

Universidade da Beira Interior

Departamento de Informática



Departamento de
Informática

C-Team: *Challenge-Accepted*

Elaborado por:

38950 — Diogo José Real Lavareda
39392 — Joana Elias Almeida
41266 — Diogo Castanheira Simões
41358 — Beatriz Tavares da Costa
41381 — Igor Cordeiro Bordalo Nunes

Orientador:

Professor Doutor Pedro Ricardo Morais Inácio

22 de maio de 2021

Resumo

A criptografia é célebre pelas suas práticas e técnicas de comunicação segura, contudo a mesma não é usada apenas na iminência de inimigos. Sendo que na era atual a criptografia já se encontra um pouco por todo o nosso quotidiano, porque não usar as suas ferramentas para o nosso lazer?

Sendo este um tema ligado à criptografia, escolhemos abordá-lo de forma informativa e acessível para que o utilizador tenha uma experiência segura ao desfrutar do seu tempo.

Nesta medida, foi criada uma plataforma denominada *Challenge-Accepted*, que oferece a todos os seus utilizadores a oportunidade de desenvolver desafios criptográficos e também de resolver os deixados por outros utilizadores.

Conteúdo

Conteúdo	ii
Lista de Figuras	iv
Lista de Tabelas	v
1 Introdução	1
1.1 Descrição da proposta	1
1.2 Constituição do grupo	1
1.3 Organização do Documento	2
2 Engenharia de Software	3
2.1 Introdução	3
2.2 Ferramentas e tecnologias utilizadas	3
2.3 Requisitos	3
2.3.1 Requisitos funcionais	4
2.3.2 Requisitos não-funcionais	5
2.4 Casos de Uso	5
2.5 Modelo relacional da Base de Dados	5
2.6 Arquitetura do sistema	8
2.7 Conclusões	8
3 Implementação	10
3.1 Introdução	10
3.2 Interface — <i>Text-based User Interface</i> (TUI)	10
3.3 Escolhas de Implementação	11
3.4 Detalhes de Implementação	11
3.4.1 Navegação	11
3.4.2 Estruturação dos <i>packages</i> e classes	12
3.4.3 Cifras e algoritmos de <i>hash</i>	12
3.4.4 <i>Webservice</i>	12

3.4.5	Serviço de autenticação de clientes	13
3.5	Manual de Instalação	13
3.6	Manual de Utilização	14
3.7	Conclusões	14
4	Reflexão Crítica e Problemas Encontrados	15
4.1	Introdução	15
4.2	Objetivos Propostos vs. Alcançados	15
4.3	Divisão do Trabalho pelos Elementos do Grupo	17
4.4	Problemas Encontrados	17
4.5	Reflexão Crítica	17
4.5.1	Pontos Fortes	18
4.5.2	Pontos Fracos	18
4.5.3	Ameaças	18
4.5.4	Oportunidades	18
4.6	Conclusões	18
5	Conclusões e Trabalho Futuro	19
5.1	Conclusões Principais	19
5.2	Trabalho Futuro	19
	Bibliografia	20

Lista de Figuras

2.1	Diagrama de casos de uso: processo de registo e <i>Login</i>	6
2.2	Diagrama de casos de uso: <i>homepage</i>	6
2.3	Diagrama de casos de uso: processo de propor desafios	7
2.4	Diagrama de casos de uso: processo de responder a desafios	7
2.5	Modelo relacional	8
2.6	Diagrama da arquitetura do sistema	9
3.1	Falha de autenticação	13

Lista de Tabelas

1.1	Constituição da equipa <i>C-Team</i>	1
2.1	Ferramentas utilizadas	4
3.1	Cores por tipo de mensagem	11
4.1	Objetivos propostos vs. alcançados	16
4.2	Distribuição de tarefas	17

Acrónimos

AES	<i>Advanced Encryption Standard</i>
BD	Base de Dados
eduroam	<i>Education Roaming</i>
FQDN	<i>Fully Qualified Domain Name</i>
GUI	<i>Graphical User Interface</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
JSON	<i>JavaScript Object Notation</i>
RSA	<i>Rivest-Shamir-Adleman</i>
SSL	<i>Secure Sockets Layer</i>
SWOT	<i>Strength, Weakness, Opportunity, and Threat Analysis</i>
TUI	<i>Text-based User Interface</i>
UBI	Universidade da Beira Interior
URL	<i>Uniform Resource Locator</i>

Capítulo 1

Introdução

1.1 Descrição da proposta

A criptografia é uma das estratégias mais importantes para garantir segurança de dados. Atualmente enfrentamos uma era de globalização com ataque a sistemas para roubo de informações privilegiadas, com o intuito de proveito próprio ou venda das mesmas a terceiros, entre outros fins.

Sendo uma temática tão importante e atual, pretende-se levar a mesma ao público geral e poder transmitir a “magia” dos desafios criptográficos.

A aplicação *Challenge-Accepted* tem como objetivos permitir aos utilizadores registados publicar e resolver desafios criptográficos, bem como tornar-se numa ferramenta educativa na área da criptografia.

