

# IEDCS: Identity Enabled Distribution Control System

Segurança 2015/2016

Relatório realizado por:

- Bruno Silva (68535)
- Bernardo Ferreira (67413)

## Descrição

O IEDCS é um sistema de DRM (Digital Rights Management) para ebooks. Neste projeto de segurança o nosso objectivo é criar um sistema completo, desde o servidor ao cliente, para fazer a gestão e distribuição dos livros.

O nosso projeto consiste em duas partes, o servidor e o player, que vão ser descritos com mais pormenor mais a baixo.

Dadas as metas pretendidas para esta fase do projeto, podemos afirmar que foram conseguidas com sucesso uma vez que temos um sistema funcional, onde um utilizador se pode registar no sistema, comprar livros, ver os livros que comprou e ler os livros. Todo este processo se encontra dentro dos requisitos pedidos onde, sempre que um utilizador pretende ler um livro necessita sempre do servidor, nunca são enviadas as chaves necessárias a decifra do ficheiro e toda a comunicação é realizada usando https e um certificado x509.

Como aspetos a melhorar, consideramos que deveria haver uma melhor gestão dos erros e no caso de a aplicação correr em UNIX, não conseguimos a tempo criar uma device key baseada no MAC adress do computador, o que nos obrigou a utilizar uma device key 'hardcoded'. De salientar que este problema não afeta o programa se este correr em MacOS ou Windows. Na próxima estes problemas serão resolvidos.

# Componente Server

## Descrição:

O IEDCS Server é o componente responsável por garantir a boa utilização e distribuição do conteúdo que fornece. Isto significa que ele tem de garantir a gestão de utilizadores, das compras desses utilizadores. Além dessas funções ele tem de assegurar a distribuição dos conteúdos adquiridos pelos utilizadores de forma segura e controlada.

O servidor, no nosso sistema, está implementado como um servidor web usando a linguagem de programação Python e a framework de Django. A comunicação com o player e o web site é feita através de uma API com endpoints HTTP REST.

## Descrição da API:

A API no servidor está implementada com a “Django REST framework”, implementa endpoints para permitir a comunicação entre o servidor e o cliente seja este o player ou uma pagina web. Os seguintes endpoints foram implementados a medida das necessidades do sistema e fornecem a funcionalidade pedida assim como validações simples a todos os parâmetros recebidos:

`/books/get_books/`

Endpoint que suporta o método GET para receber a lista de todos os livros existentes no sistema. Cliente recebe a lista completa em json contendo informação como o: título, autor, url para a capa, língua em qual está o livro escrito, e um identificador único do livro.

`/books/get_book/?book_id=<0-9>`

Endpoint que suporta o método GET para receber informação específica sobre o livro indicado como argumento “book\_id”. A informação recebida é a mesma do endpoint anterior mas apenas para o livro pedido.

`/users/create_user/`

Endpoint que suporta o método POST para criar um novo utilizador no sistema. Para registar um utilizador novo no sistema é necessário fornecer os parâmetros: primeiro nome, ultimo nome, e-mail e password. No momento de criação de um novo utilizador é gerado um valor aleatório de 16 bytes que é a user key. Este valor é guardado na base de dados numa tabela diferente da de utilizadores, e é usado para cifrar as compras daquele utilizador.

Após o registo e ainda na mesma função o utilizador é autenticado e é realizado o login do mesmo e é definido um cookie com o primeiro nome e e-mail do utilizador para ser usado pela página web.

#### `/users/user_login/`

Endpoint que suporta o método POST para fazer login a utilizadores do sistema. O login requer os parâmetros: e-mail e password. De seguida utiliza os métodos do Django para realizar o login do utilizador no sistema.

#### `/users/user_logout/`

Endpoint que suporta o método POST para fazer logout a utilizadores do sistema. O logout não requer quaisquer parâmetros, usando apenas o request. Para efetuar o logout são usados os métodos do Django.

