Identity Enabled Distribution Control System

Universidade de Aveiro

Diogo Silva 60337
Tânia Alves 60340

Identity Enabled Distribution Control System

1st Project

Segurança

Universidade de Aveiro

Diogo Silva 60337 Tânia Alves 60340

13 de Novembro de 2015

Conteúdo

Context

This project was done for Security, for the 2015/2016 lective year. It aims to create an end to end secure digital rights management system to handle the distribution of video files, music files or books.

Introduction

Our project is made of two main components: the player and the server. The server is in charge of controlling the user access to the protected files. The players requests and plays the files from the server. In order for the user to have access to the titles he/she wants, we also have a web application where the user can buy the titles to play later. To reach the goal of this project, we also needed a database that helped manage the user and file related information.

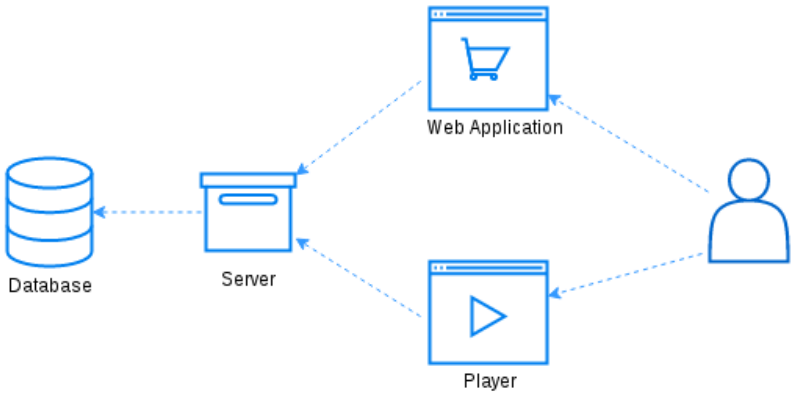

Figura 1: Component overview

Capítulo 1

Database

This chapter includes information about the database we used to support the server. We stored information about the users, files, players and devices that were later used.

1.1 Database

For the database we thought of this layout:

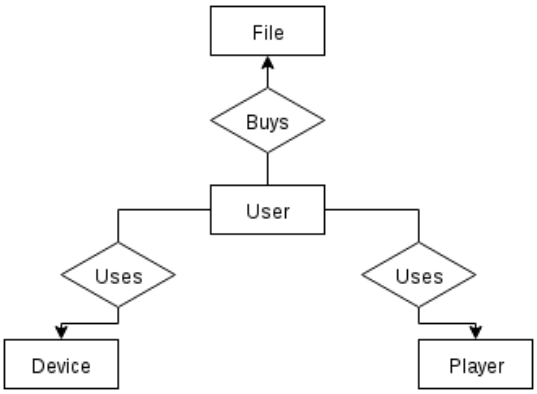

Figura 1.1: DB Schema

We need to save information about the Users that belong to the system and buy the files. The Files that are stored in the server and are then sent to the players. The Players that will interact with the server and play the files. And the Devices where the users play the files.

1.1.1 Database tables

Using the information presented, we derived the tables for the database, with all the attributes necessary for our implementation of the system.

