**CLOUD-IN-ONE: INTEGRACIÓN LOCAL DE SERVICIOS DE ALMACENAMIENTO EN LA NUBE**

AUTOR

GUZMÁN PEDRAJAS, VÍCTOR

TUTOR

Saiz Noeda, Maximiliano

DEPARTAMENTO

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

CURSO:

2014 – 2015

Índice

[Introducción 1](#_Toc423259596)

[¿Qué es? 1](#_Toc423259597)

[¿Por qué hacerlo? 1](#_Toc423259598)

[Diferencias con otros servicios similares 1](#_Toc423259599)

[Justificación 1](#_Toc423259600)

[Estructura del código 2](#_Toc423259601)

[UML 2](#_Toc423259602)

[Patrones de diseño usados 2](#_Toc423259603)

[Algoritmo de aplicación de cambios 3](#_Toc423259604)

[Explicación básica 3](#_Toc423259605)

[Pseudocódigo? 3](#_Toc423259606)

[Encriptación 3](#_Toc423259607)

[Explicación básica 3](#_Toc423259608)

[Algoritmo de encriptación 3](#_Toc423259609)

[Librería(s) de encriptación 3](#_Toc423259610)

[¿Para qué sirve esta encriptación? ¿Ante qué protege? 3](#_Toc423259611)

[Debilidades - ¿Ante qué NO protege? 3](#_Toc423259612)

[Casos de uso 3](#_Toc423259613)

[Uso normal – 1 pc 3](#_Toc423259614)

[Uso normal – 2 pc’s 3](#_Toc423259615)

[Uso esporádico – cio.exe 3](#_Toc423259616)

[Uso esporádico – cio-crypt.exe 3](#_Toc423259617)

[Resultados y conclusiones 4](#_Toc423259618)

[Bibliografía y enlaces de interés 5](#_Toc423259619)

# Introducción

## ¿Qué es?

## ¿Por qué hacerlo?

## Diferencias con otros servicios similares

# Justificación

Blablablá

# Estructura del código

CLOUD-IN-ONE se ha programado en Python usando un paradigma orientado a objetos.

#TODO: explicación…

## UML

El siguiente diagrama representa las clases de la aplicación:

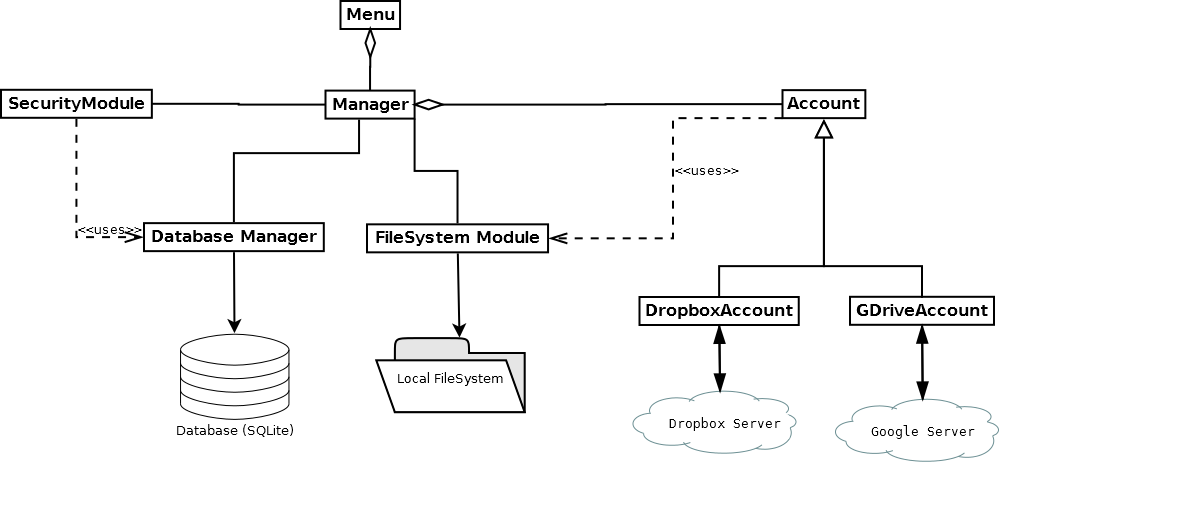


Ilustración 1 - Diagrama UML general

#TODO: explicación…

En este diagrama se ha omitido la clase *Logger*, que se encarga de gestionar la escritura de un fichero de Log tanto por consola como a un fichero. Esta clase está vinculada a todas las demás clases, que crean una instancia en su constructor.

## Patrones de diseño usados

La sección que controla el acceso a las cuentas implementa un patrón **Strategy**, lo que permite añadir fácilmente nuevos servidores remotos, haciendo que implementen la interfaz *Account*.

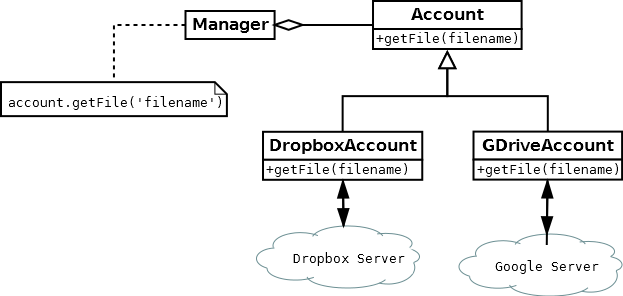


Ilustración 2 - Ejemplo del patrón Strategy con el método getFile()

A su vez, las clases que llaman a los servidores remotos implementan un patrón **Façade**, que simplifica el uso de las APIs remotas por parte del resto de la aplicación.

# Algoritmo de aplicación de cambios

## Explicación básica

## Pseudocódigo?

# Encriptación

## Explicación básica

### Algoritmo de encriptación

### Librería(s) de encriptación

## ¿Para qué sirve esta encriptación? ¿Ante qué protege?

## Debilidades - ¿Ante qué NO protege?

# Casos de uso

## Uso normal – 1 pc

## Uso normal – 2 pc’s

## Uso esporádico – cio.exe

## Uso esporádico – cio-crypt.exe

# Resultados y conclusiones

- Que pensaba hacer

- que dificultades he encontrado

- cómo lo he arreglado

- qué he conseguido hacer

# Bibliografía y enlaces de interés

blablabla