#### `/users/buy_book/`

Endpoint que suporta o método POST para um utilizador poder comprar um livro. Para a compra de um livro é necessário apenas enviar como argumento o identificador do livro os outros parâmetros necessários são conseguidos a partir do request.

Para efetuar uma compra o Django cria um random e de seguida cria uma entrada na tabela de compras com o utilizador atual, o identificador do livro e o random associado que vai ser usado para cifrar leituras daquela compra.

#### `/users/get_purchases/`

Endpoint que suporta o método POST para receber todas as compras de um utilizador. Recebe como argumento o e-mail do utilizador para o qual queremos as compras e retorna uma lista em json de todas as compras desse utilizador.

#### `/users/register_device/`

Endpoint que suporta o método POST para registar um device com um user. Recebe como argumento o e-mail do utilizador, a device key e um nome do device. Caso o nome do device seja vazio este é preenchido com "<first\_name\_user>'s Computer". Numa situação em que esse dispositivo já esteja associado aquele user é dada uma resposta a informar dessa situação.

#### `/requests/read_book/`

Endpoint que suporta o método POST para inicializar o processo de leitura de um livro. Como argumento recebe o id do livro que o utilizador quer ler e valida se ele já comprou o livro. Como resposta é enviado um header com os parâmetros que o cliente tem

de enviar para garantir que a leitura do livro é possível. Esses parâmetros são: book\_id, device\_key, player\_version, location, so e time.

#### /requests/validate/

Endpoint que suporta o método POST para validar os dados do utilizador e cifrar o livro para ele ler. Como argumentos este método recebe todos os parâmetros que pediu ao utilizador na chamada ao endpoint anterior (/requests/read\_book/). Após as validações dos parâmetros o método acede a base de dados às tabelas das restrições e verifica se os dados do utilizador estão dentro dos permitidos para a leitura do livro. Depois de garantir as restrições é efetuado o início da cifra do livro. Para isso é gerada uma file key que vai depender do random da compra, da player\_key, da user\_key e da device\_key da seguinte forma:  $file\_key = E(device\_key, E(user\_key, E(player\_key, random)))$ , considerando a função de cifra  $E(chave, texto)$ . Todas estas cifras são realizadas com AES no modo CBC.

Após estar gerada a file key, essa chave é usada para cifrar, do mesmo modo anterior, o livro por blocos e gerado um novo ficheiro para o livro onde este está cifrado.

#### /requests/get\_file/

Endpoint que suporta o método POST para receber o ficheiro cifrado previamente. Este método recebe o book\_id do livro pretendido e envia o conteúdo do livro com a codificação base64. Após o utilizador chamar este método o livro cifrado é apagado e para uma nova leitura todo o processo tem de ser repetido.

#### /requests/decrypt/

Endpoint que suporta o método POST para permitir auxílio ao utilizador de fazer a decifra do ficheiro. Este método é necessário para o utilizador necessitar sempre da interação do servidor para conseguir ler o ficheiro, uma vez que a user key nunca sai do servidor e ela é usada no processo.

## Descrição da WebStore:

Neste projecto a webstore não era um dos objetivos principais do trabalho, dado está condição foi implementado um modelo simples que permite apenas registar utilizadores, fazer login, ver os livros disponíveis e efetuar compras.

# Componente Player

## Descrição:

O IEDCS Palyer é o componente responsável pela reprodução de um livro previamente comprado e distribuído pelo servidor.