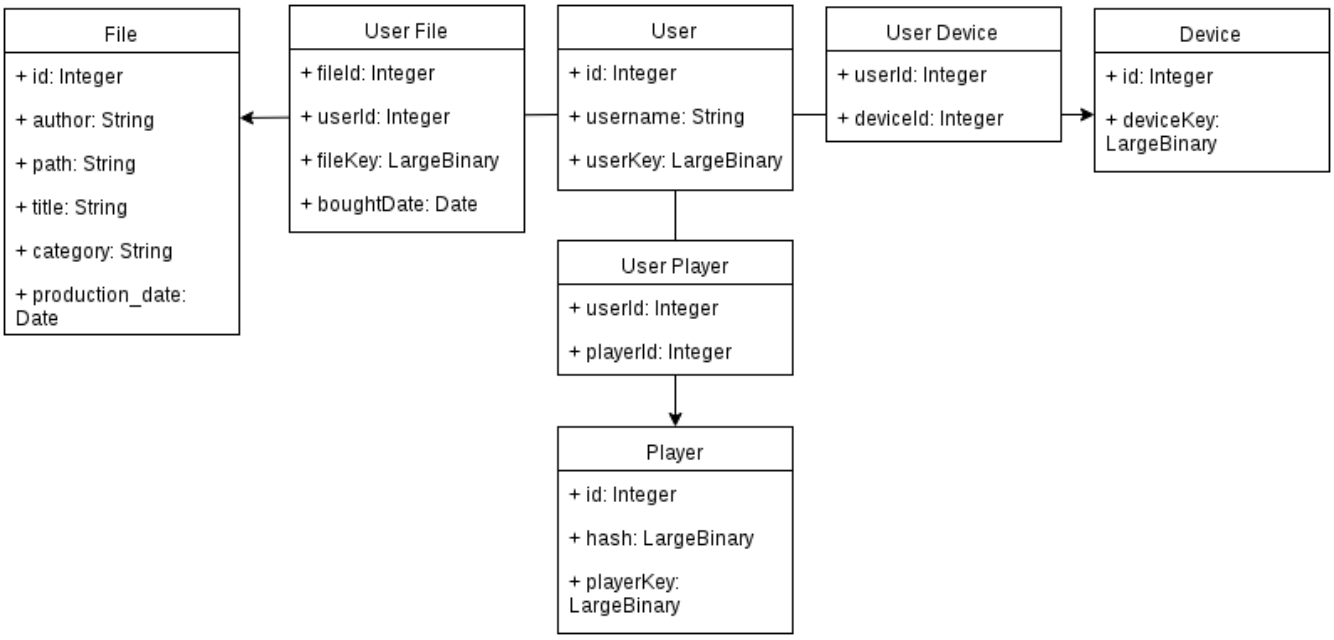

Figura 1.2: DB Tables

Each "main"entity (User, Device, File and Player) will have an id that identifies the entity in the database. We then have the required key for each one of them as well.

1.1.2 Technologies used

For the database we used:

PostgreSQL Open source, object relational database management system. We chose this over SQLite for example because it has a better support for storing secure data.

SQLAlchemy This is an open source SQL toolkit. Works as an object-relation mapper for Python. This allowed us to create database scripts that created the tables and populated the database.

1.2 Serviços Web

Tal como sugerido na descrição do projecto foram implementados 6 serviços diferentes, 2 deles relacionados com o chat e 4 com a imagem.

/chat/send: usado para enviar uma mensagem.
/chat/recv: usado para obter as últimas mensagens.
/image/send: usado para enviar uma imagem.

/image/get/all: usado para obter a lista de imagens
/image/get/thumb: usado para obter a versão reduzida de uma imagem.
/image/get/img: usado para obter uma imagem

Convém referir que existe uma base de dados em SQLite no servidor que guarda todos os dados relativos as mensagens do chat e as imagens.
Tendo então duas tabelas, uma para o chat e outra para as imagens (mais precisamente para o path da imagem).

A tabela das images (images) contém 5 campos:

- id (INTEGER PRIMARY KEY AUTOINCREMENT), um identificador único para cada imagem
- filename (TEXT), nome do ficheiro
- latitude (REAL), que corresponde a latitude onde a imagem está a ser colocada
- longitude (REAL), que corresponde a longitude onde a imagem está a ser colocada
- author (TEXT), identifica a quem pertence a imagem, sendo colocado o nome em marca de agua no canto superior esquerdo da imagem

A tabela das mensagens do chat (chat) contém 6 campos:

- id (INTEGER PRIMARY KEY AUTOINCREMENT), um identificador único para cada mensagem
- author (TEXT), que corresponde a quem pertence a mensagem
- message (TEXT), mensagem enviada
- latitude (REAL), local donde foi enviada a mensagem
- longitude (REAL), local donde foi enviada a mensagem
- time (REAL), tempo a que a foi enviada a mensagem

1.2.1 Chat

Envio de mensagem

O envio da mensagem é realizado através de um POST para /chat/send com mensagem JSON como no seguinte exemplo:

```
{
    "author": "Diogo Silva",
    "message" : "Mensagem de teste!",
    "latitude" : 48.654848,
    "longitude" : -8.6454
}
```

Sendo que o serviço após receber esta mensagem, a única coisa que vai fazer é colocar na base de dados com o tempo actual (usando a biblioteca time de python) de forma a ficar armazenada.

Recepção de mensagem

A recepção de mensagens já é um bocado mais complexa do que o envio, sendo feito através de um GET para /chat/recv com os parametros lat (latitude), lon (longitude) e time (corresponde ao tempo da ultima mensagem recebida). Como por exemplo:
/chat/recv?lat=48.654848&lon=-8.6454&time=12512512312

Se o campo time for igual a 0 (quer dizer que não existem mensagens carregadas ainda), o serviço vai buscar a base de dados 20 mensagens com menos de 30 minutos e dessas mensagens só manda aquelas cujo têm uma distância inferior a 1 km.