1.2 Constituição do grupo

O presente projeto foi realizado pela equipa *C-Team*, constituída pelos elementos listados na Tabela 1.1.

Nº	Nome	Alcunha
38950	Diogo José Real Lavareda	<i>Lavareda</i>
39392	Joana Elias Almeida	<i>Joaninha</i>
41266	Diogo Castanheira Simões	<i>Ash</i>
41358	Beatriz Tavares da Costa	<i>Bea</i>
41381	Igor Cordeiro Bordalo Nunes	<i>Etileno</i>

Tabela 1.1: Constituição da equipa *C-Team*.

1.3 Organização do Documento

De modo a refletir o projeto realizado, este relatório encontra-se estruturado em cinco capítulos, nomeadamente:

1. No primeiro capítulo — **Introdução** — são apresentados o projeto, os seus objetivos, a equipa desenvolvedora e a respetiva organização do relatório.
2. No segundo capítulo — **Engenharia de Software** — são elaborados os diagramas de casos de uso da aplicação que orientam a respetiva implementação.
3. No terceiro capítulo — **Implementação** — são descritas as escolhas e os detalhes de implementação da aplicação, bem como as tecnologias utilizadas durante o seu desenvolvimento.
4. No quarto capítulo — **Reflexão Crítica e Problemas Encontrados** — são indicados os objetivos alcançados, quais as tarefas realizadas por cada membro do grupo, assim como são expostos os problemas enfrentados e é feita uma reflexão crítica sobre o trabalho.
5. No quinto capítulo — **Conclusões e Trabalho Futuro** — são analisados os conhecimentos adquiridos ao longo do desenvolvimento do projeto e, em contrapartida, o que não se conseguiu alcançar e que poderá ser explorado futuramente.

Capítulo 2

Engenharia de Software

2.1 Introdução

Foram primeiramente delineados vários pontos fulcrais, em particular:

- Ferramentas e tecnologias (Secção 2.2): perante uma equipa de 5 pessoas, foi vital determinar quais as tecnologias a utilizar não só para implementar a aplicação mas também para gerir o trabalho paralelo que iria decorrer;
- Requisitos funcionais e não-funcionais (Secção 2.3): a aplicação deve cumprir uma série de requisitos a fim de poder alcançar os objetivos propostos;
- Diagramas de casos de uso (Secção 2.4): a fim de se perceber as atividades a desenhar e o respetivo código-fonte que as interliga, diferentes casos de uso foram estudados.

2.2 Ferramentas e tecnologias utilizadas

As ferramentas utilizadas no âmbito da realização do projeto, sumariadas na Tabela 2.1, visam três componentes essenciais na sua gestão: 1) linguagem de programação, 2) servidor, 3) administração da Base de Dados, 4) controlo de versões.

2.3 Requisitos

De forma a ir de encontro aos objetivos propostos do projeto (Secção 1.1), uma série de requisitos funcionais e não-funcionais foi delineada.

<i>Software</i>	<i>Versão</i>
Linguagem de Programação	
<i>Python</i>	3.8.0 a 3.9.5
Servidores	
<i>Microsoft Windows Server</i>	2019
<i>MariaDB</i>	10.4.18
<i>Flask</i>	2.0.0
Administração Base de Dados (BD)	
<i>DBeaver</i>	21.0.5
Controlo de versões	
<i>git</i>	2.17.1
<i>GitKraken</i>	7.6.1

Tabela 2.1: Ferramentas e tecnologias utilizadas, organizadas por categoria.

2.3.1 Requisitos funcionais

A plataforma deve:

1. Possuir um ecrã de boas-vindas e instruções de uso do sistema;
2. Ter um ecrã de registo e *login* de utilizadores;
3. Ter uma *homepage* com acesso direto às seguintes funcionalidades:
 - (a) Resolução de desafios disponíveis;
 - (b) Propor desafios;
 - (c) Informações sobre a aplicação;
 - (d) *Scoreboard* dos utilizadores;
4. Permitir responder a desafios com as seguintes cifras/algoritmos:
 - (a) Cifra de César [1];
 - (b) El Gamal [2];
 - (c) Vigenere [3];
 - (d) *One Time Pad* [4];
 - (e) AES-128-CBC [5, 6];
 - (f) AES-128-CTR [5, 6];

(g) AES-128-ECB [5, 6];

(h) MD5 [7];

(i) SHA256 [7];

(j) SHA512 [7];

5. Permitir submeter dois tipos de desafios, em que um é de decifra de mensagem e outro de descoberta de mensagem.

2.3.2 Requisitos não-funcionais

A plataforma deve:

1. Permitir apenas uma tentativa a cada 15 segundos para os desafios cifrados;
2. Impedir a resolução de desafios propostos pelo próprio utilizador;
3. Ter uma *Text-based User Interface* (TUI) minimalista e *user-friendly*;
4. Ser segura em termos do armazenamento dos dados na base de dados (se possível cifrar os dados guardados duplamente no caso dos desafios);
5. Apenas permitir *passwords* fortes [8] e *emails* válidos [9].

