Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación móvil de gestión de servicios de almacenamiento en la nube

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Kevin Sotomayor Vergara

Tutor: Juan Sánchez Díaz

2017-2018

Agradecimientos

En primer lugar, quiero agradecer la colaboración de mi tutor en este trabajo, el profesor Juan Sánchez Díaz el cual con sus consejos y paciencia ha sabido apoyarme en este trabajo de final de grado en todo momento.

Quiero dedicar todo el esfuerzo de estos últimos 5 años de carrera a mi abuelo Vicente. Gracias por indicarme el camino, por los consejos y por hacer ver cual era mi vocación desde pequeño.

A mis padres y mi hermana, por aguantar todas las veces por las que me han escuchado decir: "Lo siento, tengo que hacer el TFG". Gracias por la paciencia y gracias por apoyarme en todo lo que hago familia.

A mi mejor amiga, mil gracias por hacerme ver cual era el camino adecuado cuando más lo necesitaba. Gracias, por tanto. A mis mejores amigos de la carrera y de vida, gracias por cada consejo, cada ayuda y cada vez que me habéis aguantado durante estos años.

Por último, a mi pareja. Gracias por haber sido un gran apoyo en todo momento.

Resumen

En este trabajo se presenta el desarrollo de una aplicación móvil destinada al sistema operativo iOS que permite la gestión de diferentes servicios de almacenamiento en la nube. El objetivo de este desarrollo es la creación e implantación de un esquema que unifique servicios de almacenamiento remoto. Este trabajo se centra en extraer un modelo común de integración que facilite manejar los diferentes datos de los servicios cloud en una misma lógica de negocio. El modelo resultante es aplicado en el desarrollo de una app de la cual se incluye el análisis, arquitectura, tecnología y otros aspectos importantes para el desarrollo de una aplicación móvil de gestión de nubes.

Palabras clave: desarrollo, iOS, esquema, integración, nube, análisis, arquitectura

Abstract

This paper presents the development of a mobile application for iOS operating system that allows the management of different storage services in the cloud. The aim of this development is to create and implement a scheme that unifies remote storage services. This work focuses on extracting a common integration model which facilities the management of the different data of cloud services in the same business logic. The resulting model is applied in the development of an app which includes analysis, architecture, technology and other important aspects for the development of a mobile cloud management application.

Keywords: development, cloud, iOS, scheme, integration, analysis, architecture

Tabla de contenidos

Índice de ilustraciones

Índice de tablas

1. Introducción

La computación en la nube es un nuevo paradigma de computación que permite la utilización de servicios bajo demanda que residen en Internet en lugar de dispositivos individuales. Las aplicaciones se ejecutan sobre servidores remotos que intercambian información con usuarios que pueden utilizar diferentes dispositivos móviles. La computación en la nube es una nueva opción de negocio para muchas compañías que permiten, entre otras cosas, reducir substancialmente el costo de espacio de almacenamiento local por espacio de almacenamiento en servidores remotos. Dicha característica permite a los usuarios alojar sus archivos en los servidores de los centros de datos de las compañías que ofrecen servicio de almacenamiento en la nube. Las compañías brindan al usuario una serie de opciones para alojar su contenido en la nube y tenerlo accesible en la red en todo momento ya que muchos de los servicios son accesibles a través de distintas plataformas como pueden ser móviles Android, iOS, Windows Phone, Blackberry, ordenadores con macOS o Windows, e incluso desde cualquier navegador web.

En la actualidad existen diversas empresas que ofrecen servicios de almacenamiento en la nube, por ejemplo, entre las más conocidas están Dropbox, Google Drive, OneDrive, iCloud, Mega, Box, Amazon S3, entre otras. Estas empresas ofrecen a los usuarios comprar, alquilar o contratar el espacio de almacenamiento según las necesidades del mismo.

Cada uno de estos servicios ofrece una cantidad básica inicial de espacio de almacenamiento en la nube disponible de manera gratuita y a su vez ofrece diferentes planes de contratación para el usuario. Cada compañía ofrece servicios de todo tipo, desde cuentas básicas con un almacenamiento mínimo, pasando por cuentas Premium que permiten alojar TB de información en la nube sin límites, cuentas de Empresa, licencias de Software con la contratación de espacio de almacenamiento, etc.

Este trabajo describe el desarrollo e implantación de un esquema canónico para la unificación de diferentes servicios de gestión de almacenamiento en la nube. Este esquema ofrece un acceso uniforme a un conjunto de fuentes de datos heterogéneas y autónomas. Una vez especificado el modelo común de datos para la integración, se lleva a cabo el desarrollo de una aplicación móvil que permita hacer uso del mismo, empezando por Dropbox como plataforma inicial para la implementación de dicho esquema.

2. Motivación

Los dispositivos móviles se han convertido en el centro de toda actividad online y por ello muchas de las compañías mencionadas anteriormente ofrecen acceso a sus servicios mediante aplicaciones que se ejecutan en los diferentes sistemas operativos móviles. Los usuarios que cuentan con almacenamiento en la nube suelen tener instalado en sus móviles aplicaciones que permitan acceder a sus archivos y poder así gestionarlos, además, cuando un usuario cuenta con diferentes servicios de almacenamiento en la nube, cuenta también con diferentes aplicaciones que permitan acceder a sus datos lo que conlleva tener varias aplicaciones de diferentes empresas que ofrecen un servicio en común: la gestión de archivos en la nube.

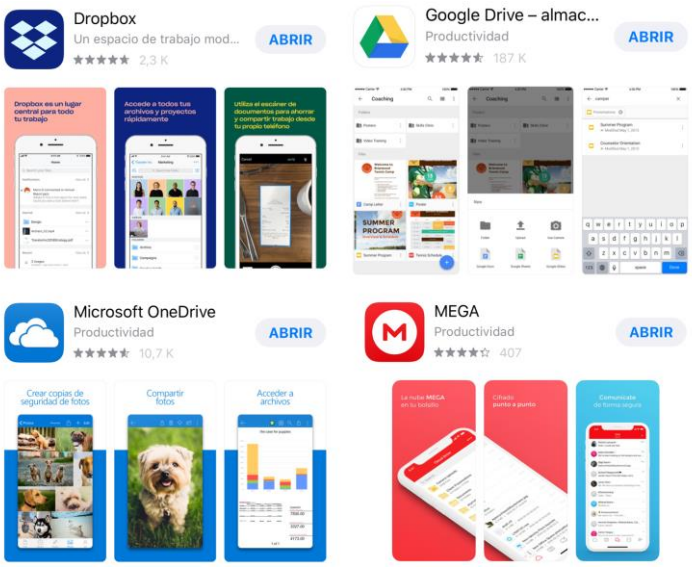

Ilustración 1 Principales aplicaciones de almacenamiento en la nube

La unificación de los principales servicios de almacenamiento en la nube en una sola aplicación que permita hacer la misma gestión que las aplicaciones oficiales, ofrece al usuario hacer una gestión más cómoda, teniendo así un acceso homogéneo para cada uno de sus archivos.

Este trabajo, además del desarrollo de un esquema que comparten diferentes servicios en un único modelo de datos y otra serie de motivaciones técnicas que supone la integración de datos, tiene otras motivaciones de ámbito personal con un gran objetivo, aprender una nueva tecnología de desarrollo móvil.

2.1. Motivación personal

Este trabajo tiene como una de las principales motivaciones ayudarme a crecer como desarrollador y como profesional dedicado al mundo del desarrollo móvil. Desde el año 2011 empecé a programar en lenguajes como Java y Android, los cuales me han permitido abarcar una serie de conocimientos que me han servido de base para la carrera y de cara al ámbito profesional al que me dedico. Este trabajo de final de grado me permitía entonces juntar mis ganas de seguir aprendiendo y la realización de un trabajo

final que tuviese las suficientes motivaciones académicas y personales que buscaba. Fue entonces gracias a la propuesta de trabajo de final de grado de mi tutor, Juan Sánchez Díaz, que tenía la posibilidad de llevar a cabo un desarrollo móvil y en la tecnología con la que me sintiera más cómodo. Swift era el siguiente lenguaje a tener en cuenta para poder extender mis conocimientos y así es como finalmente ha sido elegida como la tecnología a usar en este trabajo.

Swift es el lenguaje de programación de Apple que fue presentado en la WWDC de 2014 que permite desarrollar aplicaciones para el iPhone, iPad, Apple Tv, Mac y Apple Watch gracias a su entorno de desarrollo Xcode que se ejecuta sobre equipos que tienen macOS como sistema operativo.

3. Objetivos

El objetivo principal de este trabajo es la creación e implementación de un esquema canónico que nace a partir del análisis y estudio de las diferentes fuentes de datos relacionadas con los diferentes servicios de almacenamiento en la nube. Una vez definido el esquema se implementará en el desarrollo de una aplicación para dispositivos móviles.

Es trabajo también contiene los siguientes objetivos:

- Proporcionar un mecanismo que permita realizar el uso de consultas de forma uniforme al conjunto de fuente de datos.
- Permitir la escalabilidad del número de fuentes para la integración de diferentes servicios.
- Definir un modelo de lógica de negocio haciendo uso del esquema canónico que se define en este trabajo para ser usado en el desarrollo de una aplicación móvil.
- Desarrollar una aplicación que contemple el funcionamiento básico del uso de gestión de ficheros en la nube y que esté preparada para la implementación sencilla de múltiples fuentes de datos.
- Desarrollar una solución software para móviles que permita una buena mantenibilidad creando una arquitectura en capas que ofrezca una alta cohesión y un bajo acoplamiento.

4. Estado del arte

En este punto del documento se realiza un análisis de la situación actual de aplicaciones que proveen medios similares a los que se lleva a cabo en este trabajo académico. La tecnología escogida para el desarrollo de la aplicación de gestión de nubes es iOS y el sistema operativo de Apple ofrece infinidad de aplicaciones disponibles en la aplicación App Store, la cual permite descargar aplicaciones al iPhone, iPad, Apple Watch y Apple TV.

La App Store es conocida por ser de las primeras aplicaciones que aparecieron como market de aplicaciones. Fue presentada en 2008 con el anuncio del iPhone 3G presentado por Steve Jobs y entonces la tienda de aplicaciones contaba con 500 aplicaciones en su catálogo. El servicio está disponible en 155 países y tiene 500 millones de visitantes a la semana. En la actualidad la plataforma tiene alrededor de dos millones de aplicaciones según datos referentes al primer trimestre de este año 2018.

Existen aplicaciones de todo tipo en la App Store y en ella también se encuentran aplicaciones que ofrecen al usuario una gestión cómoda de sus archivos locales y remotos, por ello se analizará aplicaciones que realicen una funcionalidad similar a la que se describe en este trabajo.

Se ha separado por secciones dichas aplicaciones debido a que algunas de ellas tienen características similares. Las secciones son:

4.1. Gestor de transferencia de ficheros

Las aplicaciones de categorizadas como gestor de transferencia de ficheros hacen referencia a las aplicaciones que presentan una interfaz que permite buscar ficheros en la red y descargarlos para poder hacer uso de los mismos en el almacenamiento local del dispositivo .

4.1.1. Browser

Browser es una aplicación que permite gestionar ficheros del almacenamiento local del dispositivo (sólo los ficheros relacionados con la aplicación Browser) y permite descargar ficheros mediante una URL (dirección de un recurso web) la cual se añade en el apartado de descargas.

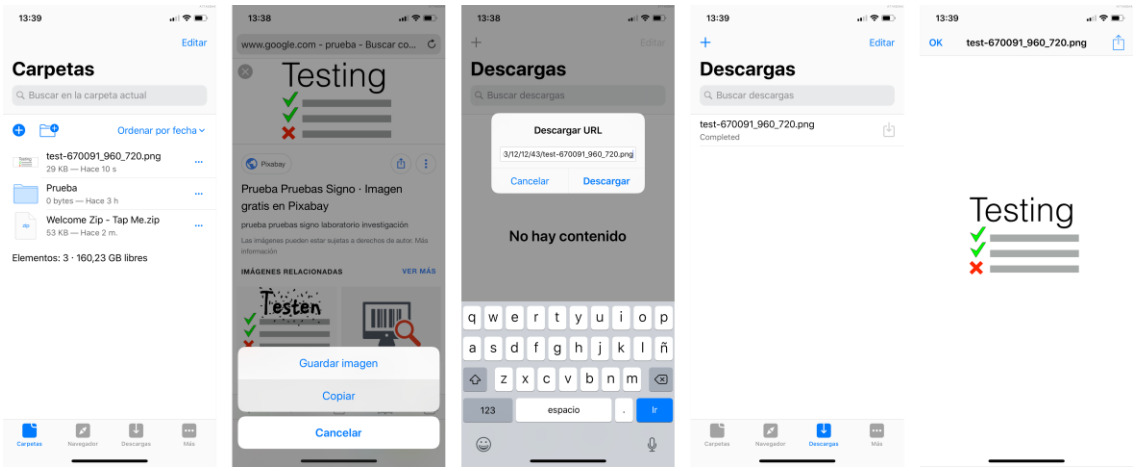

Ilustración 2 Interfaz de la aplicación Browser

Funcionalidades

- Previsualización de ficheros.
- Navegador propio.
- Gestor de descargas.

- Descarga de ficheros.
- Bloqueo con código de seguridad.

Carencias

- No permite añadir servicios de almacenamiento en la nube.
- Sólo gestiona archivos en local.

4.1.2. Files App

Files App es una aplicación que permite gestionar ficheros del almacenamiento local del dispositivo (sólo los ficheros relacionados con la aplicación Files App) y permite descargar ficheros mediante una URL la cual se añade en el apartado de descargas. Gran similitud con Browser a nivel de interfaz y experiencia de usuario.