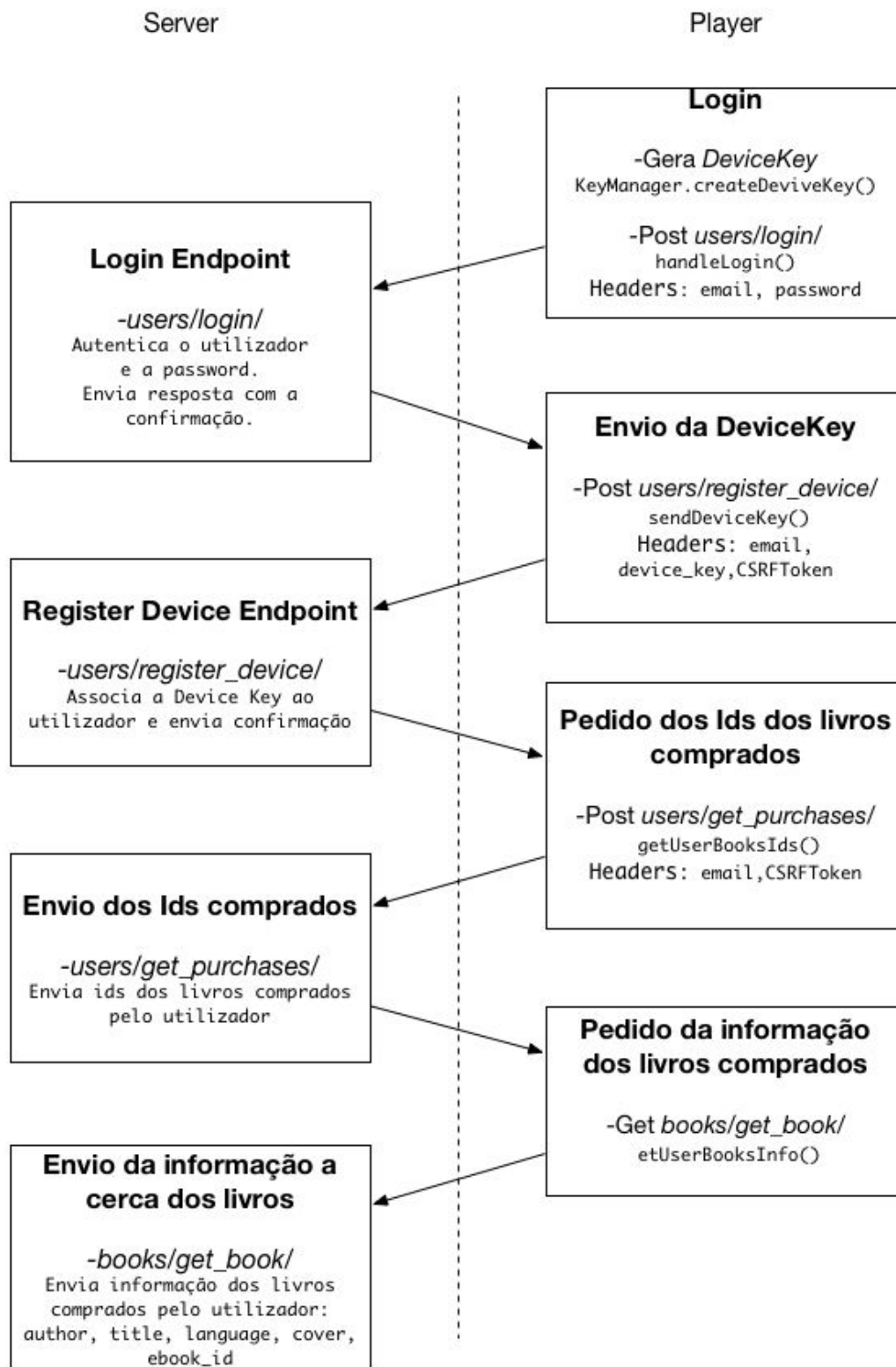
O Player está implementado como uma aplicação Java, utilizando a biblioteca gráfica JavaFX. A comunicação entre a aplicação e o servidor é feita utilizando a biblioteca HttpClient da Apache.

