagógica depende tanto da clareza visual e narrativa como do rigor dos conteúdos.

A escolha de uma abordagem baseada na interatividade e na experimentação revelou-se crucial para envolver os utilizadores e facilitar a assimilação dos conceitos. Elementos como simuladores, quizzes e jogos deixaram de ser meros complementos, passando a assumir um papel central no modelo de ensino adoptado.

Durante o processo de implementação, surgiram decisões técnicas relevantes relacionadas com o desempenho, a compatibilidade e a escalabilidade. A opção por consolidar a aplicação num único ficheiro HTML, com estilos e scripts incorporados, permitiu simplificar o processo de integração e distribuição, tornando o produto final mais estável e acessível em diferentes dispositivos e navegadores.

Outro ponto de reflexão resultou da necessidade de ajustar o nível de abstração dos conteúdos. Em tópicos como os códigos lineares ou a matriz geradora, tornou-se evidente que uma exposição gradual, com forte apoio visual e exemplos guiados, é essencial para garantir a compreensão e evitar a sobrecarga cognitiva.

Por fim, destaca-se a importância da validação contínua junto da equipa orientadora. As reações positivas e as sugestões recolhidas durante as reuniões permitiram refinar a experiência e adicionar melhorias ao produto.

Em síntese, a construção desta aplicação demonstrou que aproximar a matemática aplicada dos alunos exige não apenas boas explicações, mas também boas experiências. O equilíbrio entre interatividade, clareza e desafio foi o fio condutor de todo o projeto.

7 Conclusão

O desenvolvimento desta aplicação demonstrou que é possível transformar conteúdos tradicionalmente complexos como os códigos de correção de erros em experiências educativas acessíveis, interativas e motivadoras. O projeto não se limitou à criação de uma ferramenta funcional, mas construiu uma ponte entre a teoria e a prática, entre o rigor matemático e a aprendizagem visual.

A estratégia seguida privilegiou uma abordagem centrada no utilizador, apoiada em elementos gráficos, interativos e lúdicos. Ao longo do percurso, cada decisão técnica, estética ou pedagógica contribuiu para consolidar uma plataforma coesa, ajustada aos objetivos propostos e sensível às necessidades do público-alvo.

Mais do que cumprir um plano de trabalhos, esta aplicação constitui um exemplo de como a tecnologia pode potenciar o ensino de tópicos especializados em contextos escolares. A articulação entre jogos e simuladores permitiu não só reforçar a compreensão dos conteúdos, mas também fomentar o envolvimento e a autonomia dos utilizadores.

Embora esta etapa esteja concluída, o projeto permanece aberto à evolução. A flexibilidade da arquitetura e a receptividade observada nos testes apontam para um forte potencial de crescimento, tanto na diversificação

dos conteúdos como na sua adaptação a novos públicos ou contextos educativos.

Em suma, a aplicação atingiu os objetivos delineados, demonstrando que o ensino de códigos de correção de erros pode ser não só possível, mas também apelativo, dinâmico e eficaz. Fica assim lançado o desafio para futuras explorações nesta interseção entre educação, interatividade e Teoria da Informação.

Referências

- [1] Códigos de Correção de Erros — Site do Projeto. Disponível em:
https://leandrosilva2107.github.io/Site_Projeto/
- [2] Repositório do Site do Projeto. Disponível em:
https://github.com/leandrosilva2107/Site_Projeto
- [3] R. Hill, *A First Course in Coding Theory*, Oxford University Press, 1986.
- [4] W. C. Huffman, V. Pless, *Fundamentals of Error-Correcting Codes*, Cambridge University Press, 2003.
- [5] S. Lin, D. J. Costello, *Error Control Coding*, Pearson, 2004.
- [6] F. J. MacWilliams, N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, 1977.
- [7] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003.
- [8] E. R. Berlekamp, *Algebraic Coding Theory*, World Scientific, 2015.
- [9] R. Lidl, H. Niederreiter, *Finite Fields*, Cambridge University Press, 1997.
- [10] A. Kerren, A. Ebert, J. Meyer (Eds.), *Human-Centered Visualization Environments*, Springer, 2007.
- [11] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, 1980.
- [12] S. Axler, *Linear Algebra Done Right*, Springer, 2015.
- [13] L. W. Johnson, R. D. Riess, J. T. Arnold, *Introduction to Linear Algebra*, 5^a ed., Pearson, 2009.