2.4 Casos de Uso

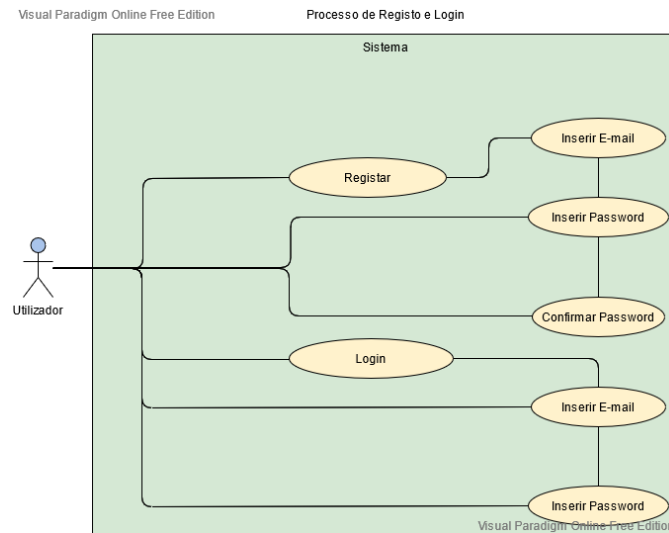
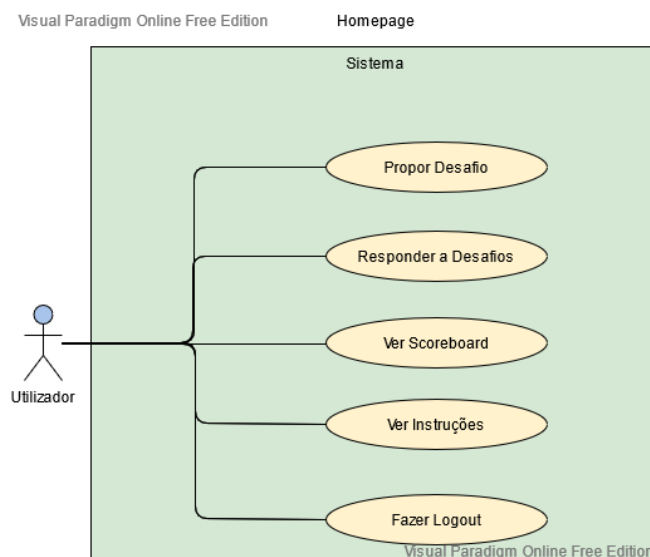
Para a plataforma *Challenge-Accepted*, foram identificados os seguintes casos de uso:

1. Processo de Registo e *Login* (Figura 2.1);
2. Acesso à *Homepage* (Figura 2.2);
3. Processo de propor desafios (Figura 2.3);
4. Processo de responder a desafios (Figura 2.4);

Os diagramas foram elaborados com recurso ao *Visual Paradigm Online*.

2.5 Modelo relacional da Base de Dados

É apresentada na Figura 2.5 o modelo relacional da Base de Dados delineada para o presente projeto.

Figura 2.1: Diagrama de casos de uso: processo de registo e *Login*Figura 2.2: Diagrama de casos de uso: *homepage*.

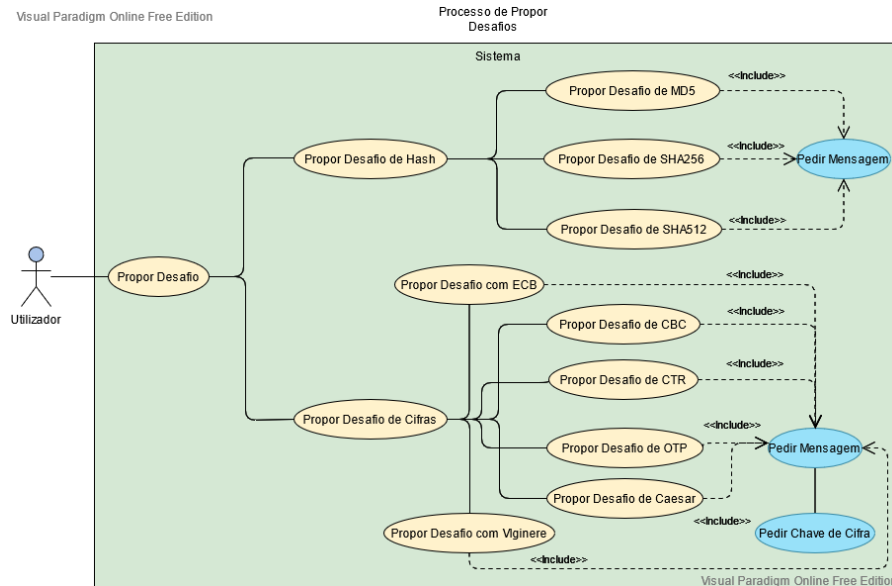


Figura 2.3: Diagrama de casos de uso: processo de propor desafios.