## Fluxo do programa:

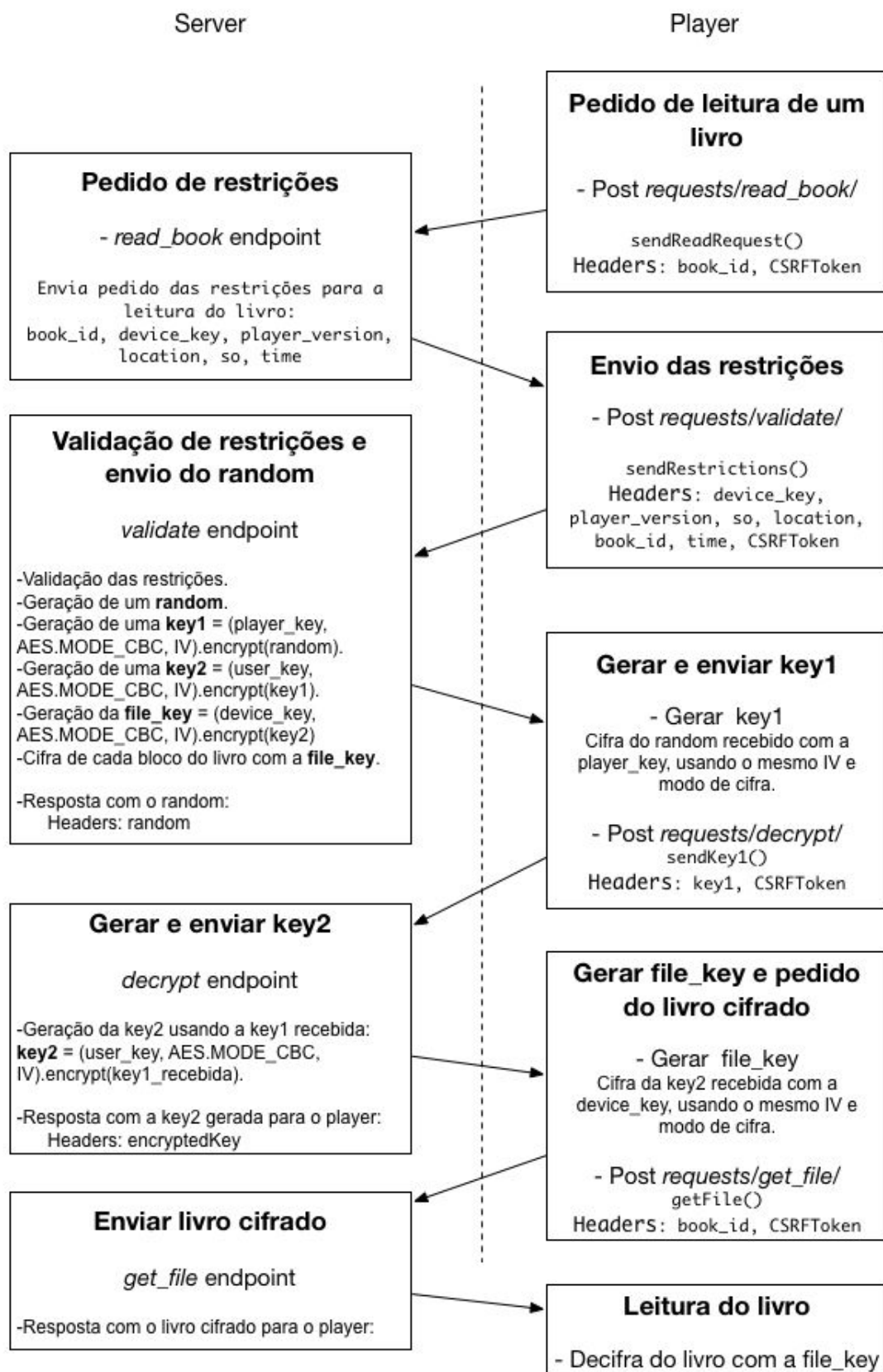
Depois de iniciar a aplicação é criada uma *Device Key* como sendo o número de série do computador utilizado, de modo a identificar unicamente o dispositivo que está a ser utilizado, e, é de seguida apresentada uma janela de login, onde devem ser introduzidos os dados de autenticação previamente usados na criação de uma conta na Web Store. Estes dados são enviados para o servidor e no caso de os dados estarem corretos, a *Device Key* é também enviada de modo a identificar o dispositivo em que o utilizador está a usar a aplicação.