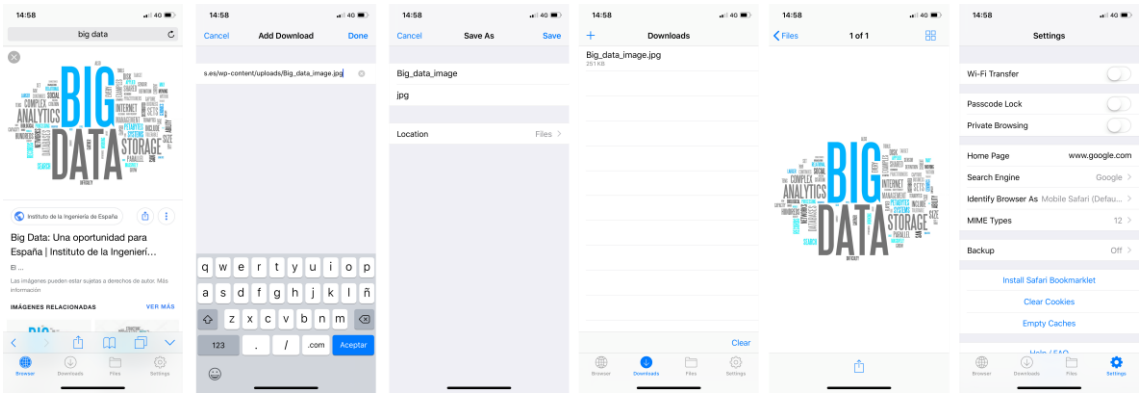

Ilustración 3 Interfaz de la aplicación Files App

Funcionalidades

- Previsualización de ficheros.
- Navegador propio.
- Gestor de descargas.
- Descarga de ficheros.
- Copia de seguridad en iTunes de los ficheros descargados.
- Navegación privada.
- Wifi Transfer. Acceder a la aplicación mediante un navegador para visualizar ficheros y carpetas.
- Bloqueo con código de seguridad.

Carencias

- No permite añadir servicios de almacenamiento en la nube.

- Sólo gestiona archivos en local, no se suben automáticamente cuando se añaden a la aplicación. (Crear copia de seguridad en iTunes automáticamente para un almacenamiento en la nube extra).

4.1.3. File Master

File Master es una aplicación más sencilla que las mencionadas anteriormente. Permite gestionar ficheros locales del dispositivo (imágenes y archivos de audio) y la transferencia de ficheros mediante la red Wifi. Destaca por su sencillez de uso, una interfaz clara y verdaderamente útil.

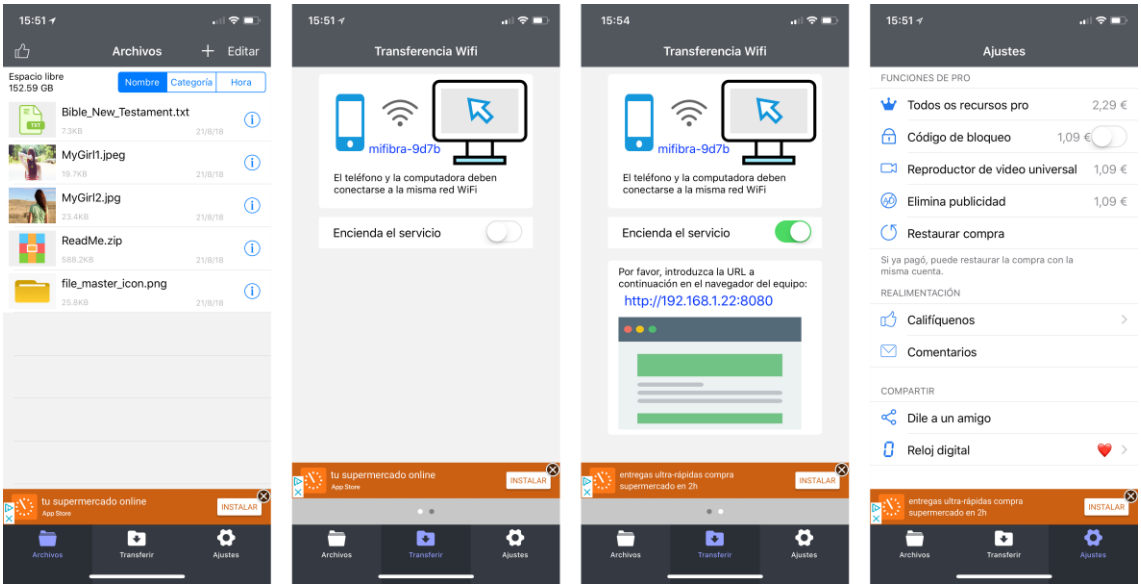

Ilustración 4 Interfaz de la aplicación File Master

Carencias

- No se puede acceder a la dirección IP para conectar mediante el navegador.
- No permite añadir servicios de almacenamiento en la nube.
- Sólo gestiona archivos del dispositivo.

4.2. Gestor de nubes

Las aplicaciones de categorizadas como gestor de nubes hacen referencia a las aplicaciones que presentan una interfaz que permita integrar diferentes servicios de almacenamiento remoto en una misma aplicación. Este tipo de aplicaciones proveen al usuario una interfaz que le permite gestionar y visualizar los ficheros de los diferentes servicios.

4.2.1. Archivos

Archivos es una aplicación nativa de iOS 11 que fue añadida al sistema operativo reemplazando la antigua aplicación iCloud que permitía gestionar los ficheros almacenados en dicha nube. Esta aplicación permite integrar muchos de los servicios de

almacenamiento en la nube en una sola aplicación, funcionalidades similares a lo que se describe en este trabajo, sin embargo, esta aplicación nace a raíz de otro objetivo principal: crear homogeneidad para las aplicaciones basadas en documentos.

Document Based Apps

Este es el nombre de la nueva característica que presentaba Apple en la conferencia de desarrolladores WWDC de 2016. Esta característica provee una interfaz homogénea a muchas otras aplicaciones creadas por Apple que permiten crear y gestionar archivos de tipo presentación, hoja de cálculo y documentos. Esta funcionalidad agregada en la API interna de Swift permite crear aplicaciones que hacen uso de documentos, permitiendo así el uso de esta característica a los desarrolladores para integrar una interfaz común para las aplicaciones que generan documentos con un tipo específico.

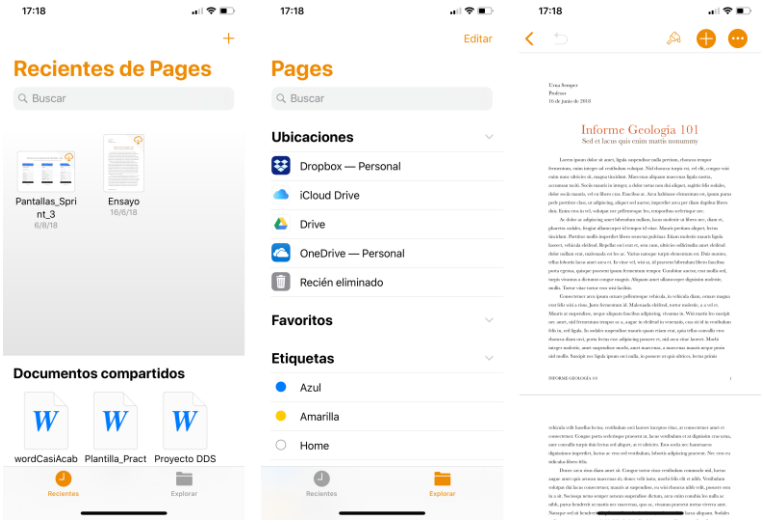

Ilustración 5 Interfaz de la aplicación Pages

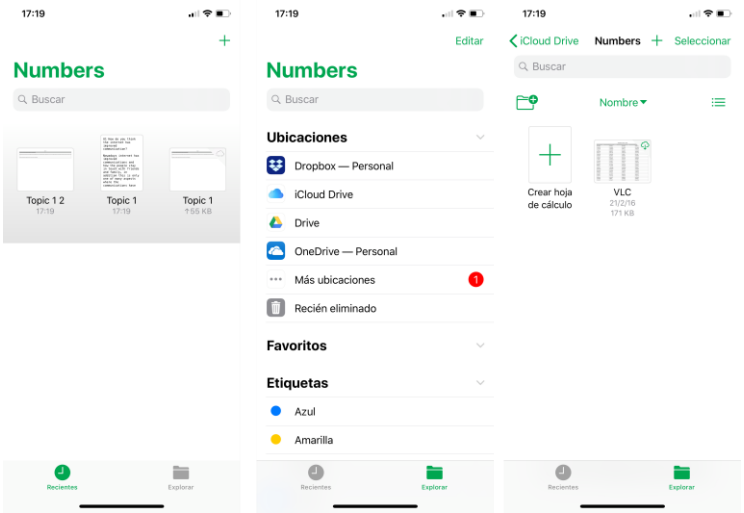

Ilustración 6 Interfaz de la aplicación Numbers

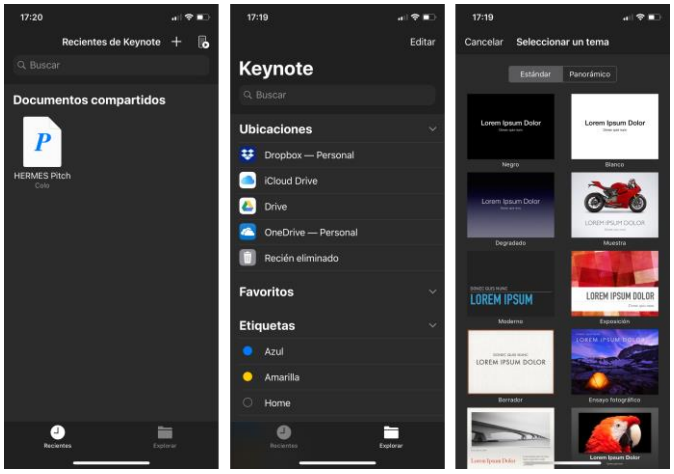

Ilustración 7 Interfaz de la aplicación Keynote

Apple quiere que los desarrolladores integren este tipo de interfaz a las aplicaciones que gestionan documentos, dejando así a la aplicación de Archivos como una plantilla de comunicación entre diferentes servicios para un objetivo común: una gestión de ficheros homogénea que pueda acceder a todos los documentos de tus servicios de almacenamiento en la nube.

Funcionalidades

- Unificación de diferentes servicios de almacenamiento en la nube.
- Interfaz de usuario homogénea independientemente de la nube seleccionada.
- Descarga de ficheros.
- Previsualización de ficheros.
- Gestión CRUD (crear, leer, actualizar y eliminar) de cualquier tipo de documento en la nube.

Carencias

- Buscador ineficaz.
- Una interfaz tan homogénea y similar a la aplicación Archivos conlleva a confusión. La aplicación de Archivos funciona de manera independiente de las aplicaciones que usan una gestión de documentos mediante Document Based Apps, lo cual ofrece una interfaz similar a la de Archivos, pero siendo usada desde una aplicación en sí diferente.
- Las nubes que se pueden integrar necesitan que la aplicación esté instalada en el dispositivo. Este es un problema grave para una aplicación que tiene como objetivo centralizar en una misma aplicación varios de los servicios de almacenamiento en la nube para dar mayor accesibilidad al usuario. Tener instaladas las mismas aplicaciones que luego se usan en la aplicación de Archivos no sirve de nada si lo que se pretende es no instalar estas aplicaciones para tenerlo todo en una sola aplicación. Debido a esto,

Archivos carece de una implementación interna centrada en la integración de datos y no dependiente de las aplicaciones instaladas en el sistema.

4.2.2. Pocket Brief Case

Pocket Brief Case es una aplicación sencilla que combina el almacenamiento local con el almacenamiento en la nube, pero no se la puede considerar una aplicación de transferencia de ficheros debido a que no tiene un gestor de descargas mediante un navegador como las aplicaciones anteriormente mencionadas.

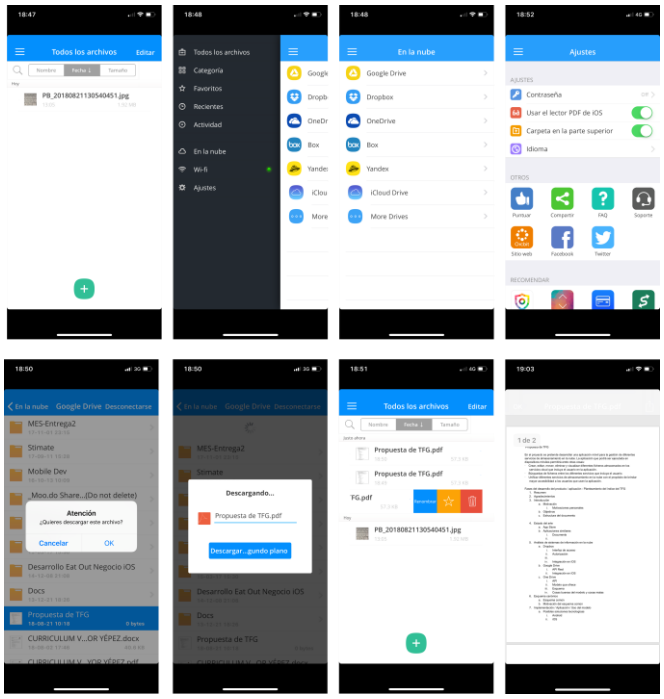

Ilustración 8 Interfaz de la aplicación Pocket Brief Case

Funcionalidades

- Unificación de diferentes servicios de almacenamiento en la nube.
- Uso del almacenamiento local con carpeta dedicada a la aplicación. Esto permite que las imágenes subidas a la aplicación no se fusionen con las del carrete de imágenes de la cámara del dispositivo.
- Descarga de ficheros.
- Previsualización de ficheros.
- Wifi Transfer. Acceder a la aplicación mediante un navegador para visualizar ficheros y carpetas.
- Gestión CRUD de cualquier tipo de documento en la nube.

Carencias

- Interfaz no adaptada a ciertos dispositivos, por ejemplo: iPhone X

- Muchos fallos de conexión con las nubes lo que conlleva una disminución de la fiabilidad hacia el usuario. En la Ilustración 9 se presentan capturas de los fallos de conexión con Dropbox y Google Drive.