Caso o campo seja diferente de 0, quer dizer que já existem mensagens carregadas, então o serviço vai buscar todas as mensagens que foram enviadas após esse tempo e dessas apenas responde ao cliente com as que têm distância inferior a 1 km.

A resposta é enviada em formato JSON, com os campos time, distance, author e message.

1.2.2 Imagem

Envio de imagem

O envio da imagem para o servidor é feito através de um POST para /image/send com os parametros imagem (image), latitude (lat), longitude (lon), autor (author) e um campo opcional filtro que corresponde ao nome do filtro pretendido para a imagem.

Quando o serviço é invocado ele percorre vários passos até introduzir a imagem na base de dados:

- Gera um nome para o ficheiro da imagem utilizando MD5 de forma a evitar nomes repetidos na mesma pasta
- Escreve o nome do autor no canto superior esquerdo da imagem em marca de água
- Se houver filtro selecciona, aplica-se a imagem
- Cria uma imagem em versão reduzida (thumbnail)
- Cifra com AES[**?**] a versão completa e a versão thumbnail.

- Introduz os dados relativos a imagem na base de dados (nome do ficheiro, latitude, longitude, autor)

Todas as imagens vão ser guardadas numa pastas isolada chamada uploads/

Recepção de uma lista de imagens

A recepção de uma lista de imagens disponíveis no servidor é feita através de um GET para /image/get/all em que tem como parametros latitude, longitude e distancia.

Sendo que o servidor responde com uma mensagem em JSON correspondente a todas as imagens que estão a menos do que a distância passada por argumento em relação ao ponto (latitude, longitude) também passado por argumento. Esta informação é obtida através da latitude e longitude de cada imagem existência na tabela images da base de dados.
Um exemplo de resposta seria:

```
[
    {
        "id": 2,
        "latitude": 46.54545,
        "longitude": -8.08844
    },
    {
        "id": 5,
        "latitude": 46.7345,
        "longitude": -8.1231

    }
]
```

Recepção da imagem completa

A recepção da imagem completa é feita através de um GET para /image/get/img com o parametro id que pode ser obtido fazendo o GET para /image/get/all.
Caso o id não exista na base de dados, é imprimido uma mensagem de erro, caso contrário é enviada uma resposta para o utilizador no formato da imagem pretendida (extensão relativa).

Antes da imagem ser enviada, o servidor vai decifrar a imagem, porque sempre que uma imagem é colocada no servidor, vai ser colocada cifrada.

Recepção da imagem reduzida (thumbnail)

A recepção da imagem completa é identica a recepção da imagem completa só que é feita através de um GET para /image/get/thumb. E em vez de ir procurar pelo nome do ficheiro que a base de dados informa relativo ao id, vai procurar por esse nome concatenado com _thumb, que é como a imagem está armazenada no sistema.

1.3 Extras

Esta secção diz respeito a todas as funcionalidades adicionais da aplicação.

1.3.1 Cifra para a Imagem

A cifra que se optou por escolher para cifrar/decifrar a imagem foi AES[**?**] devido a ser uma cifra por blocos, que é mais comum ser utilizada para dados armazenados como se trata do caso da imagem.
Para além da escolha da cifra utilizada para a imagem, também foram adoptadas outras opções.

A cifra por blocos que se vai utilizar tem blocos de 16 em 16 bytes, sendo que a chave tem de ter tamanho 16 bytes.

Para evitar obrigar a introdução de uma chave com 16 bytes optou-se por deixar introduzir qualquer tamanho em que se for menor que 16 bytes é aplicado uma função de síntese SHA sobre a chave aproveitando-se os 16 primeiros bytes do resultado, caso a chave já tenha mais que 16 bytes, apenas se aproveita os primeiros 16, sem ter de aplicar qualquer função de síntese.

Após cifrar se o bloco não estivesse alinhado, aplicou-se a definição de padding PKCS #7, sendo que ao decifrar a mensagem era preciso retirar o padding caso existisse.

1.3.2 Testes

Foram efectuados testes para os serviços web, mas como também para a cifra AES.

Relativamente a cifra AES, foram efectuadas 3 testes distintos:

- Mensagem com tamanho aleatório diferente de um multiplo de 16 bytes
- Mensagem com tamanho igual a 16 bytes
- Mensagem com tamanho igual ao tamanho do bloco completo a ser lido (512 bytes)

Estes testes permitiram que o padding PKCS #7 [**?**] fosse testado na totalidade e ainda permitiu testar a cifra.
Sendo que se cifrava a mensagem e depois decifrava, e se a mensagem fosse igual a inicial, significava que tinha passado no teste.