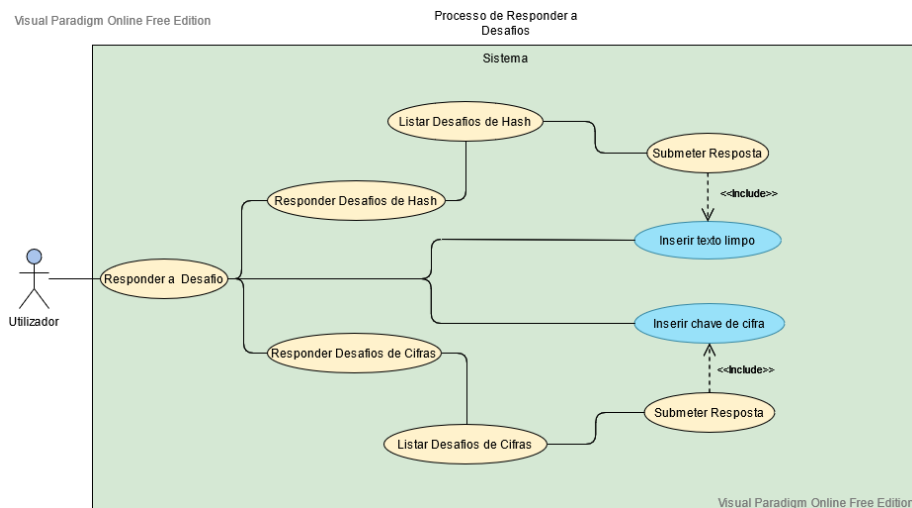


Figura 2.4: Diagrama de casos de uso: processo de responder a desafios.

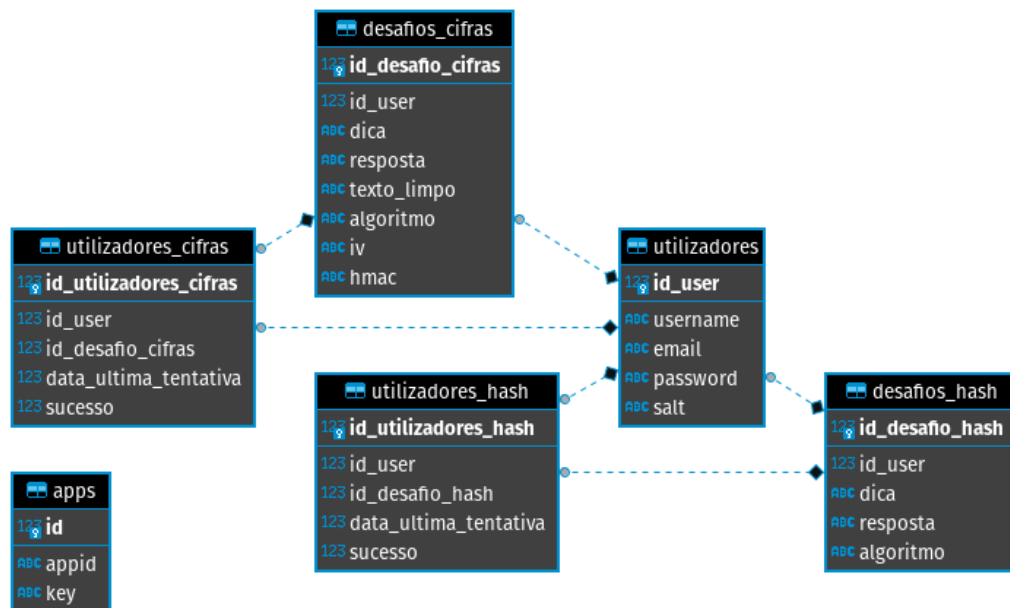


Figura 2.5: Modelo relacional da Base de Dados.

2.6 Arquitetura do sistema

Uma vez que a base de dados é remota e o serviço prevê a utilização por vários utilizadores, é relevante separar a camada de **cliente** da camada de **servidor**. Tal implica que, por exemplo, o código da aplicação do cliente não contenha instruções de acesso direto à base de dados.

Desta forma, e a fim de agilizar o desenvolvimento e teste do serviço, o sistema foi concebido com a seguinte arquitetura (Figura 2.6):

- **Base de Dados** — Alojada num servidor virtual *Windows Server 2019*;
- **Webservice** — Alojado num servidor *CentOS*, realiza a ponte entre o cliente e a base de dados, convertendo os pedidos do cliente em *queries* para a BD;
- **Cliente** — Aplicação local que comunica com o *Webservice* por *Hypertext Transfer Protocol Secure* (HTTPS).