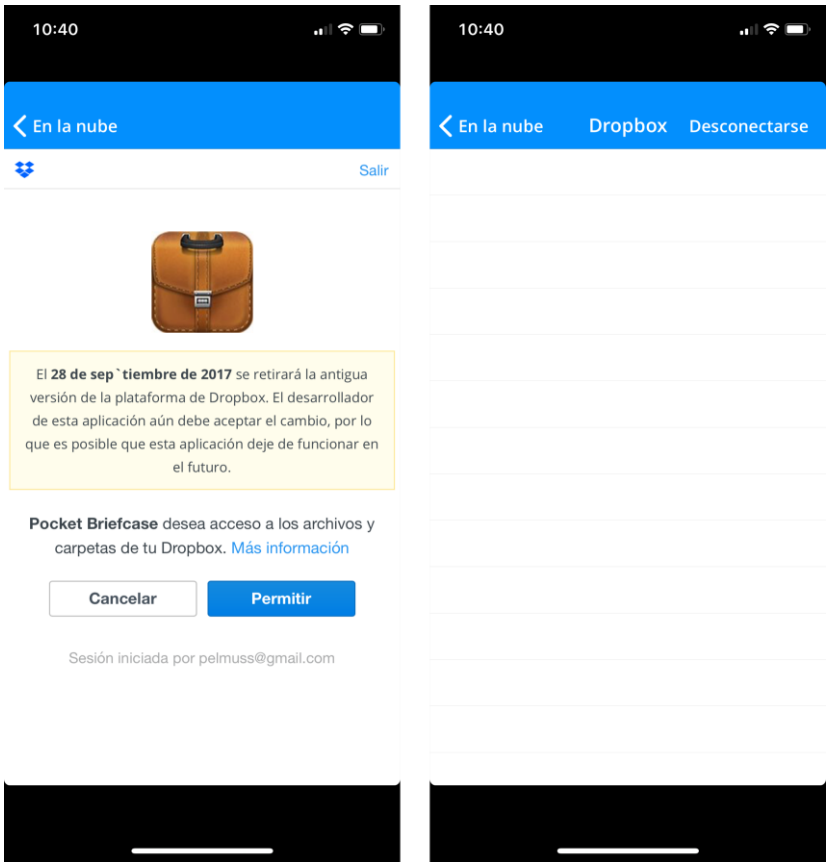

Ilustración 9 Problemas de conexión de nubes en Pocket Brief Case

4.3. Híbridos

Las aplicaciones de categorizadas como híbridos hacen referencia a las aplicaciones que presentan una interfaz que permita integrar diferentes servicios de almacenamiento remoto en una misma aplicación además de tener incluido un gestor de descargas del cual se pueda hacer uso desde un navegador.

4.3.1. Documents

Documents es una aplicación que combina el almacenamiento local con el almacenamiento en la nube y además permite la transferencia de ficheros gracias a su gestor de descargas mediante un navegador como las aplicaciones anteriormente mencionadas.

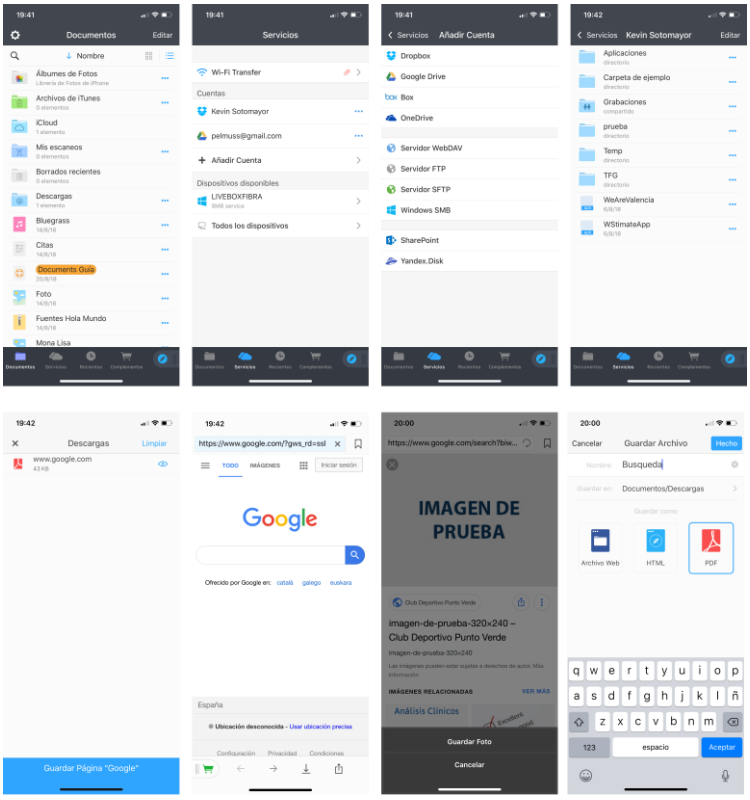

Ilustración 10 Interfaz de la aplicación Documents

Funcionalidades

- Unificación de diferentes servicios de almacenamiento en la nube.
- Uso del almacenamiento local con carpeta dedicada a la aplicación para las descargas de imágenes, .pdf y contenido web.
- Descarga de ficheros en .pdf.
- Previsualización de ficheros.
- Wifi Transfer. Acceder a la aplicación mediante un navegador para visualizar ficheros y carpetas.
- Conexión a servidores FTP y SFTP.
- Conexión a servidores WebDAV.
- Gestión CRUD de cualquier tipo de documento en la nube.

Carencias

- Demasiados archivos de ejemplos de la aplicación en la pestaña de Documentos que pueden llevar a confusión al usuario.
- Gestor de descarga sólo permite imágenes y .pdf

- No permite buscar entre múltiples nubes.
- El desarrollador no da indicios a que se pueden incluir más nubes de las que ya presenta la app.

4.3.2. FileManager

FileManager es una aplicación que permite descargar ficheros mediante su navegador interno el cual tiene acceso a un gestor de descargas donde se visualizan todos los elementos descargados, además esta aplicación ofrece una pestaña donde se pueden conectar diversos servicios como Dropbox, Google Drive, iCloud y OneDrive

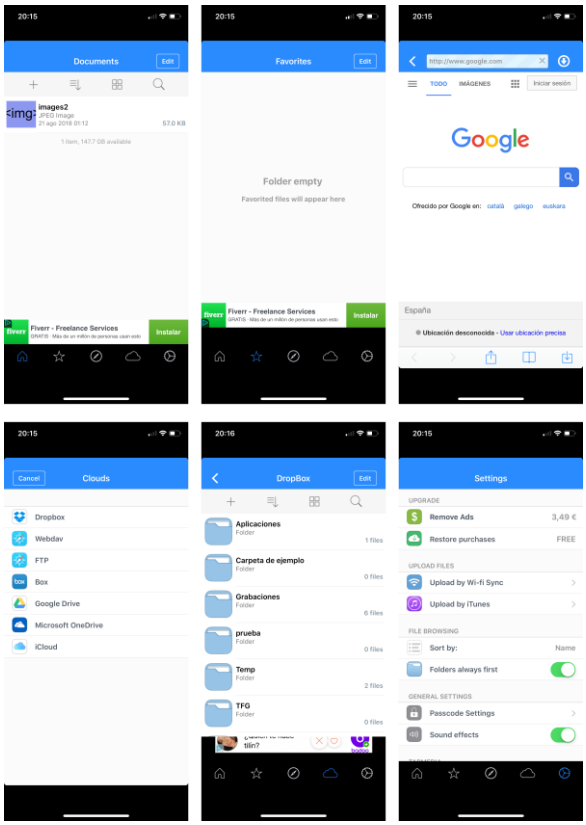

Ilustración 11 Interfaz de la aplicación FileManager

Funcionalidades

- Unificación de diferentes servicios de almacenamiento en la nube.
- Uso del almacenamiento local con carpeta dedicada a la aplicación para las descargas de imágenes.
- Previsualización de ficheros.
- Conexión a servidores FTP.
- Conexión a servidores WebDAV.
- Gestión CRUD de cualquier tipo de documento en la nube.

Carencias

- Interfaz no adaptada a ciertos dispositivos, por ejemplo: iPhone X.
- Gestor de descarga sólo permite imágenes. No se puede añadir una URL para descargar un fichero como las aplicaciones de gestión de ficheros.
- No cuenta con Wifi Transfer. Lo cual permite acceder a la aplicación mediante un navegador para visualizar ficheros y carpetas.

4.4. Crítica al estado del arte

La App Store cuenta con aplicaciones de gran calidad que pasan un proceso complejo de aprobación para la publicación en la tienda de aplicaciones, sin embargo, las aplicaciones analizadas anteriormente cumplen con el objetivo para el cual fueron diseñadas, pero en este trabajo se busca una opción sencilla y directa, que cumpla con los objetivos mencionados anteriormente.

La siguiente tabla muestra un resumen de las carencias importantes para este trabajo.

	Gestor de transferencia de ficheros			Gestor de nubes		Híbridos	
Objetivos	Browser	Files App	Files Master	Archivos	Pocket Brief Case	Documents	File Manager
Incluir múltiples servicios de almacenamiento en la nube	✗	✗	✗	✓	✓	✓	✓
Búsqueda entre múltiples servicios en la nube	✗	✗	✗	✓	✗	✗	✗
Crear, editar, mover, eliminar, visualizar ficheros de múltiples servicios cloud	✗	✗	✗	✓	✓	✓	✓
UI & UX sencilla, no ambigua con unificación de servicios cloud	✗	✗	✗	✗	✗	✓	✓
Escalabilidad de servicios cloud a incluir en la aplicación	✗	✗	✗	✓	✗	✗	✗
No necesidad de aplicaciones oficiales instaladas en el sistema	✗	✗	✗	✗	✓	✓	✓

Tabla 1 Comparativa entre aplicaciones analizadas

4.5. Propuesta

La propuesta de trabajo que se presenta en este documento detalla el análisis y posterior desarrollo de un prototipo de aplicación de gestión de nubes que permita al usuario suplir los las carencias mencionadas en la tabla 1 sobre la comparativa entre las aplicaciones analizadas. El desarrollo servirá como prueba de concepto para hacer uso del modelo común o esquema canónico que se explica en este trabajo. Esta solución permite unificar más de un servicio de almacenamiento en la nube, mejorar la escalabilidad de la inclusión de servicios, hacer uso de guías de estilo que brindan al usuario una UI (Interfaz de usuario) & UX (Experiencia de usuario) sencilla y que cumpla con los objetivos definidos anteriormente.

5. Análisis

En este apartado se define el problema que tiene la integración de diferentes servicios en la nube, los datos. Se define como requisito principal la recopilación de los atributos fundamentales para poder gestionar diferentes servicios con un modelo de datos común. Finalmente se analizan los principales servicios de almacenamiento en la nube para tener en cuenta los datos y metadatos utilizados con el fin de poder extraer una solución que permita interactuar de manera homogénea con una variedad de datos sistemas diferentes.

5.1. Problema

La gestión de diferentes sistemas de almacenamiento de datos en la nube trae consigo el problema de la heterogeneidad semántica de datos. La heterogeneidad semántica en si viene dada por la variedad de servicios a integrar con diferentes modelos de datos, es decir, los diferentes servicios de por si han desarrollado por separado cada uno sus modelos de datos que son diferentes entre sí. Esta problemática hace que en este trabajo se busque ocultar la heterogeneidad semántica, haciendo que servicios distintos aparenten ser un único sistema para una aplicación de gestión de nubes.

5.2. Requisito

Modelo de datos común o esquema canónico es la descripción que se le da a la estructura de datos formada a raíz de un estudio previo del conjunto de datos y metadatos que manejan los diferentes sistemas de información que ofrecen almacenamiento en servidores remotos.

Se tiene como requisito principal permitir la unificación de diferentes servicios de almacenamiento en la nube en un modelo de datos común al que llamaremos a partir de ahora **esquema canónico**. Unificar servicios es una tarea donde se tiene en cuenta cómo se expresan los datos que han de transformarse desde las fuentes al

esquema canónico. La definición de la transformación debe contemplar los siguientes aspectos:

- Nombre de atributos y relaciones entre entidades
    - Propiedades similares con nombres diferentes en esquemas fuente y canónico.
    - Mismos nombres para propiedades que pueden tener significados de conceptos diferentes en cada uno de los esquemas.
- Organización de tablas y modelos.
- Granularidad (mayor / menor nivel de detalle según la necesidad).
- Diversas variantes a nivel de dato (tipo de datos, formatos, escalas, etc.).

El conjunto de todas estas diferencias es lo que se conoce como Heterogeneidad Semántica.

5.3. Fuentes de datos

La aplicación de gestión de servicios de almacenamiento en la nube, en adelante **Oort**, requiere de tantas fuentes de datos como servicios a incluir en la aplicación, sin embargo, para tener un alcance más cerrado, se ha hecho una selección de tres de los servicios más importantes y más usados del mercado de las aplicaciones móviles en la App Store de Apple.

Siendo así elegidas desde el ranking de Apps más descargadas de su categoría de productividad.

Ilustración 12 Apps de nubes más descargadas de la App Store

5.3.1. Google Drive

Google Drive es un servicio de almacenamiento de archivos en la nube propiedad de una de las empresas más grandes del mundo, Google LLC. Drive es el reemplazo de lo que fue la aplicación Google Docs la cual gestionaba documentos en la nube y de colaboración online de tipo Google Docs, Google Slides y Google Spreadsheets (Procesador de texto, presentaciones y hoja de cálculo). Este servicio en la actualidad no sólo gestiona el almacenamiento de ficheros y directorios, sino también unifica el

espacio de almacenamiento de correo electrónico de la aplicación Gmail y las imágenes de la red social Google+.

Este servicio cuenta con una web que ofrece acceso a su API REST en la cual se ofrece documentación para empezar a crear aplicaciones que hacen uso de su API tales como **Drive-enabled apps**, las cuales se integran dentro del espacio personal de cada usuario y permite extender la funcionalidad de la unidad de almacenamiento que ofrece Google Drive. Entre estas apps se encuentran editores de fotos, visualizadores, etc. Aplicaciones clientes que representan información como lo que se explica en este documento, una aplicación que gestiona documentos del espacio personal de un usuario.

Autorización

Este servicio requiere de una previa habilitación de la API de Google Drive, que cuenta con una aplicación llamada Google API Console, la cual administra todas las credenciales de acceso a las APIs y servicios que ofrecen de las diferentes aplicaciones de Google. Para que la aplicación de gestión de nubes pueda acceder a la unidad de almacenamiento de Google Drive es necesaria la previa autorización por parte del usuario, donde se ofrece iniciar sesión para que al acceder se asignen los permisos correspondientes para hacer una gestión de los ficheros.

Ilustración 13 Autorización de uso de Google Drive para la aplicación de gestión de nubes

Acceso a datos

Google pone a disposición de los desarrolladores diversas opciones para que se pueda hacer uso de su API REST, por ello ofrece alternativas para acceder a sus datos desde muchos lenguajes de programación tales como **Java**, **Objective-C**, **JavaScript**, **Go**, **PHP**, **Ruby**, **Node.js**, entre otros muy conocidos por la comunidad de desarrollo. Google Drive puede ser usado desde este tipo de librerías que ofrece Google en su GitHub, pero además cuenta con acceso desde una librería concreta y reducida para Android y el uso normal de una API REST por http (definición de conjunto de métodos de petición en el cual se indica la acción que se desea realizar para un recurso en la red).