De seguida, as informações sobre os livros comprados pelo utilizador são pedidas ao servidor, e é apresentada uma janela com a listagem destes livros. O utilizador pode escolher um livro para ver mais informações e seleccioná-lo para ler. O processo de pedido do livro ao servidor bem como toda a interação Server <-> Player vai ser descrita no seguinte diagrama:

- Login:



- Leitura de um livro:





## Instalação e execução:

### Server

Para a instalação do server basta correr dentro da pasta “m1/server/” o script “install.sh”. Embora o script não corra em sudo ele vai pedir a palavra passe para instalar o python-pip caso não esteja instalado. De seguida ele instala o python (1.8.6), e outras bibliotecas necessárias à execução do programa. A seguir o script apaga a base de dados, caso já exista uma, faz migrate para gerar as tabelas da base de dados novamente. Um script de população de base de dados é executado de seguida que adiciona alguns livros a base de dados, as restrições a serem implementadas e os players validos.

Para executar o servidor corre-se o comando “python manage.py runserver\_plus --cert certs/”, isto executa o servidor de desenvolvimento do django com https usando os certificados que estão na pasta “certs”.

Após o servidor ser executado, terá de se aceder a <https://127.0.0.1:8000/>, e carregar em “Sign up FREE” para criar uma conta. De seguida basta comprar os livros desejados.

### Player

Para executar o programa, deve-se ir a pasta “m1/player/IEDCS/deploy/” e executar o jar que lá se encontra com o comando “java -jar IEDCS.jar”.

Após o player estar em execução basta fazer login com as mesmas credenciais que foram criada na webstore, após ter entrado com sucesso vai ser levado para uma página onde aparecem os livros que já comprou. basta selecionar o pretendido e carregar em “Read Book”. Vai ser levado para um leitor com o livro.

## Links auxiliares:

### Server

Cifrar em python com AES:

<http://eli.thegreenplace.net/2010/06/25/aes-encryption-of-files-in-python-with-pycrypto>

<http://stackoverflow.com/questions/12524994/encrypt-decrypt-using-pycrypto-aes-256>

Criar utilizadores e gerir login/logout:

<https://docs.djangoproject.com/en/1.8/topics/auth/default/>

Testar conexões em https com o servidor de desenvolvimento e com um certificado auto assinado:

<http://stackoverflow.com/questions/8023126/how-can-i-test-https-connections-with-django-as-easily-as-i-can-non-https-connec>

Definições para o https no Django:

[https://docs.djangoproject.com/en/1.8/ref/settings/#std:setting-SECURE\\_HSTS\\_INCLUDE\\_SUBDOMAINS](https://docs.djangoproject.com/en/1.8/ref/settings/#std:setting-SECURE_HSTS_INCLUDE_SUBDOMAINS)

### Player:

Ignorar verificação dos certificados SSL para chamar endpoints em https:

<http://literatejava.com/networks/ignore-ssl-certificate-errors-apache-httpclient-4-4/>

Tutorial de iniciação ao JavaFX:

<http://code.makery.ch/library/javafx-8-tutorial/>

Encontrar o IP público:

<http://stackoverflow.com/questions/2939218/getting-the-external-ip-address-in-java>

Tutorial Apache HttpClient:

<http://hc.apache.org/httpclient-3.x/userguide.html>

Encontrar localização através do IP:

<http://www.mkymong.com/java/java-find-location-using-ip-address/>

Encontrar o MAC Address em java:

<http://www.mkymong.com/java/how-to-get-mac-address-in-java/>