2.7 Conclusões

Na posse de um plano delineado segundo as práticas comuns da área da Engenharia de *Software*, segue-se a fase de implementação, a qual deve seguir os requisitos

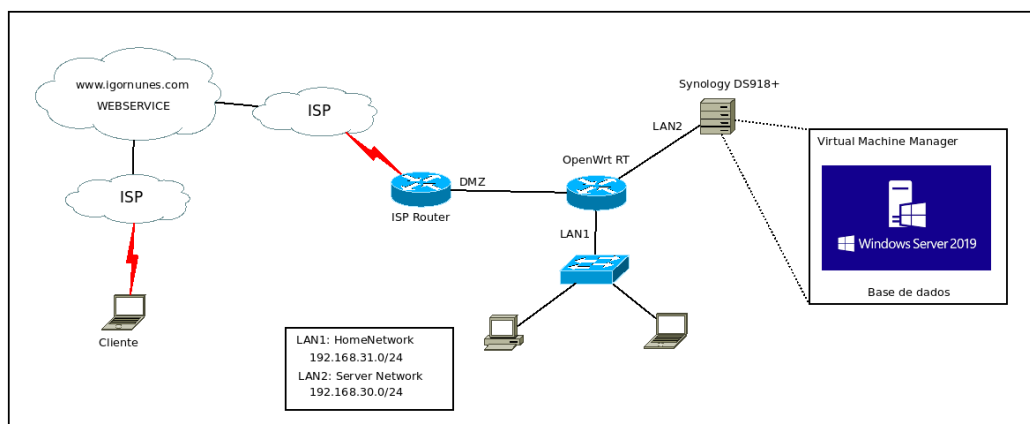


Figura 2.6: Diagrama da arquitetura do sistema.

determinados e tem por base os casos de uso estudados. Por fim, o sistema deverá seguir a arquitetura esquematizada.

Capítulo 3

Implementação

3.1 Introdução

A fase de implementação envolveu a execução paralela de diferentes tarefas pelos vários elementos da equipa. Este Capítulo aborda em particular os seguintes aspetos desta fase do projeto:

- Interface (Secção 3.2): aborda as escolhas feitas na construção da *Text-based User Interface* (TUI);
- Escolhas de implementação (Secção 3.3): explica as decisões feitas durante a implementação do código-fonte;
- Detalhes de implementação (Secção 3.4): explora os detalhes mais importantes do código-fonte.

Adicionalmente, são descritos o manual de instalação (Secção 3.5) e o manual de utilização (Secção 3.6).

3.2 Interface — TUI

Para a interface foram analisadas primeiramente duas opções existentes em bibliotecas do *Python*: *npyscreen* e *picotui*. Contudo, ambas revelaram ter uma curva de aprendizagem que não compensaria face às restantes tarefas a realizar na elaboração do projeto.

Neste sentido, a TUI segue uma filosofia minimalista e clássica de leitura de dados introduzidos por parte do utilizador, incluindo as opções dos menus para navegação.

Não obstante, a aplicação segue um código de cores para diferenciar os diferentes tipos de informação dados ao utilizador (Tabela 3.1).






Cor	Utilização	
Vermelho	Mensagem de erro (<i>error</i>)	
Amarelo	Mensagem de aviso (<i>warning</i>)	
Verde	Mensagem de sucesso (<i>success</i>)	
Ciano	Informação da aplicação (<i>info</i>)	
Cinza	Mensagens de <i>debug</i> (reservado)	

Tabela 3.1: Paleta de cores utilizada por cada tipo de mensagem dada ao utilizador.

3.3 Escolhas de Implementação

De entre as escolhas efetuadas durante a implementação da aplicação, três em particular destacam-se:

- **Python:**

A seleção da linguagem de programação teve como critérios ser de alto nível, fornecer bibliotecas atualizadas de segurança e criptografia, e disponibilizar *frameworks* acessíveis para a criação de um *webservice*. A escolha final recaiu, portanto, na linguagem *Python*.

- **MariaDB:**

Uma vez que o grupo se encontra familiarizado com bases de dados relacionais, e sendo este um modelo adequado para os dados a guardar, optou-se por uma solução *open-source*, estável, atualizada e com reputação no mundo profissional: *MariaDB*.

- **Flask:**

Após a escolha da linguagem *Python*, a *framework* que rapidamente se destacou para a criação do *webservice* foi o *Flask*. Destaca-se o facto da curva de aprendizagem desta ferramenta ser curta.

3.4 Detalhes de Implementação

3.4.1 Navegação

A navegação é feita com menus, os quais são objetos instanciados da classe `Menu`, criada para este projeto. Esta classe permite fazer a gestão automatizada de menus, sendo uma função invocada quando uma dada opção válida do menu é selecionada.

Tal permite tirar partido da *stack* de invocações do sistema operativo, levando a que a navegação entre menus seja resultado da hierarquia destas chamadas.

3.4.2 Estruturação dos *packages* e classes

A linguagem *Python*, porquanto não siga a mesma filosofia de *packages* da linguagem *Java*, tem mecanismos que o permitem simular até certo ponto.