Drive también pone a disposición de una librería que permite a los desarrolladores tener acceso a los servicios que proveen datos desde la API de Drive, usa OAuth 2 como protocolo de autorización segura. G Suite API es el SDK que actúa como nexo de conexión a los servicios de la API V2 de Google Drive, ya que G Suite API ofrece acceso a múltiples APIs según la necesidad del desarrollador. Concretamente para el desarrollo en iOS existe G Suite API para Objective-C (Lenguaje predecesor para

programar aplicaciones en iOS). Objective-C es un lenguaje que hoy en día convive con Swift como lenguaje de programación para aplicaciones iOS. Se puede integrar a través de CocoaPods o Carthage en un proyecto en Xcode, se puede encontrar más información en el GitHub dedicado a este proyecto.

Modelo de datos

A través del SDK de la plataforma que pone a disposición Google, en este caso la Suite de Google para desarrollo iOS, Google Drive ofrece información sobre cómo representa la información del contenido de Drive como **File** (Fichero). File es un conjunto de metadatos y contenido que puede ser añadido, actualizado, copiado y borrado.

Ficheros

Cada fichero es identificado por un identificador único. Estos identificadores son estables y permanecen activos hasta que el fichero cumple su ciclo de vida (desde que se crea hasta que es eliminado, en el caso de ser eliminado por completo) incluso si éste cambia de nombre.

Metadatos

Cada fichero contiene metadatos que describen su contenido. Estos metadatos incluyen el nombre del fichero, tipo, fecha de creación y veces que se ha modificado.

Permisos

Todos los ficheros tienen propietario. Los usuarios controlan quienes acceden a sus ficheros y crean listas de control de acceso para cada elemento. Se definen ciertos roles que permiten ciertas operaciones como leer el contenido de un fichero, leer metadatos, etc. Según el tipo de rol. Entre los roles están organizador / propietario, escritor, comentarista, lector. Se puede saber más en la guía que ofrece Drive sobre permisos: **About Permissions**.

Contenido

Los ficheros en algunos casos pueden tener contenido binario o texto asociado que se puede cargar o descargar. Imágenes o vídeos, archivos de texto o PDFs son un claro ejemplo de ficheros que tienen contenido de este tipo y esto permite que Google Drive pueda indexarlos mediante machine learning (aprendizaje automático) para identificar objetos y lugares en las imágenes.

Revisiones

Los ficheros cuentan con un histórico del contenido asociado al cual se le denomina revisión. Las revisiones se van actualizando y no se almacenan indefinidamente. Se puede llegar a incluir versiones fijas si es necesario para evitar que se actualice este apartado.

La ilustración 14 muestra a modo ilustrativo la forma básica y acortada de un File – Fichero de Google Drive.

Fichero Google Drive

```
{
  "kind": "drive#file",
  "id": string,
  "name": string,
  "mimeType": string,
  "description": string,
  "starred": boolean,
  "trashed": boolean,
  "explicitlyTrashed": boolean,
  "trashingUser": {
    (key) : string
  },
  "trashedTime": datetime,
  "parents": [
    string
  ],
  "properties": {
    (key): string
  },
  "appProperties": {
    (key): string
  },
  "spaces": [
    string
  ],
  "version": long,
  "webContentLink": string,
  "webViewLink": string,
  "iconLink": string,
  "hasThumbnail": boolean,
  "thumbnailLink": string,
  "thumbnailVersion": long,
  "viewedByMe": boolean,
  "viewedByMeTime": datetime,
  "createdTime": datetime,
  "modifiedTime": datetime,
  "modifiedByMeTime": datetime,
  "modifiedByMe": boolean,
  "sharedWithMeTime": datetime,
  "sharingUser": {
      (key): string
  },
  "owners": [
    {
      (key): string
    }
  ],
  "teamDriveId": string,
  "lastModifyingUser": {
    "kind": "drive#user",
    "displayName": string,
    "photoLink": string,
    "me": boolean,
    "permissionId": string,
    "emailAddress": string
  },
  "shared": boolean,
  "ownedByMe": boolean,
  "capabilities": {
    (key): string
  },
  "viewersCanCopyContent": boolean,
  "copyRequiresWriterPermission": boolean,
  "writersCanShare": boolean,
  "permissions": [
    permissions Resource
  ],
  "permissionIds": [
    string
  ],
  "hasAugmentedPermissions": boolean,
  "folderColorRgb": string,
  "originalFilename": string,
  "fullFileExtension": string,
  "fileExtension": string,
  "md5Checksum": string,
  "size": long,
  "quotaBytesUsed": long,
  "headRevisionId": string,
  "contentHints": {
    "thumbnail": {
      "image": bytes,
      "mimeType": string
    },
    "indexableText": string
  },
  "imageMediaMetadata": {
    (key): string
  },
  "videoMediaMetadata": {
    (key): string
  },
  "isAppAuthorized": boolean
}
```

Ilustración 14 Datos de File - Fichero de Google Drive

Miniaturas

Google Drive automáticamente genera miniaturas para muchos tipos de contenido, sin embargo, para accesos directos y otros tipos de archivos similares Drive no puede generarlos.

Tipo de ficheros

Todos los archivos de Drive contienen la misma estructura, la de un File, sin embargo, su comportamiento y características dependen del tipo de archivo.

Blobs

Los blobs son archivos que contienen texto o contenido binario, como imágenes, videos y archivos PDFs. El contenido blob se puede cargar o descargar tal cual o en algunos casos se puede convertir a Google Docs, Hojas de cálculo o Presentaciones.

Carpetas

Las carpetas son contenedores utilizados para organizar el contenido del usuario en Google Drive. Las carpetas sólo contienen metadatos y no se pueden cargar ni descargar, pero puede actuar como padres de otros archivos y carpetas.

Atajos

Los atajos son accesos directos que en si son ficheros de un tipo especial utilizados para vincular contenido de terceros a Drive. Los atajos sólo contienen metadatos y no pueden cargarse o descargarse a través de Google Drive.

Documentos de Google

Estos documentos son archivos de tipo Google Docs, Hojas de cálculo, Presentaciones y otras aplicaciones similares propias de Google. El contenido puede ser importado o exportado a través de formatos compatibles, ya que se puede cargar un PDF y convertirlo a un documento de Google Docs o una Presentación de puede ser exportada a un archivo .pptx o más conocido como una presentación de PowerPoint

Restricciones

A diferencia de otros servicios, Google es bastante permisivo en el momento de prestar sus servicios a los desarrolladores. Google cuenta con una política de precio por petición y uso bastante amplia. Esta política empieza con una cuota inicial de 1.000.000.000 de peticiones al día. 10 peticiones por segundo por cada usuario que use los servicios.

La consola de desarrolladores de Google permite ver las cuotas según qué proyecto en la pestaña de Cuotas donde se indica la cuota inicial por cada uno de ellos y se puede modificar.

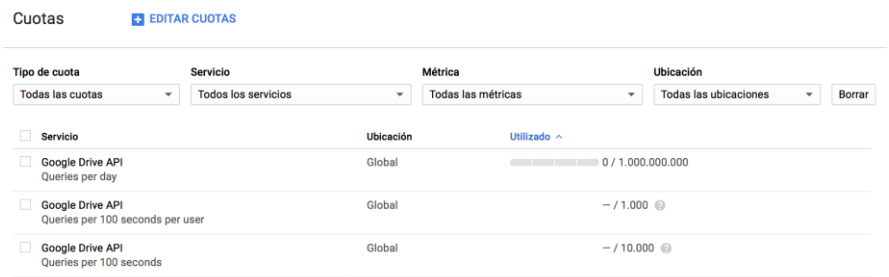

Ilustración 15 Cuotas de uso de Google Drive

5.3.2. Dropbox

Dropbox es uno de los servicios de alojamiento de archivos en la nube más conocidos del mundo. Dropbox es parte de la compañía Dropbox Inc. la cual cuenta con más de 500 millones de usuarios registrados por todo el mundo. En la actualidad dispone de aplicaciones móviles para Android, Windows Phone, Blackberry e iOS.

Este servicio cuenta con una web dedicada para los desarrolladores que pretenden hacer uso de sus API's. Existen dos tipos de API, una para acceder a cuentas de Dropbox Personales y otra para cuentas de Dropbox Bussines donde se permite acceder a las funcionalidades preparadas para empresas que ofrece la compañía.

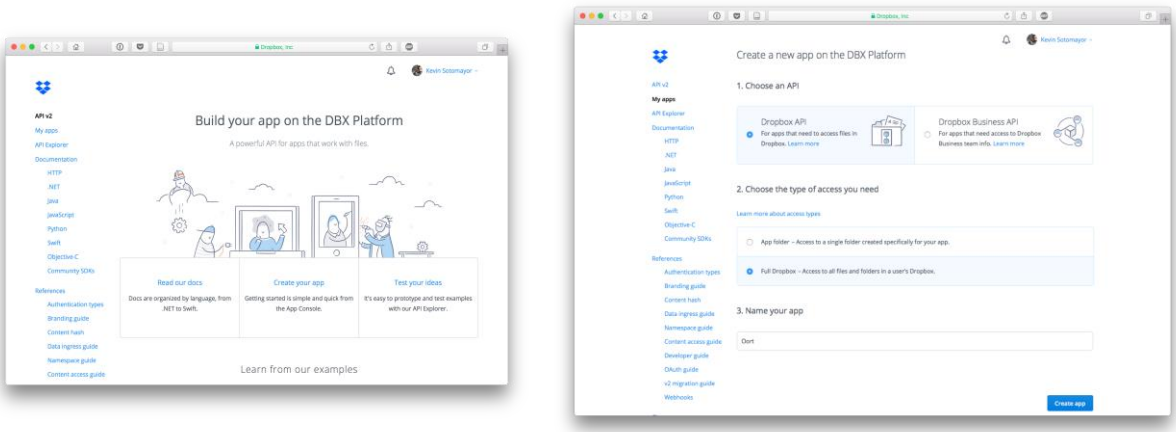

Ilustración 16 Sitio web para desarrolladores - Dropbox

Autorización

Gracias a este portal dedicado a los desarrolladores, Dropbox permite dar de alta una aplicación para poder hacer uso de lo que ofrece sus API's. Para hacer uso de todo lo que puede ofrecer Dropbox como servicio es necesario una **App Key**, ésta es un conjunto de caracteres que hace referencia a la aplicación que va a hacer uso de los datos en el momento que se quiere acceder con la cuenta de Dropbox para poder acceder al servicio, en otras palabras, en el momento en el que se ha de querer introducir la cuenta de Dropbox para poder visualizar los archivos, se necesita dar permisos a la aplicación referenciada con esta App Key para que ésta pueda hacer uso de los datos existentes en Dropbox.

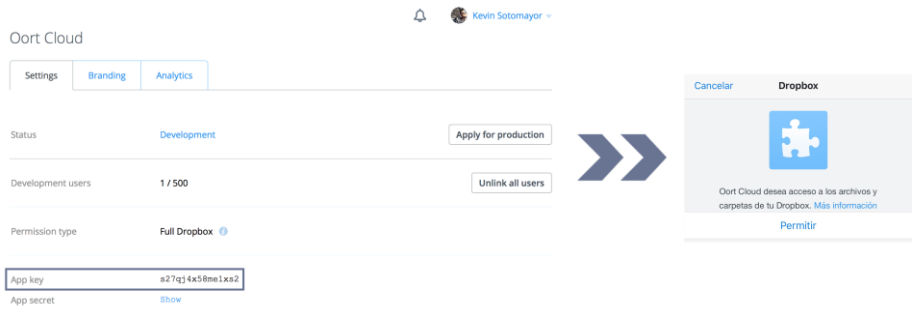

Ilustración 17 Autorización de uso de Dropbox para aplicación de gestión de nubes9

Acceso a datos

Para el desarrollo de la aplicación de gestión de nubes para sistemas iOS, Dropbox pone a disposición de una librería que permite a los desarrolladores tener acceso a los servicios que proveen datos desde el API de Dropbox, la cual usa OAuth 2.0 como protocolo de autorización. SwiftyDropbox es el SDK / librería que permite integrar la API V2 de Dropbox tanto para desarrollos en iOS como en macOS. Se puede integrar a través de CocoaPods o Carthage en un proyecto en Xcode, se puede encontrar más información en el GitHub dedicado a este proyecto.

La web para desarrolladores de Dropbox también ofrece más plataformas sobre las que poder hacer uso de diferentes SDK según el lenguaje de programación.

Ilustración 18 Plataformas para el uso de las API de Dropbox

Modelo de datos

A través del SDK de la plataforma elegida, en este caso SwiftyDropbox para desarrollo iOS, Dropbox ofrece una documentación del SDK en la cual se especifica una serie de

objetos y métodos de acceso a los datos de manera asíncrona. Esta permite extraer información de ficheros y directorios como lo indica su documentación.

A diferencia de Google Drive, Dropbox es un sistema de almacenamiento que no tiene en cuenta diversos tipos de fichero ya que Google Drive está enfocado a guardar aplicaciones generadas a través de la suite de Drive, Google Docs, Hojas de Cálculo, Presentaciones, etc. De igual manera, Dropbox también cuenta con una serie de datos y metadatos similares a los de Google Drive entre los cuales se puede observar información básica de ficheros como identificadores, rutas, nombre, metadatos complementarios, información de permisos, información de elemento compartido, entre otros.

Dropbox ofrece Files.Metadata como el objeto principal para representar archivos, ya sean carpetas o ficheros, donde se puede observar información compartida para ambos tipos de datos el cual dispone de propiedades en común como son el nombre, la ruta en mayúscula y minúscula, información sobre compartidos y descripción que alberga los metadatos internos de un fichero y una carpeta en concreto además de la información extra que tenga estos meta-metadatos.

Ilustración 19 Datos básicos para carpetas y ficheros - Dropbox

- Para los **ficheros**, SwiftyDropbox devuelve resultados simples en los cuales se incluye información básica de ficheros tales como el nombre, identificador, ruta, entre otros atributos. Se puede ver un ejemplo en la ilustración 20.

Ilustración 20 Datos de ficheros - Dropbox

- Para los **directorios simples**, SwiftyDropbox devuelve resultados simples en los cuales se incluye información básica de directorios tales como la ruta, identificador y nombre. Se puede ver un ejemplo en la ilustración 21.

Ilustración 21 Datos de directorio no compartido – Dropbox

- Para los **directorios compartidos**, SwiftyDropbox devuelve información extra además de la que se puede obtener cuando se hace la petición de un directorio simple. Esta información está incluida en el objeto principal como una información compartida. Se puede ver un ejemplo en la ilustración 22.