Para os testes dos serviços web foram efectuados 3 testes para o chat e 1 teste para as imagens.
Para testar o /chat/send foi criado um formato json com a mensagem e enviado para a função que envia a mensagem para a base de dados, após a mensagem ser enviada é verificado se a mensagem se encontra ou não na base de dados.

Para testar o /chat/recv foram efectuados dois testes distintos.
O primeiro teste consiste em inserir na base de dados uma mensagem com o tempo actual e depois chama-se o serviço para verificar se consegue ou não retornar a mensagem.

O segundo teste consiste em inserir novamente uma mensagem na base de dados com o tempo actual e depois chamar o serviço mas informar que a ultima mensagem recebida foi a mais de 30 minutos, sendo assim o servidor não deve enviar qualquer tipo de mensagem porque já ultrapassou o tempo limite (30 minutos).

Para testar o serviço de imagens apenas se realizou um teste para o /image/get/all tendo em conta que os outros serviços relativos as imagens são serviços que se verifica visualizando a imagem em si e não consultando um campo na base de dados.

Esse teste consiste em simular uma entrada de uma imagem na base de dados e pedir todas as imagens a distância de 0 km para o mesmo ponto, o que é suposto não devolver nada devido a ser uma distância nula. Depois também se testou inserir uma outra entrada na base de dados e verificar se o número de imagens retornadas pelo serviço tinha incrementado uma unidade para a distância correcta.

1.3.3 Filtros das Imagens

As imagens caputaras podem ser alvo de aplicação de um filtro, entre os quais:

- Elevada saturação
- Baixa saturação
- Elevada intensidade
- Baixa intensidade

- Tons de Cinza
- Sépia
- Vignette

Saturação

Para aplicar o filtro de saturação a uma imagem, tem de se alterar a intensidade das cores da mesma. No entanto o modo de representação da cor tem de ser alterado de RGB para YCbCr, que representa a luminância, crominância azul e crominância vermelha, e depois alterar os canais Cb e Cr, multiplicando-os por um factor. Estes canais são alterados uma vez que representam a saturação pela diferença em relação ao seu valor central (128). Existem dois filtros que alteram os niveis de saturação, distinguindo-se pela distância do seu valor ao valor central. Programáticamente, para isto acontecer, o factor de multiplicação é superior a um quando queremos que a saturação seja elevada e inferior a um caso contrário.

Intensidade

No caso de se querer aplicar o filtro de intensidade, tem de se alterar, tal como anteriormente, a intensidade das cores, convertendo igualmente o modo de representação da imagem para YCbCr, no entanto, neste caso, altera-se o valor Y, que possui informação sobre a intensidade, multiplicando-se este canal por um factor.
A aplicação deste efeito pode resultar em duas imagens distintas, sendo que se o factor for superior a um, a intensidade é maior e se for inferior a um acontece o caso contrário e a imagem fica mais escura.

Tons de Cinza

Para aplicar este filtro a uma imagem tornando-a em tons de cinza, basta remover toda a informação sobre as cores, deixando apenas os valores da intensidade. Para isso, é apenas necessária a conversão da imagem para a representação L, que apenas tem um canal que representa a Luminância.

Sépia

Este filtro faz com uma imagem fique com aspecto mais antigo, bastando para isso manipular os valores da representação RGB, pixel a pixel, multiplicando cada um deles por:

```
nr = r*0.189+g*0.769+b*0.393
ng = r*0.168+g*0.686+b*0.349
nb = r*0.131+g*0.534+b*0.272
```

Onde nr, ng e nb representam os novos valores para os canais RGB, e r, g e b os valores anteriores.

Vignette

O vignette é um efeito que torna as bordas das imagens mais escuras, mostrando assim um defeito das lentes fotográficas. Para o aplicar, é necessário calcular um factor de atenuação, o centro da imagem e depois multiplica-se os valores RGB por um factor diferente, conforme o afastamento ao centro da imagem.

Capítulo 2

Tarefas

Tendo em conta que o projecto foi desenvolvido por apenas dois alunos foi mais fácil de gerir a divisão de tarefas.
Fazer a divisão das tarefas foi bastante simples, um ficou com a parte de backend (webservice para o chat/imagem, cifra, testes) a excepção dos filtros das imagens, e o outro ficou com a parte de frontend (layout das páginas, scripts e styles).