Desta forma, os vários ficheiros que constituem bibliotecas da aplicação foram distribuídas por pastas, ficando o interpretador *Python* a conhecer a sua localização graças à criação de ficheiros `__init__.py` na raiz de cada pasta que se queira considerar como uma *package*:

- `challenge`: Classes com métodos estáticos que implementam elementos da TUI específicos ao programa;
- `dbhelper`: Comunicação com o *webservice*;
- `login`: Gestão de *login*, registo e sessão local de um utilizador;
- `tui`: Elementos essenciais à TUI;
- `utils`: Utilitários variados transversais às restantes bibliotecas.

Por outro lado, diferentes objetivos e funcionalidades são divididos em classes distintas com métodos que permitem melhor abstrair, sequencialmente, as operações que se tornam progressivamente de mais “baixo nível” (i.e., da TUI à comunicação com o *webservice*).

3.4.3 Cifras e algoritmos de *hash*

Os algoritmos implementados foram reunidos nas classes `Cypher` e `Hash` do *package* `client.util`. Os respetivos métodos foram implementados com recurso a bibliotecas disponibilizadas pela ferramenta *pip3*, as quais têm as vantagens de ser atuais, *open-source* e vastamente utilizadas pela comunidade criptográfica e de *development*.

3.4.4 *Webservice*

A fim de o cliente comunicar com o serviço alojado na *cloud* de forma encriptada e segura, foi criado um *webservice* com recurso à *framework* ***Flask***. Este escuta por pedidos no porto 443 (protocolo HTTPS) e, conforme o método do pedido (GET, POST ou PATCH) e o *Uniform Resource Locator* (URL) por onde este é efetuado, o *webservice* executa uma ação programada e devolve uma resposta com o resultado do pedido.

Os dados são bidirecionalmente encapsulados no formato *JavaScript Object Notation* (JSON). O *webservice* em particular devolve uma resposta com indicação de sucesso no pedido (fornecendo os dados respetivos) ou de erro (com a respetiva mensagem associada).

O *webservice*, a cada pedido (e após confirmar que se trata de um pedido efetuado por uma *app* autorizada (Secção 3.4.5)), abre uma conexão com a Base de Dados e processa os resultados da *query* de forma a devolvê-los num formato aceite pelo cliente (*string* ou dicionário).

O *webservice* encontra-se atualmente alojado no domínio `https://chapted.igornunes.com`.

3.4.5 Serviço de autenticação de clientes

Um cliente apenas pode realizar pedidos ao *webservice* caso seja uma *app* autorizada. Para tal, o cliente deve enfrentar um desafio proposto pelo *webservice* a fim de a autenticar.

Para tal, é gerado um *header* no pedido HTTPS com os dados necessários para o *webservice* constatar que o cliente superou o desafio e é, portanto, válido.

O algoritmo envolve um ID da aplicação cliente, uma chave associada, e a geração do *hash* do corpo do pedido HTTPS, assim como um *timestamp* e um *nonce*. No final do desafio, a *signature* deve coincidir com o *HMAC* dos dados anteriores processados.

A não presença deste *header* (Figura 3.1) ou a presença de dados que não permitem resolver o desafio do *webservice* invalidam o acesso a este, sendo o pedido recusado.

O algoritmo de *hash* utilizado no processo é o *SHA256*.

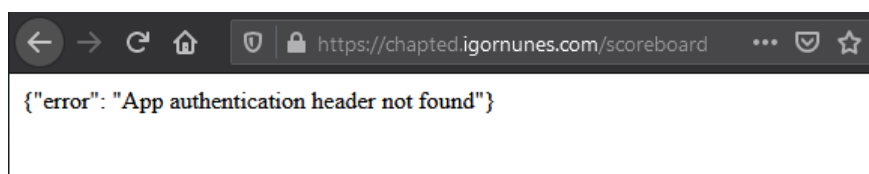


Figura 3.1: Exemplo de falha de autenticação ao aceder através de um *browser*.

3.5 Manual de Instalação

Para a primeira utilização da aplicação é necessário, no terminal, mudar para o diretório onde se encontra o ficheiro `setup.py` e executar o comando `pip3 install .` a fim de instalar as dependências necessárias do *Python*.

Após este passo, o programa pode ser utilizado conforme indicado no manual de utilização (Capítulo 3.6).

3.6 Manual de Utilização

Ao correr a aplicação *Challenge-Accepted* (com o comando `python3 app.py`), é apresentado ao utilizador um primeiro menu, no qual pode escolher entre criar uma conta (caso seja a primeira utilização da plataforma), fazer *login*, ver informações da aplicação ou sair do programa. Para o caso de criar conta, é-lhe pedido o fornecimento de um *email* válido e uma *password*, a qual é confirmada uma segunda vez.

Após o *login*, é apresentada ao utilizador uma *homepage* onde o utilizador tem acesso às várias funcionalidades da aplicação: propor desafio, responder a desafio, listar desafios e ver *scoreboard* dos jogadores.