Ilustración 22 Datos de directorio compartido - Dropbox

Tanto ficheros como directorios cuentan con un atributo **path_display** y **path_lower**, y es que Dropbox prevé posibles problemas para las aplicaciones que almacenan metadatos de los archivos de los usuarios en bases de datos (como SQLite o Postgres) que distinguen entre mayúsculas y minúsculas, sin embargo, Dropbox hace un esfuerzo por preservar la distinción y en el caso correcto el path_display de los metadatos contendrá el caso correcto de ubicación del fichero / directorio correspondiente.

Desde la ilustración 19 hasta la 22 se muestra a modo ilustrativo la forma básica de objetos que contienen información sobre ficheros y carpetas de Dropbox.

Restricciones

Existe una serie de restricciones a la hora de crear una aplicación que use los servicios de la API de Dropbox debido a la cantidad de peticiones que se pueden llevar a cabo desde la aplicación que usa dichos servicios. Cuando se crea una aplicación que usa la API de Dropbox se le asigna un estado, esta aplicación funcionaría igual que una aplicación con estado de producción, excepto que solo se puede vincular con hasta 500 usuarios de Dropbox en total. Cuando se alcanzan los primeros 50 usuarios, Dropbox ofrece la posibilidad de poder solicitar un estado para usar en producción,

lo cual quiere decir que la aplicación que usa estos servicios está preparada para ser presentada en el mercado de aplicaciones para que pueda ser descargada por todo el mundo.

Si se supera este límite impuesto por Dropbox, éste congela la capacidad de uso de la aplicación que usa el API de Dropbox ya que la aplicación necesitaría un estado de producción que permita extender su uso a más usuarios.

Cuando se solicita el estado de producción la compañía impone una serie de pautas para los desarrolladores, estas pautas pueden ser información sobre el uso de la API de Dropbox, icono y guía de diseño a seguir que ofrece Dropbox, cumplimentación de términos y servicios, entre otros.

5.3.3. OneDrive

OneDrive es lo que antes se conocía como SkyDrive de Microsoft, es un servicio de almacenamiento de archivos en la nube de la empresa Microsoft. Estrenado en febrero de 2014, OneDrive ofrece un servicio de almacenamiento en la nube y la suite ofimática de Microsoft Office en ciertos paquetes del servicio. La suite ofimática de Microsoft Office incluye programas como Word, Excel, PowerPoint, entre otros y permiten almacenar la información en la nube y la colaboración en documentos online.

OneDrive Dev Center es la web para desarrolladores que ofrece Microsoft para crear aplicaciones que usen los servicios que provee OneDrive. La API REST de OneDrive es parte de la API de Microsoft Graph que permite conexiones al contenido de los servicios de OneDrive y SharePoint. Microsoft Graph es una API común de los servicios de Microsoft, una solución bastante parecida a la que ofrece Google con su G Suite APIs y su acceso como interfaz común para toda su API REST.

Ilustración 23 API de Microsoft Graph

Autorización

El servicio de OneDrive, al igual que los otros servicios mencionados anteriormente, requiere tener una app ID y registrar la aplicación que va a hacer uso de los servicios de OneDrive. Para registrar una nueva aplicación hace falta ir la aplicación Microsoft

Application Registration y obtener un identificador para añadirlo en el código de la aplicación se vaya a realizar. Es necesario tener una cuenta Microsoft o tener una cuenta de estudiante.

Registrar una aplicación para obtener un ID es muy sencillo desde la web donde se registran las aplicaciones de Microsoft, similar a lo que provee Google con su Consola para añadir proyectos según qué tipo de API se va a utilizar, pero algo más sencillo, basta con introducir el nombre de la aplicación y Microsoft asigna un Application Id que se puede utilizar para poder autorizar a la aplicación de gestión de nubes a que tenga acceso al contenido que almacena esta nube. Como se aprecia en la ilustración 24, una vez se inicia sesión y se autoriza a la aplicación para que pueda generar peticiones seguras gracias al uso de OAuth 2.0 como protocolo de autorización.

La ilustración 24 proviene de un ejemplo obtenido de la web de Microsoft el cual representa un proyecto de ejemplo que se puede generar para que pueda ser usado de referencia para los desarrolladores una vez se ha completado los datos que permiten generar un identificador de aplicación.

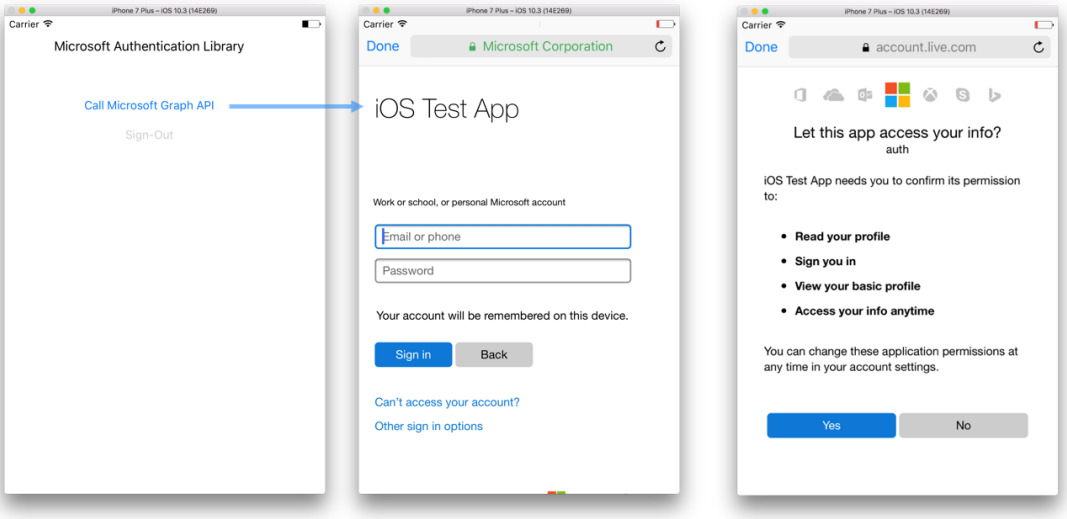

Ilustración 24 Autorización de uso de OneDrive –Documentación de Microsoft

Acceso a datos

Como se ha visto en la ilustración 23, Microsoft ofrece a los desarrolladores muchas opciones para implementar y hacer uso de su Graph API, por ello ofrece alternativas para acceder a sus datos desde muchos lenguajes de programación tal y como se muestra en la ilustración 25. Microsoft Graph es un SDK, Software Development Kit o conjunto de herramientas que se ofrecen a los desarrolladores para hacer uso de ciertos servicios y funcionalidades. Se encuentra en la web de Graph donde se puede hacer uso de un Quick Start (inicio rápido con ejemplo de prueba) en el cual se selecciona la tecnología a usar y genera un proyecto de ejemplo que se puede abrir con el IDE de la tecnología escogida.

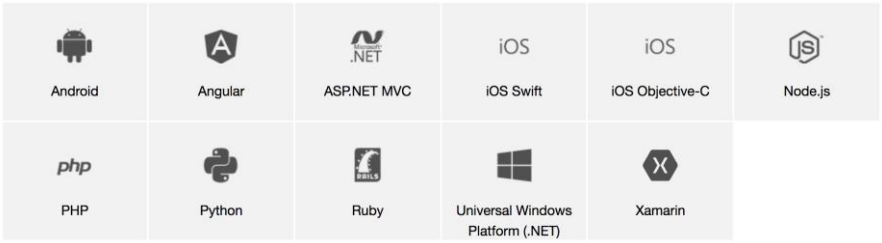

Ilustración 25 Plataformas compatibles para el uso de OneDrive

Este proyecto generado por la web del Quick Start se autodescarga una vez se ha generado el identificador de la aplicación; después de descargarse el .zip, mediante CocoaPods se puede generar un proyecto en Xcode. El proyecto Xcode contiene el código con la cual se obtiene la autorización para dar paso al uso de los servicios que ofrece Graph API y su acceso a OneDrive.

Modelo de datos

OneDrive dispone del tipo de recurso **driveItem**, el cual es una representación de un archivo del usuario. Los archivos pueden ser ficheros, carpeta u otro elemento almacenado en una unidad.

s recursos **driveItem** cuentan con propiedades del tipo **facet**, las cuales proporcionan datos sobre las identidades del driveItem. Por ejemplo, las carpetas tienen **un folder facet**, un fichero tiene un **file facet**, una imagen tiene un **image facet** además de su file facet correspondiente que indica que es un fichero sólo que se especifica con más detalle qué tipo de fichero es.

Los elementos con facet de tipo folder actúan como contenedores de elementos y por lo tanto tienen referencia a un hijo que indica que esa carpeta contiene una colección de driveItems**.**

Este recurso driveItem es un recurso que hereda información de un objeto base llamado **BaseItem**, el cual cuenta con información básica para recursos que derivan del mismo.

Recurso BaseItem - OneDrive

```
{
  "id": "string (identifier)",
  "createdBy": { "@odata.type": "microsoft.graph.identitySet" },
  "createdDateTime": "datetime",
  "description": "string",
  "eTag": "string",
  "lastModifiedBy": { "@odata.type": "microsoft.graph.identitySet" },
  "lastModifiedDateTime": "datetime",
  "name": "string",
  "parentReference": { "@odata.type": "microsoft.graph.itemReference" },
  "webUrl": "url"
}
```

Ilustración 26 Recuso BaseItem - OneDrive

Como se ve en la ilustración 27, un recurso driveItem contiene todas las propiedades que puede tener un archivo ya sea un fichero o carpeta que esté almacenada en la nube personal de OneDrive.

Se puede observar en dicho objeto que algunos tipos de datos @odata.type son referencias a otros objetos de Microsoft Graph los cuales están definidos dentro del SDK y por ello no se ve un tipo de dato concreto.

Recurso DriveItem - OneDrive

```
{
  "audio": { "@odata.type": "microsoft.graph.audio" },
  "cTag": "string (etag)",
  "deleted": { "@odata.type": "microsoft.graph.deleted"},
  "description": "string",
  "file": { "@odata.type": "microsoft.graph.file" },
  "fileSystemInfo": { "@odata.type": "microsoft.graph.fileSystemInfo" },
  "folder": { "@odata.type": "microsoft.graph.folder" },
  "image": { "@odata.type": "microsoft.graph.image" },
  "location": { "@odata.type": "microsoft.graph.geoCoordinates" },
  "package": { "@odata.type": "microsoft.graph.package" },
  "photo": { "@odata.type": "microsoft.graph.photo" },
  "remoteItem": { "@odata.type": "microsoft.graph.remoteItem" },
  "root": { "@odata.type": "microsoft.graph.root" },
  "searchResult": { "@odata.type": "microsoft.graph.searchResult" },
  "shared": { "@odata.type": "microsoft.graph.shared" },
  "sharepointIds": { "@odata.type": "microsoft.graph.sharepointIds" },
  "size": 1024,
  "specialFolder": { "@odata.type": "microsoft.graph.specialFolder" },
  "video": { "@odata.type": "microsoft.graph.video" },
  "webDavUrl": "string",

  /* relationships */
  "content": { "@odata.type": "Edm.Stream" },
  "createdByUser": { "@odata.type": "microsoft.graph.user" },
  "lastModifiedByUser": { "@odata.type": "microsoft.graph.user" },
  "children": [ { "@odata.type": "microsoft.graph.driveItem" }],
  "thumbnails": [ {"@odata.type": "microsoft.graph.thumbnailSet"}],
  "permissions": [ {"@odata.type": "microsoft.graph.permission"} ],

  /* inherited from baseItem */
  "id": "string (identifier)",
  "createdBy": {"@odata.type": "microsoft.graph.identitySet"},
  "createdDateTime": "String (timestamp)",
  "eTag": "string",
  "lastModifiedBy": {"@odata.type": "microsoft.graph.identitySet"},
  "lastModifiedDateTime": "String (timestamp)",
  "name": "string",
  "parentReference": {"@odata.type": "microsoft.graph.itemReference"},
  "webUrl": "string",

  /* instance annotations */
  "@microsoft.graph.conflictBehavior": "string",
  "@microsoft.graph.downloadUrl": "url",
  "@microsoft.graph.sourceUrl": "url"
}
```

Ilustración 27 Recurso DriveItem - OneDrive

Restricciones

Microsoft Graph restringe el número de peticiones simultáneas a un servicio para evitar un uso excesivo de sus recursos. Graph admite un gran volumen de solicitudes, sin embargo, los márgenes de limitación varían según los escenarios. Por ejemplo, si se produce una gran cantidad de escrituras, la posibilidad de limitación es más grande que si sólo se va a realizar lecturas.

Cuando se produce un desbordamiento del umbral de peticiones, Microsoft Graph limita las solicitudes de ese cliente durante un periodo de tiempo. Los límites del umbral varían según el tipo de solicitud y en si las solicitudes tienen que estar equilibradas según su tipo.

Microsoft desde su web recomienda cierto tipo de acciones a tener en cuenta según los casos en los que se supera el umbral de peticiones, sin embargo, no expone la

cantidad exacta de posibles peticiones que se hagan a la API de Microsoft. Entre los procedimientos a tener en cuenta para tratar las limitaciones se encuentran reducir el número de operaciones por solicitud, reducir la frecuencia de las llamadas e incluso evitar reintentos inmediatos, dado que las solicitudes se acumulan contra los límites de uso. También ofrece un control de errores a través del error http 429 que permite detectar la limitación hacer un mejor tratamiento de la misma.

5.4. Solución

La ocultación de la heterogeneidad semántica pasa por el estudio previo de las fuentes de datos. Como se ha podido observar en los puntos anteriores, cada fuente de datos tiene un modelo diferente al otro, pasando por Google Drive y su elemento File, Dropbox con su SDK SwiftyDropbox ofreciendo una versión más reducida de su API para su implementación en iOS, OneDrive y su objeto driveItem y datos propios de Microsoft Graph, etc.

Estas fuentes de datos cuentan con datos básicos comunes entre sí, sin embargo, se ha de tener en cuenta que no todos estos servicios de almacenamiento en la nube llaman a las propiedades de un modelo de la misma manera, o incluso, una fuente de datos tiene propiedades que otro no. Por ejemplo, se puede observar que Dropbox cuenta con **path_display** y **path_lower**, pero Google Drive y OneDrive no cuentan con ninguna propiedad **path (ruta)** que indique la ruta de dicho archivo, estos usan referencias a los ficheros padres mediante una propiedad **parentReference**, la cual permitiría saber dónde se encuentra dicho archivo.