2.1 Divisão de tarefas

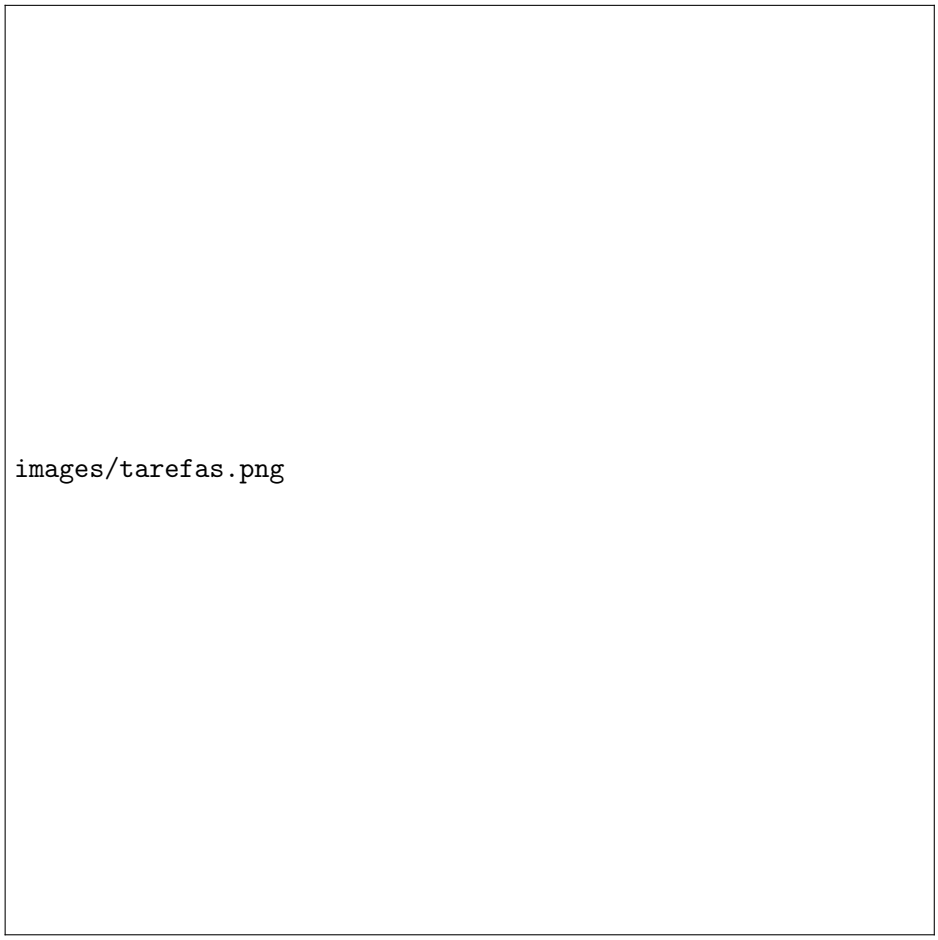

Figura 2.1: Tarefas no CodeUA

Tal como mostra a figura, dá para perceber que o responsável pelo backend foi o Diogo Silva enquanto que o responsável pelo frontend foi o Tomé Marques, além do frontend o Tomé também ficou responsável por fazer os filtros das imagens.

Mais detalhadamente, o Diogo Silva realizou o Web Service na totalidade, a cifra AES[**?**] para as imagens, os testes e os scripts relativos ao envio/recepção da imagem e envio/recepção de mensagens, ou seja, a comunicação com o servidor.
Por outro lado, o Tomé Marques realizou o o design da aplicação móvel (o seu layout) na totalidade, como também os scripts relativos a carregar o mapa e actualizar a sua posição e ainda realizou os filtros para as imagens.

Relativamente a este relatório, ambas as partes preencheram o relatório naquilo que lhes competia.

Conclusão

A aplicação web desenvolvida apesar de inicialmente se pretender desenvolver para dispositivos móveis, também se acabou por adaptar funcionando assim em qualquer dispositivo.

A aplicação apresenta tal como previsto uma secção de chat que permite comunicar com utilizados da mesma aplicação que estejam a uma distância inferior a 1 km, muito parecido com o actual chat open-source Firechat[**?**], como também permite a partilha de imagens para quem pretenda partilhar o local onde está, por exemplo.

Com isto, o grupo adquiriu os conhecimentos básicos relativos a vários campos distintos, tais como, Python, cifras, serviços web, imagem, programação web, entre outros.

Lista de Figuras