Um utilizador poderá submeter vários tipos de desafios, mas não poderá responder aos que por ele foram criados. Conforme o utilizador responda com sucesso aos desafios propostos, é-lhe atribuída uma pontuação e o mesmo poderá verificar em que lugar se encontra no *scoreboard*.

Está disponível com o argumento `--debug` (ou `-d`) um modo de *debugging* para desenvolvimento.

3.7 Conclusões

A exposição dos pontos mais importantes relacionados com a fase de implementação do serviço *Challenge-Accepted* permitiu à equipa fazer uma retrospectiva do seu trabalho e perceber quais foram os pontos fortes e os pontos fracos do resultado final. Tal abre a porta para a fase de reflexão crítica.

Capítulo 4

Reflexão Crítica e Problemas Encontrados

4.1 Introdução

Não obstante o bom planeamento feito *a priori* na fase de engenharia de *software* (Capítulo 2), o projeto *Challenge-Accepted* enfrentou contratempos, não tendo sido possível chegar a todas as ambições inicialmente imaginadas. Reflita-se, portanto, sobre o desenvolvimento deste projeto.

Neste Capítulo são explorados os seguintes tópicos:

- Objetivos propostos vs. alcançados (Secção 4.2): compara os objetivos inicialmente propostos com aqueles que foram concluídos no projeto final;
- Divisão de trabalho pelos elementos do grupo (Secção 4.3): lista as tarefas realizadas por cada elemento da equipa;
- Problemas encontrados (Secção 4.4): na sequência da Secção 4.2, explora os problemas encontrados durante a implementação da aplicação;
- Reflexão crítica (Secção 4.5): é feita uma *Strength, Weakness, Opportunity, and Threat Analysis* (SWOT) em retrospectiva pela equipa acerca do projeto.

4.2 Objetivos Propostos vs. Alcançados

A Tabela 4.1 expõe os objetivos propostos inicialmente para o projeto e identifica quais foram alcançados na sua plenitude, quais foram alcançados parcialmente, e quais não tiveram sucesso.

Objetivo proposto	Alcançado?
Registo de utilizadores com representação segura da palavra-passe na base de dados	●
Submissão de desafios do tipo cifra de mensagens, codificando-a em <i>BASE64</i>	●
Cálculo de código de autenticação de mensagens que permite verificar se uma mensagem foi bem decifrada	●
Submissão de desafios do tipo valor de <i>hash</i> , codificando-a em <i>BASE64</i>	●
Resposta a desafios e verificação do sucesso da tentativa	●
Desafios com cifras AES-128-ECB, AES-128-CBC e AES-128-CTR	●
Desafios com funções de <i>hash</i> MD5, SHA256 e SHA512	●
Limite de uma tentativa a cada 15 segundos para os desafios	●
Verificar se uma mensagem foi bem decifrada através de assinaturas digitais RSA	—
Suporte ao algoritmo El Gamal	—
Outros tipos de desafios criptográficos	●

Tabela 4.1: Objetivos propostos e respetiva indicação de sucesso.

Legenda. ● Alcançado em pleno; ○ Alcançado parcialmente. — Não alcançado.

4.3 Divisão do Trabalho pelos Elementos do Grupo

Tarefa	DL	JA	DS	BC	IN
Engenharia de <i>Software</i>		○		●	
Instalação da infraestrutura de suporte	●				●
Desenvolvimento da base de dados	●			○	
<i>Webservice</i> com autenticação da <i>app</i>					●
Implementação dos algoritmos de cifra	●	●		●	
<i>Refactoring</i> do código final			○		●
Validação de dados de utilizador			●		
Documentação do código			●		
Testes e tentativas de ataque	●		○		
Gestão do repositório <i>git</i>					●
Relatório		●		●	●
Apresentação		●		●	

Tabela 4.2: Distribuição de tarefas pelos elementos do grupo.

Legenda. ● principal responsável; ○ auxiliou. DL: Diogo Lavareda; JA: Joana Almeida; DS: Diogo Simões; BC: Beatriz Costa; IN: Igor Nunes.

4.4 Problemas Encontrados

Nos primeiros testes à aplicação na rede *Education Roaming* (eduroam), onde o presente projeto será defendido, a equipa deparou-se com o problema desta rede bloquear as comunicações para o porto 3300, utilizado na Base de Dados.

Também devido ao facto deste servidor não ter um *Fully Qualified Domain Name* (FQDN) para aplicar certificado *Secure Sockets Layer* (SSL), optou-se por separar o *webservice* da Base de Dados de forma a minimizar as alterações à estrutura do servidor e para permitir que a aplicação possa ser executada no ambiente de rede da Universidade da Beira Interior (UBI). Como vantagem adicional, a plena separação do cliente e da base de dados com um *webservice* intermédio permite a comunicação por HTTPS (porto 443) e adiciona uma camada de segurança com autenticação da aplicação que lhe acede (Figura 2.6).

4.5 Reflexão Crítica

Para reflexão da *C-Team* face ao trabalho enveredado no desenvolvimento do serviço *Challenge-Accepted*, propõe-se efetivá-la com uma análise SWOT.