La solución a la variedad de datos y sintaxis con la que cuenta cada servicio para crear una aplicación cliente que muestre transparencia al usuario de los servicios incluidos, implica crear un esquema global a partir de las propiedades de las fuentes de datos que permita la integración de múltiples fuentes, que pueda ser escalable en el tiempo y según la cantidad de servicios a incluir y que ofrezca un acceso uniforme a un conjunto de fuente de datos heterogénea.

5.4.1. Arquitectura de integración

La solución y el desarrollo de un esquema canónico requiere de un tipo de integración de datos que se denomina Integración de datos virtual. Gracias a esto, los datos permanecen en la fuente y se accede a ellos cuando se realiza la consulta de tal manera que no se gastan recursos en integración de datos que el usuario no usará hasta la próxima vez que este lo requiera.

Para este tipo de casos no es valida una arquitectura Warehousing, la cual permite la integración de datos en un repositorio único, ya que no se pretende construir un almacén de datos de diversas fuentes para ser consultadas. Este tipo de arquitecturas cuenta con un repositorio que puede ser físico o lógico y hace énfasis en la integración de datos de diferentes fuentes para fines analíticos y de consulta.

5.4.2. Mapping semántico

El mapping semántico establece la relación entre el esquema fuente y el esquema resultante de la solución que se aplicará para la integración de datos entre los diferentes servicios de almacenamiento. Esta solución resultante se denomina

esquema canónico, el cual actúa de único sistema visible para el usuario, por ello, este esquema necesita de las propiedades necesarias extraída de las fuentes de datos para cumplir con las funcionalidades básicas de una aplicación gestor de nubes tales como crear un archivo, modificarlo, moverlo, eliminarlo y acceder a su información.

En las siguientes tablas se muestra qué tipos de datos se necesitan para elaborar ciertas peticiones. Las tablas identifican los distintos servicios, las propiedades que usa cada uno para gestionar dicha petición y lo que se atribuye al esquema canónico final según la necesidad de las peticiones.

Crear Archivos	Fuente de datos	Propiedades en común
Dropbox	Path	Name
	Name	Path
Google Drive	Title	MimeType
	MimeType	ParentId
	Parents	CreatedDate
	CreatedDate	ModifiedDate
	ModifiedDate	
OneDrive	FileName	
	ParentReference	

Tabla 2 Mapping semántico - Crear archivos

Actualizar archivos	Fuente de datos	Propiedades en común
Dropbox	Path	Id
	Name	Path
Google Drive	FileId	Name
	Name	
OneDrive	Id	
	Name	

Tabla 3 Mapping semántico - Actualizar archivos

Mover Archivos	Fuente de datos	Propiedades en común
Dropbox	Path	Id
	Name	Path
Google Drive	Id	Name
	ParentId	ParentId
OneDrive	Id	
	ParentReference	
	Name	

Tabla 4 Mapping semántico - Mover archivos

Eliminar archivos	Fuente de datos	Propiedades en común
Dropbox	Path	Id
Google Drive	FileId	Path
OneDrive	Id	

Tabla 5 Mapping semántico - Eliminar archivos

Como se puede observar, hasta este punto existen datos que semánticamente hacen referencia a lo mismo, como bien puede ser FileId o Id, ParentReference o ParentId, sin embargo, cada servicio le ha dado un nombre diferente según se ha diseñado. Servicios como Dropbox contemplan el path como un identificador para mover, actualizar, crear e incluso eliminar ficheros, pero como se muestra en las ilustraciones a partir de la ilustración 20, Dropbox devuelve un identificador en los metadatos de cada respuesta de las peticiones a un archivo ya sea una carpeta o fichero.

Según la documentación oficial de Dropbox, se puede construir una ruta relativa con el ID proporcionado de una carpeta usando una barra inclinada /. Por ejemplo, se quiere acceder a la carpeta de nombre **temp** con **id:abc123xyz** al fichero **hello.txt**, se puede acceder también de la siguiente manera: **id:abc123xyz/hello.txt**

El Mapping semántico logra disolver estas diferencias. Como se puede observar en la tabla 5, que es un ejemplo sencillo para ilustrar, para eliminar un archivo es necesario la ruta (path) o su identificador (FileId, Id) dependiendo del origen del archivo se procederá a utilizar una propiedad u otra en el momento de realizar una petición para realizar dicha acción de eliminar. Es necesario entonces contar en todo momento con ambas propiedades para que el esquema canónico pueda abordar las diversas fuentes de datos.

En la tabla 6 se definen otra serie de propiedades en común de cada servicio que sirve para mostrar información al usuario sobre los archivos, ya sea una carpeta o un fichero, que visualice en el cliente móvil.

Leer archivos	Fuente de datos	Propiedades en común
Dropbox	Id	Id
	Name	Path
	Path_display	Kind
	Path_lower	FileName
	Size	Description
	Client_modified	ClientModifiedTime
Google Drive	Kind	ModifiedTime
	Id	Size
	Name	MimeType
	MimeType	ParentId
	ModifiedTime	
	Size	
	Parents	
OneDrive	Id	
	Name	
	ParentReference	
	LastModifiedDateTime	
	Size	
	File	
	Folder	
	Image	
	Audio	

Tabla 6 Mapping semántico - Leer archivos

5.4.3. Esquema canónico

Una vez presentado los modelos por separado de cada uno de los servicios que se han de integrar en la aplicación de gestión de nubes, y extraído las propiedades en común de Dropbox, Google Drive y OneDrive para llevar a cabo las funcionalidades básicas, es momento de presentar el esquema canónico con el que finalmente contará la aplicación de gestión de nubes.

Este esquema global trata de hacer independiente al modelo de sus fuentes y localizaciones, de los datos y su sintaxis, las posibles variaciones semánticas, etc.

Servicios	Fuente de datos	Esquema canónico
Dropbox	Id	Id
	Name	Path
	Path_display	Kind
	Path_lower	FileName
	Size	OriginalFileName
	Client_modified	Description
	Server_modified	ClientModifiedTime
	.Tag	ServerModifiedTime
	Rev	CreatedTime
Google Drive	Kind	ModifiedTime
	Id	Size
	Name	MimeType
	MimeType	ParentId
	Description	Thumbnail
	CreatedTime	Tag
	ModifiedTime	Rev
	OriginalFileName	Version
	Size	
	Parents	
OneDrive	Id	
	Name	
	ParentReference	
	CreatedDateTime	
	LastModifiedDateTime	
	eTag	
	Versions	
	Thumbnails	
	Size	
	Description	
	File	
	Folder	
	Image	
	Audio	

Tabla 7 Esquema canónico

En la tabla 28 se muestran las propiedades requeridas para las características mencionadas en las tablas anteriores donde se muestra el conjunto de datos necesarios para aplicar las respectivas funciones de gestión del cliente móvil.

En la ilustración 28 se muestra un esquema el envío de datos de las fuentes de los servicios de almacenamiento en la nube hacia el esquema global que ha generado.

Los datos provenientes de las fuentes llegan al cliente móvil a través de las respectivas librerías / SDK que permiten una comunicación más sencilla y permite extraer la información para que pueda ser manejada en un modelo de datos más reducido y común a todos los servicios incluidos.

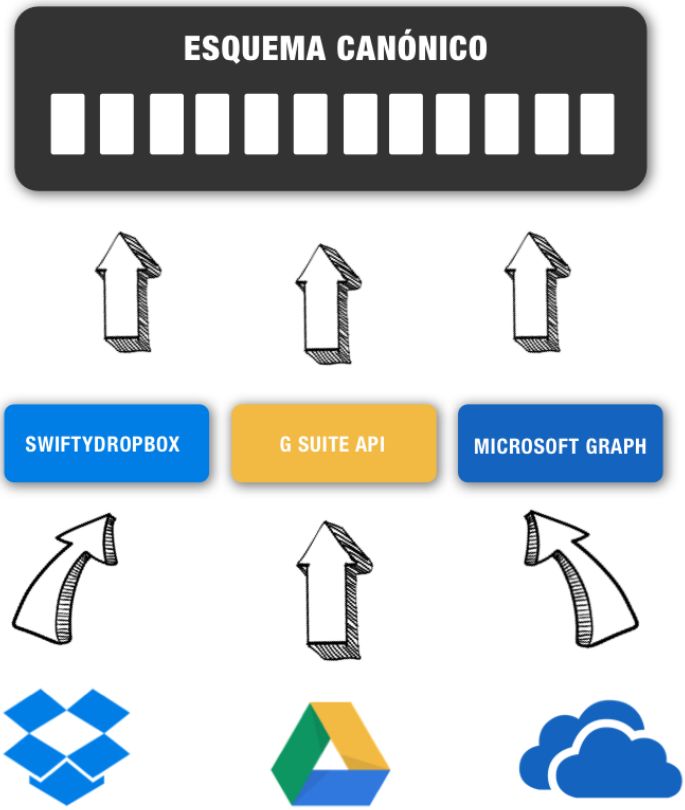

Ilustración 28 Integración virtual de datos de servicios de almacenamiento en la nube

6. Prototipo

En este documento se ha hecho un estudio de las aplicaciones con funcionalidades parecidas que ofrecen a los usuarios la posibilidad de incluir diferentes servicios de almacenamiento en la nube para ofrecer un acceso uniforme a los archivos alojados en servidores remotos, después, se ha realizado un estudio de los modelos de datos de las fuentes a integrar en una aplicación que gestione nubes, en el cual se busca una solución al problema de la heterogeneidad semántica el cual permite unificar múltiples fuentes de datos en un único modelo canónico que llega a ser transparente para el usuario ofreciendo la imagen de un único sistema.

El prototipo de este trabajo lleva a cabo la implementación y uso de dicho esquema canónico que unifica diversos servicios de almacenamiento de datos remotos. En este prototipo se dan a conocer los casos de uso de la aplicación de gestión de nubes, la tecnología empleada y solución de software a nivel arquitectónico que se propone para la implementación de tareas de integración de datos con diferentes servicios.

6.1. Casos de Uso

En esta sección se analiza los casos de uso que se producen en el prototipo de la aplicación de gestión de nubes.

6.1.1. Autorización de uso de nube

El actor **Usuario**, al iniciar la aplicación de gestión de nubes se encuentra con la autorización del uso de la cuenta y datos para poder manejar los archivos alojados en el sistema elegido como lo muestra la ilustración 29. Este caso de uso es obligatorio, ya que para acceder a la funcionalidad restante es necesario tener los permisos adecuados.

Este caso de uso queda separado del resto de casos de uso para no inferir en los demás ya que es un paso requerido para hacer cualquier acción sobre los archivos del espacio personal del usuario.

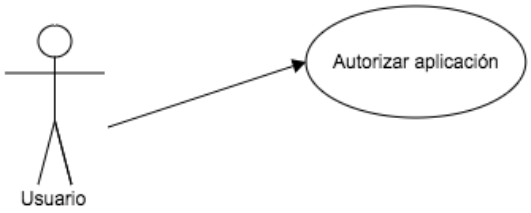

Ilustración 29 Caso de uso 1 – Autorización de uso de nube personal

6.1.2. Gestión de archivos

La gestión básica de los archivos de un sistema remoto es similar a la que se puede emplear en la gestión de archivos de un sistema local, por ello, este caso de uso ilustra las funcionalidades básicas de gestión de archivos que se contemplan en este prototipo de aplicación de gestión de nubes.

El único actor es el Usuario y él es el que puede realizar las acciones contra el sistema tal y como se muestra en la ilustración 30.

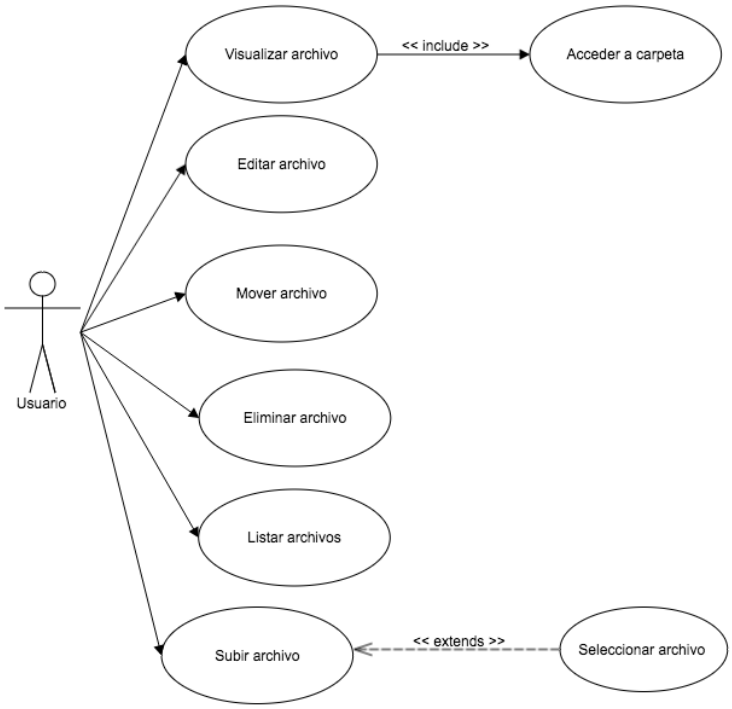

Ilustración 30 Caso de uso 2 - Gestión básica de archivos

6.1.3. Búsqueda de archivos

La búsqueda de archivos en el prototipo permite mostrar los elementos que el usuario ha introducido por teclado para poder ser listados con las coincidencias. Esta funcionalidad puede ser usada por el actor Usuario como lo indica la ilustración 31.

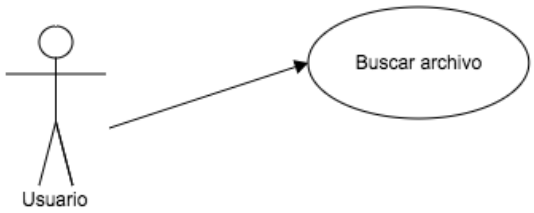

Ilustración 31 Caso de uso 3 - Buscar archivo

6.1.4. Información de usuario

El usuario cuenta con información básica del mismo dentro del prototipo de aplicación de gestión de nubes. Esta información se compone de ciertos datos que proveen los servicios tales como el nombre, apellidos, correo, etc. Otro tipo de información a mostrar es la capacidad de disco que se tiene disponible en la unidad de almacenamiento remoto.

La ilustración 32 muestra, además de la funcionalidad mencionada anteriormente, la funcionalidad de cerrar sesión que permitirá al usuario desautorizar a la aplicación del uso de los datos del servicio seleccionado.

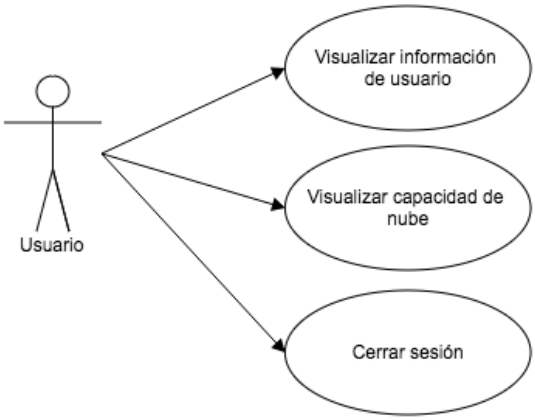

Ilustración 32 Caso de uso 4 - Información de usuario

6.2. Tecnología

La tecnología escogida para el desarrollo del prototipo es Swift, lenguaje de programación que permite realizar aplicaciones para móviles con sistema operativo iOS tales como el iPhone, iPad, iPod, Apple Watch y Apple TV. Swift es el lenguaje de programación sucesor de Objective-C, lenguaje con la que la gran mayoría de aplicaciones de la App Store están programadas a día de hoy, aunque cabe destacar que se pueden crear aplicaciones combinando ambos lenguajes gracias a la compatibilidad que permite el entorno de desarrollo Xcode.

Los **frameworks** más importantes del sistema operativo móvil de Apple son **Cocoa Touch** y **Foundation**, en ellos se encuentran más del 80% de las clases que se han usado para llevar a cabo el prototipo de la aplicación de gestión de nubes. iOS es un sistema operativo con una arquitectura basada en capas donde cada capa está compuesta por un framework, y éstos se componen de un conjunto de clases las cuales se utilizan para llevar a cabo el desarrollo de aplicaciones.

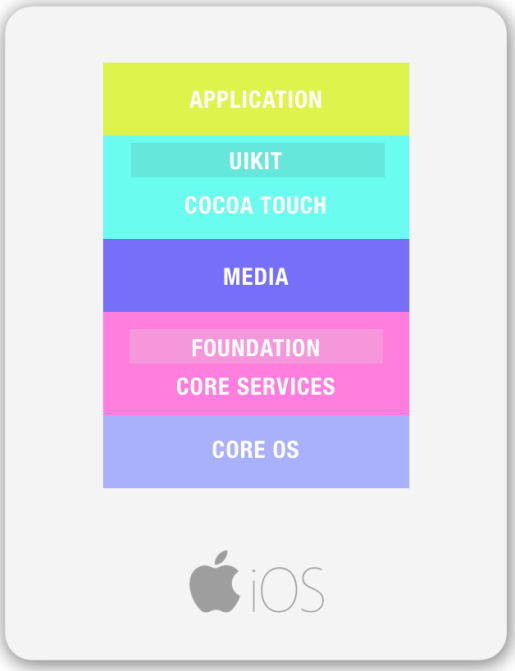

Ilustración 33 Arquitectura del sistema operativo iOS

Las capas más altas del sistema contienen los elementos necesarios para crear apps y a medida que se desciende entre las capas se encuentran frameworks más propios del kernel del sistema operativo, con frameworks como Core OS, la capa UNIX que contiene los ficheros del sistema, seguridad del sistema operativo, drivers del dispositivo, etc.

6.2.1. Entorno de desarrollo

Xcode es la herramienta que ofrece Apple para crear aplicaciones para dispositivos con iOS y aplicaciones de escritorio para sistemas con Mac OS. Presentado en octubre del 2003 junto a la versión 10.3 de Mac OS X, Xcode es el entorno de desarrollo integrado (IDE) que ofrece una interfaz amigable para los desarrolladores y que se puede descargar gratis desde la Mac App Store.

Este IDE incluye un conjunto de herramientas útiles para los desarrolladores como Interface Builder, herramienta que permite crear interfaces de usuario que se conocen como Storyboards; Compiladores del proyecto GNU (GCC) que permiten la compilación de código escrito en C, C++, Objective-C, Objective-C++, AppleScript y Java; Un potente editor de código fuente que permite la detección de errores en tiempo real; Múltiples SDK incluidos en el IDE para la creación de aplicaciones para el Apple Watch, Apple TV, Apple Car, entre otros; Herramientas de debug y testing, integración con control de versiones, simuladores de iOS, entre otras muchas herramientas que ofrece este gran entorno de desarrollo.

,ñ

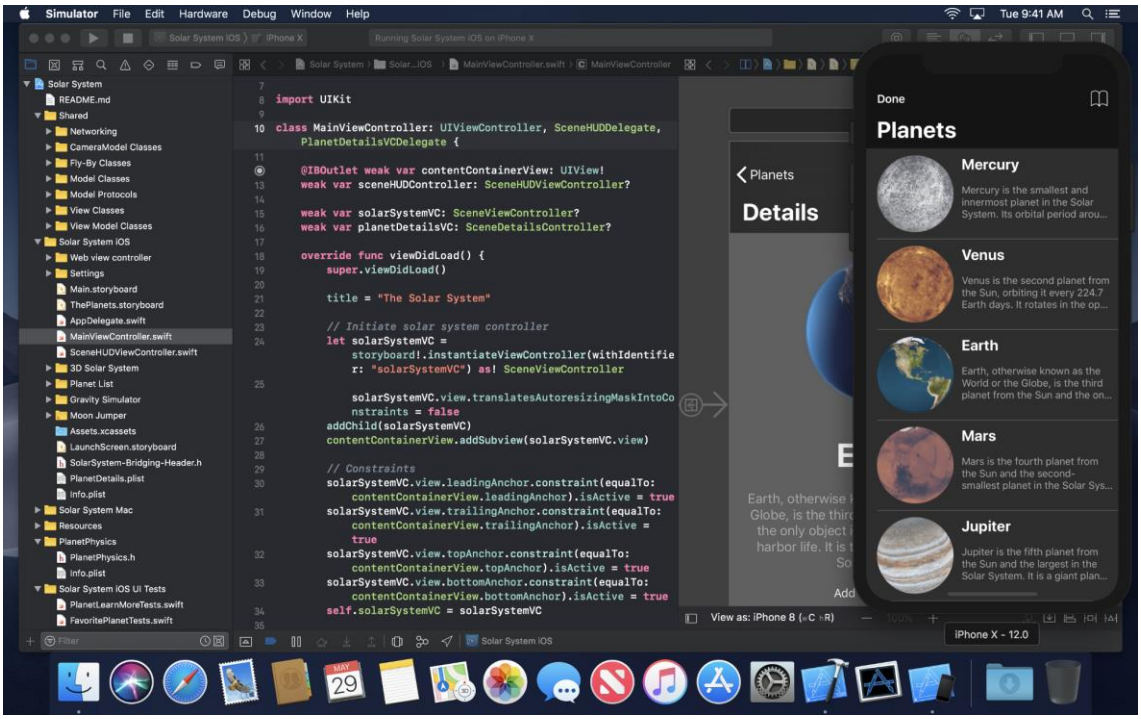

Ilustración 34 Entorno de desarrollo de aplicaciones Xcode 10

6.2.2. Control de versiones

Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedes recuperar versiones específicas más adelante según convenga. Este tipo de sistemas es muy usado en el desarrollo de software y permite revertir archivos o conjuntos del mismo a un estado anterior, revertir a un estado anterior, comparar cambios, ver quién modificó qué por última vez algo que puede ser causante de un problema, permite acotar más fácilmente un error, quién lo introdujo y cuándo, entre otras posibilidades que brinda un sistema control de versiones. Un VCS (Version Control System en

inglés) sirve en esencia para tener un histórico de cambios que se van registrando sobre un proyecto el cual permite acciones como las mencionadas anteriormente.

El sistema control de versiones utilizado para llevar a cabo el desarrollo de este prototipo de la aplicación de gestión de nubes es Git mediante la plataforma BitBucket.

Git es un sistema de control de versiones distribuido gratis y open source diseñado por Linus Torvalds. Git destaca como VCS gracias a características como la creación de ramas independientes de desarrollo, lo cual permite hacer desarrollos independientes que después se pueden mezclar, el desarrollo en paralelo, flujos de trabajos basado en funciones que luego se puede juntar a la línea principal de desarrollo, entre otras muchas bondades que ofrece este sistema.

BitBucket por otra parte es un servicio de alojamiento de proyectos de desarrollo basado en la web. Utiliza sistemas de control de versiones como Mercurial y Git y ofrece planes comerciales y gratuitos los cuales permiten tener inicialmente hasta 3 repositorios privados en su versión básica gratuita y un número ilimitado en su versión de pago. Esta plataforma es similar a GitHub que utiliza exclusivamente Git.

Ambas tecnologías juntas ofrecen una serie de servicios de gran calidad para los desarrolladores donde se puede generar peticiones de subida de código con funcionalidad nueva, revisiones de código por parte del equipo de desarrollo, comentarios ante una revisión de código, permisos entre ramas de desarrollo, despliegues en la plataforma, entre otras.

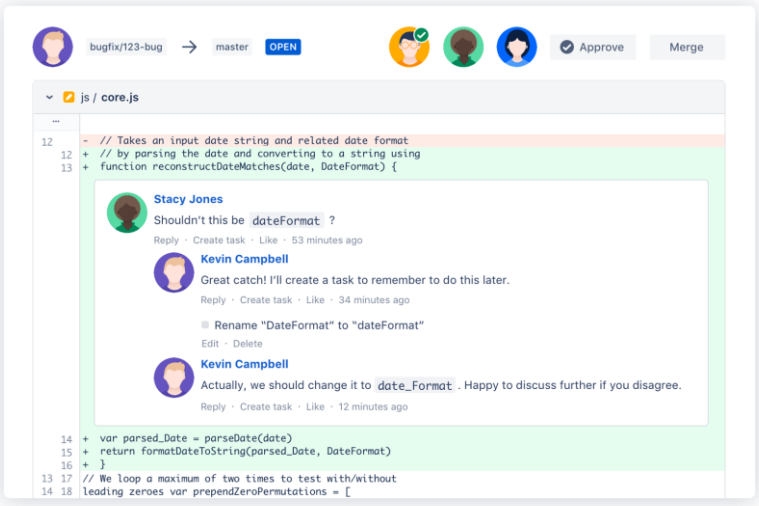

Ilustración 35 Ejemplo de revisión de código - BitBucket

6.2.3. Librerías

El uso de librerías o frameworks (módulos de software que en su conjunto ofrecen una serie de herramientas reutilizables para otros proyectos) en proyectos Xcode se realiza mediante un gestor de dependencias llamado CocoaPods. Este gestor de

dependencias ayuda a añadir dependencias de proyectos en el que se apoya este prototipo.

CocoaPods se apoya en un fichero llamado **Podfile** en el cual se especifican las dependencias que se han de descargar para el proyecto. Este fichero permite descargar los respectivos códigos fuentes y enlazarlos entre si dentro del espacio de trabajo de un proyecto Xcode que CocoaPods crea para el uso del proyecto más las librerías descargadas. Se muestra en la tabla 9 las librerías y frameworks que usa el prototipo.

Librerías y Frameworks	
SwiftyDropbox	SDK de conexión con las APIs de Dropbox.
G Suite API	SDK de conexión con las APIs de Google Drive.
Microsoft Graph	SDK de conexión con las APIs de OneDrive.
RxCocoa	Librería parte de la suite de ReactiveX (Rx). Framework que sirve de nexo para interactuar con las herramientas de Rx dentro de la API de Cocoa integradas en iOS y OS X.
RxSwift	Librería parte de la suite de ReactiveX (Rx). Framework que sirve de nexo para interactuar con las herramientas de Rx dentro del lenguaje Swift.
Realm Database	Librería alternativa a SQLite y Core Data. Sistema de gestión de base de datos mediante Objetos.

Tabla 8 Librerías y frameworks utilizadas en el prototipo

6.3. Diseño

En este apartado del documento se muestra la estructura del prototipo del cual se expone el diagrama de clases, arquitectura utilizada para implementar el software, los patrones de diseño utilizados, bocetos iniciales de la aplicación y finalmente el diseño funcional de los diferentes apartados de la aplicación de gestión de nubes.

6.3.1. Diagrama de clases

El diagrama de clases que muestra en la ilustración 36 describe lo que puede estar presente en el sistema que se está modelando, por ello se puede observar las diferentes clases que componen la lógica de negocio para el prototipo. Entre esas clases destacan las clases **CloudStorageElement**, **File** y **Directory**. Estas clases conforman el conjunto de clases que representa el esquema canónico ilustrado en la tabla 7, sin embargo, en la representación de las clases en el diagrama UML (Lenguaje de modelado de sistemas software) se puede observar que se definen los datos mediante una herencia de un elemento en común: CloudStorageElement.

La **herencia** permite que la clase File y Directory compartan propiedades como el path, id, nombre, entre otras. Otro elemento destacado del diagrama es la **composición** del CloudStorageElement y la clase Directory, ya que ésta última puede contener tanto ficheros como directorios.

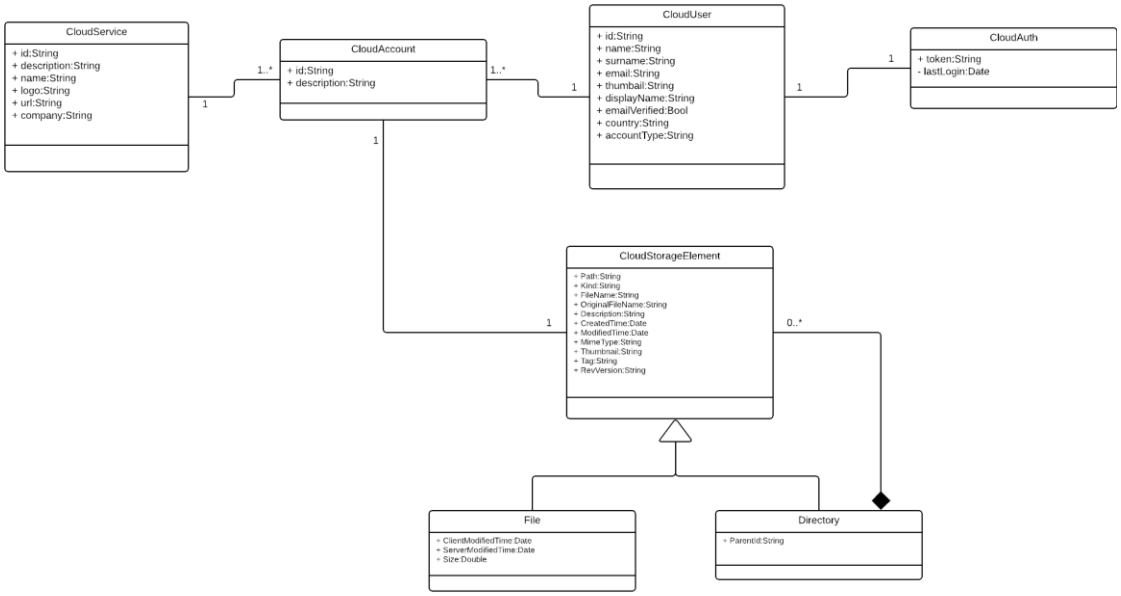

Ilustración 36 Diagrama de clases del prototipo

6.3.2. Arquitectura

La arquitectura de software hace referencia a la estructura del sistema que, idealmente, se define en etapas tempranas del desarrollo del mismo. En el caso del prototipo de la aplicación de gestión de nubes se utiliza un patrón de arquitectura MVC (Modelo Vista Controlador) el cual separa la lógica de negocio de la representación de los datos en la interfaz gráfica desde la cual interactúa el usuario.