4.5.1 Pontos Fortes

1. A aplicação do cliente é *cross-platform*;
2. São fornecidas seis cifras e três algoritmos de *hash*;
3. Implementação de um sistema distribuído (arquitetura cliente – *webservice* – Base de Dados);
4. Apenas uma aplicação autorizada pode aceder ao *webservice*.

4.5.2 Pontos Fracos

1. Falta de implementação das chaves de assinatura digital *Rivest-Shamir-Adleman* (RSA);
2. Falta da presença do algoritmo de *El Gamal*;
3. A *Text-based User Interface* (TUI) é demasiado minimalista.

4.5.3 Ameaças

1. Algumas dependências de baixo nível (*e.g.* bibliotecas externas ao *Python*) mudam entre sistemas operativos;
2. As cifras *Advanced Encryption Standard* (AES) e os valores *hash* são difíceis de resolver caso o utilizador não deixe uma dica útil (em último caso torna-se tecnicamente impossível de resolver);
3. A BD encontra-se exposta a acessos externos ao do *webservice* na arquitetura atual.

4.5.4 Oportunidades

1. Reforçar a segurança e verificação dos desafios através da assinatura digital RSA;
2. Incluir novas cifras e outros algoritmos criptográficos;
3. Implementação de uma *Graphical User Interface* (GUI) e/ou de uma *web app*.

4.6 Conclusões

Esta fase de reflexão permitiu analisar o trabalho enveredado ao longo das semanas de planeamento, execução e teste. Com esta análise, a equipa pôde tirar conclusões não só sobre o seu desempenho, mas também acerca das tecnologias utilizadas, as quais serão expostas no Capítulo seguinte.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

Este projeto e a sua elaboração permitiu adquirir uma série de novos conhecimentos e técnicas na área de criptografia, segurança e respetiva aplicação em sistemas distribuídos, assim como abriu horizontes para novos métodos e técnicas de desenvolvimento.

Apesar de certos aspetos que nos pareciam significativos para o resultado final da aplicação não terem sido implementados neste projeto (nomeadamente a assinatura digital com RSA), considera-se que o trabalho necessário e essencial para o mesmo foi concluído com sucesso, acrescentando ainda algumas cifras adicionais (nomeadamente Cifra de César, Vigenere e *One Time Pad*).

A investigação necessária para a realização deste projeto servirá como uma mais-valia para o desenvolvimento de outras aplicações cuja segurança seja uma prioridade imprescindível. Para além disso, levou a um maior entendimento da matéria teórica que é abrangida no âmbito da Unidade Curricular (UC) do projeto, através da sua aplicação de forma prática em todas as etapas do processo de desenvolvimento e implementação.

5.2 Trabalho Futuro

Uma melhor integração do *webservice* com a Base de Dados é relevante a fim de evitar acessos à última fora do contexto do *webservice*. A par desta potencial falha de segurança, a implementação da assinatura com RSA é de primeira ordem.

De igual forma, a adição de novos algoritmos criptográficos tornaria o serviço mais apelativo, nomeadamente com uma interface gráfica ou uma *web app* com *layout* mais apelativo do que um programa *Text-based User Interface* (TUI).

Bibliografia

- [1] PyPI, “caesarcipher - PyPI,” 2021, [Online] <https://pypi.org/project/caesarcipher/>. Último acesso a 9 de maio de 2021.
- [2] —, “elgamal - PyPI,” 2021, [Online] <https://pypi.org/project/elgamal/>. Último acesso a 10 de maio de 2021.
- [3] —, “vigenere - PyPI,” 2021, [Online] <https://pypi.org/project/vigenere/>. Último acesso a 10 de maio de 2021.
- [4] —, “onetimepad - PyPI,” 2021, [Online] <https://pypi.org/project/onetimepad/>. Último acesso a 10 de maio de 2021.
- [5] “AES Modes (Python),” 2021, [Online] https://asecuritysite.com/encryption/aes_modes. Último acesso a 8 de maio de 2021.
- [6] S. Overflow, “python - PyCrypto problem using AES+CTR,” 2021, [Online] <https://stackoverflow.com/questions/3154998/pycrypto-problem-using-aesctr>. Último acesso a 8 de maio de 2021.
- [7] PyCryptodome, “Crypto.Hash package PyCryptodome 3.9.9 documentation,” 2021, [Online] <https://pycryptodome.readthedocs.io/en/latest/src/hash/hash.html>. Último acesso a 17 de maio de 2021.
- [8] S. Overflow, “python - Checking the strength of a password (how to check conditions),” 2021, [Online] <https://stackoverflow.com/questions/16709638/checking-the-strength-of-a-password-how-to-check-conditions>. Último acesso a 20 de maio de 2021.
- [9] GitHub, “karolyi/py3-validate-email: Check if an email is valid with using SMTP, regexes and blacklists,” 2021, [Online] <https://github.com/karolyi/py3-validate-email>. Último acesso a 20 de maio de 2021.