6.3.2.1. Modelo Vista Controlador

El patrón de arquitectura Modelo Vista Controlador (ilustración 37) separa el dominio o lógica de negocio, la presentación de los datos y las acciones basadas en las entradas del usuario en tres partes.

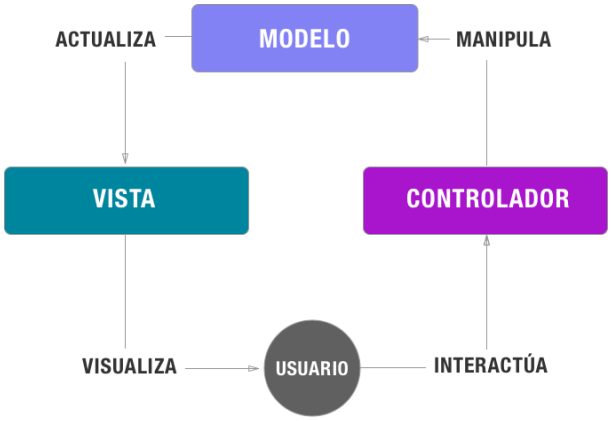

Ilustración 37 Modelo Vista Controlador

Modelo: El modelo maneja el comportamiento y los datos de la lógica de negocio de la aplicación. Contiene información acerca del dominio y su estado (normalmente

representados en la vista) y responde a los cambios de estado (normalmente desde el controlador).

Vista: La vista maneja el despliegue de información proveniente del modelo.

Controlador: El controlador interpreta las acciones de entrada por parte del usuario desde la vista e informa a la vista o al modelo para cambiar sus estados, o a ambas partes.

6.3.2.2. Patrones de diseño

En el prototipo de la aplicación de gestión de nubes también se han hecho uso de los patrones de diseño, los cuales permiten dar soluciones probadas para un problema recurrente que se produce en un cierto contexto y sistema.

En el caso del desarrollo del prototipo se ha hecho uso de patrones estructurales, como el patrón fachada el cual permite reducir la complejidad y minimizar dependencias, patrones creacionales tales como el patrón Singleton para proporcionar accesos globales a ciertos objetos, patrones de comportamiento como el patrón Observador, o el patrón Estrategia que favorece la estrategia de peticiones a realizar a las diferentes nubes.

6.3.2.2.1. Patrón Fachada

La motivación de la inclusión del patrón fachada es proporcionar un acceso unificado a los métodos compartidos entre las diferentes nubes incluidas en el prototipo, es decir, el prototipo cuenta con métodos como **listarArchivos**, el cual cuenta la misma definición para todos los servicios, este es uno de los casos donde el patrón fachada desacopla los accesos mediante la clase **DataManager**, la cual provee el acceso a los métodos de los DAOs (Objetos de acceso a datos) tanto de Dropbox, Google Drive y OneDrive los cuales cuentan con una definición de métodos propios.

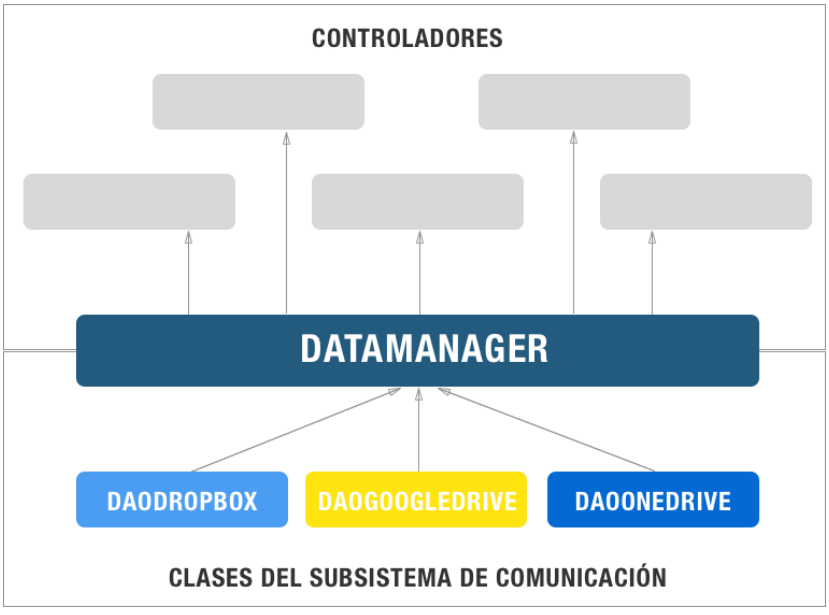

Ilustración 38 Patrón fachada

6.3.2.2.2. Patrón Estrategia

El patrón de Estrategia tiene un papel fundamental en la implementación y uso del patrón Fachada ya que este patrón ofrece encapsular las peticiones que se realizan en el prototipo según en qué nube esté interactuando el usuario, dicho de otra manera, este patrón permite dar soluciones diferentes a un mismo requisito.

Como se ha mencionado en el patrón Fachada, se cuenta con una funcionalidad de **listarArchivos** la cual se puede hacer con los diferentes servicios de nubes que tiene agregadas el usuario y dicha funcionalidad muestra una lista de archivos en pantalla, pues este patrón ofrece la posibilidad de poder resolver dicha petición del usuario de maneras diferentes ya que los servicios en si no comparten la forma de hacer peticiones al servidor a través de sus SDKs. En el caso de Dropbox hace las peticiones a través de SwiftyDropbox, en el de Google Drive a través de G Suite y en el caso de OneDrive a través del SDK que permite interactuar con Microsoft Graph.

En la ilustración 39 se presenta un diagrama de cómo está definido el patrón Estrategia con el ejemplo del método para **listaArchivos**.

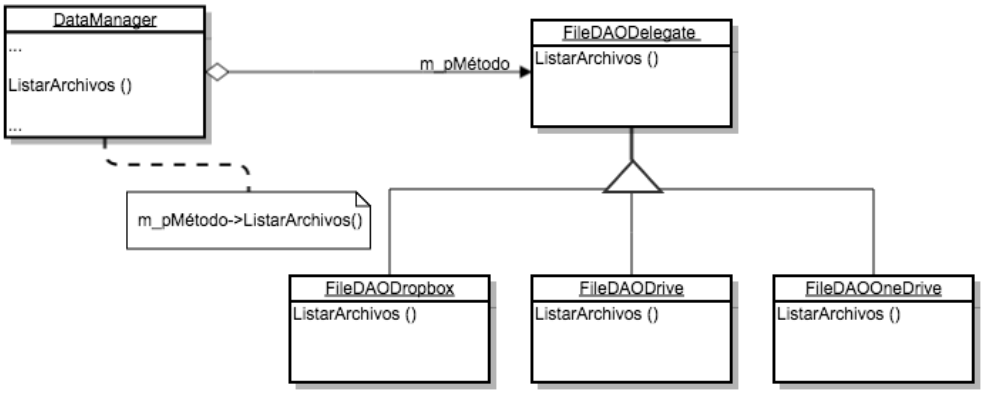

Ilustración 39 Patrón Estrategia

6.3.2.2.3. Patrón Singleton

La agregación del patrón Singleton al prototipo de aplicación de gestión de nubes intenta asegurar de que una clase tiene una sola instancia y así proporcionar un punto de acceso global a ella. La clase de la que se requiere tener una única instancia que siempre aporta lo mismo es la clase **DataManager**, la interfaz unificada para el conjunto de interfaces del subsistema de comunicación con las distintas APIs de los diversos servicios que se pueden integrar en la aplicación.

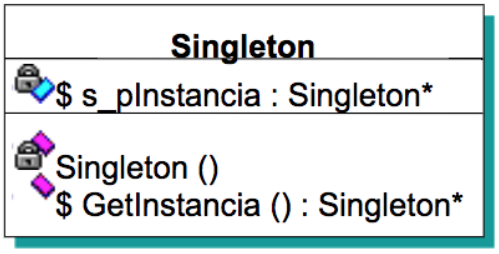

Ilustración 40 Patrón Singleton

6.3.2.2.4. Patrón Observable

El uso de este patrón de diseño viene dado por la inclusión de las librerías RxSwift y RxCocoa las cuales tienen como objetivo definir una dependencia 1:n con la vista para que cuando un objeto 1 cambie su estado, los n elementos asociados a su estado sean notificados y se actualicen automáticamente.

La principal motivación del uso de este patrón es la de actualizar los elementos visuales en el momento de realizar peticiones a los diferentes servicios de almacenamiento en la nube ya que al suscribir la vista a los diferentes eventos que actualizan la vista hace que sea cómodo representar la información en pantalla cuando la respuesta asíncrona llegue y se pueda visualizar.

La ilustración 41 muestra como actúa el patrón Observador en la implementación del prototipo donde se muestra como **observador** a la vista que actualiza una lista con archivos, la cual actúa de **sujeto** observado, de elementos que inicialmente está vacía y cambia de estado al recoger la respuesta asíncrona de los servicios remotos de las nubes integradas.

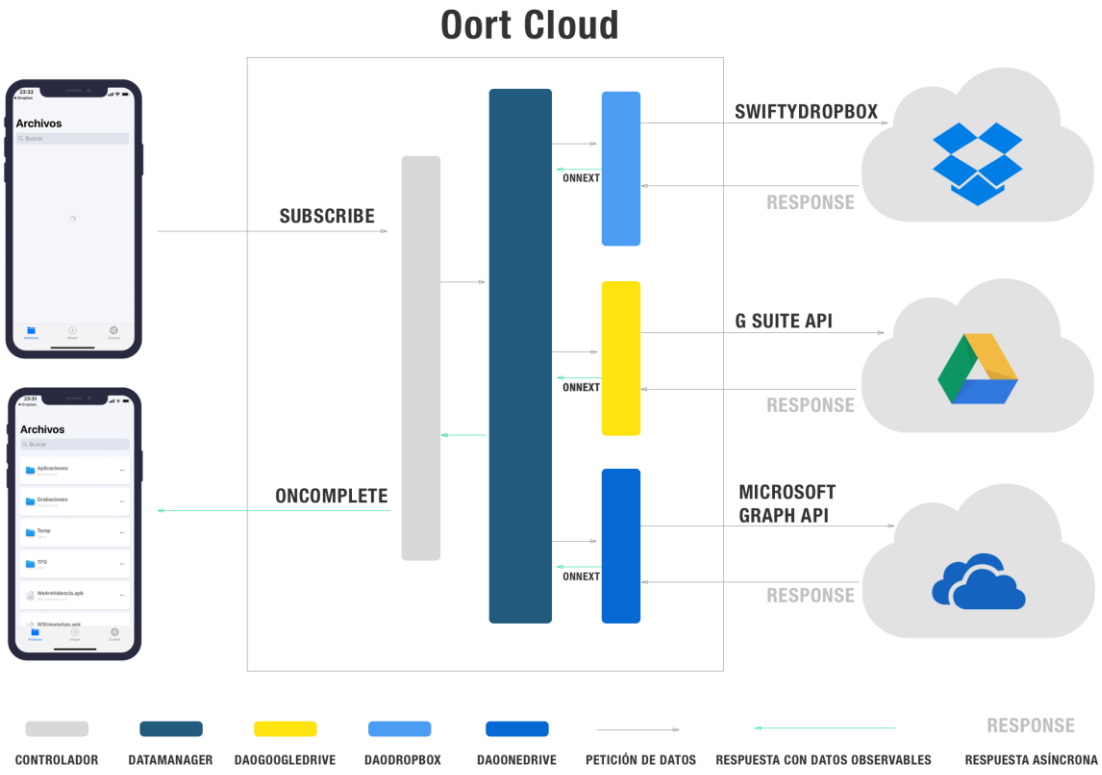

Ilustración 41 Patrón Observable del prototipo

6.3.3. Interfaz de usuario

La interfaz de usuario es el medio por el cual puede interactuar el usuario con el desarrollo que se explica en este documento sobre el prototipo. Este prototipo intenta juntar las tendencias en cuanto a diseño propuesto por Apple para sus aplicaciones en su web para desarrolladores en el cual prevalece un diseño limpio y una experiencia de usuario sencilla y útil.

6.3.3.1. Bocetos

En este apartado se adjuntan imágenes de los primeros bocetos hechos a lápiz en los que se puede observar algunos de los elementos que conforman la interfaz final de usuario, desde la ilustración 42 se puede observar el flujo del prototipo hasta las pantallas de listado de archivos, pasando por el perfil de usuario, entre otras.

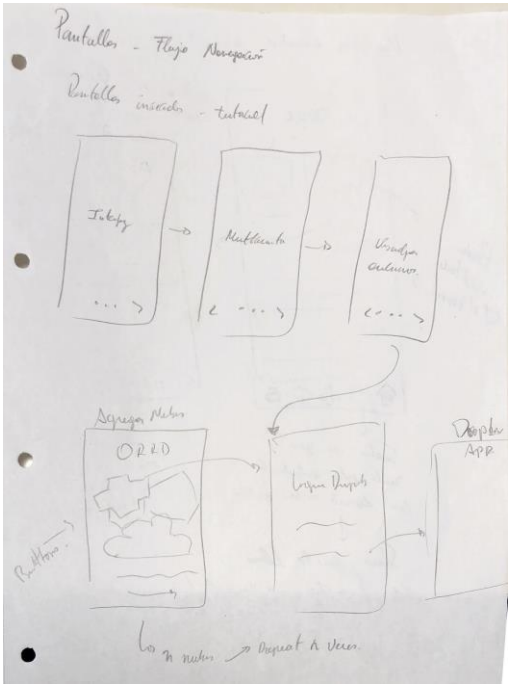

Ilustración 42 Bocetos - Flujo inicial

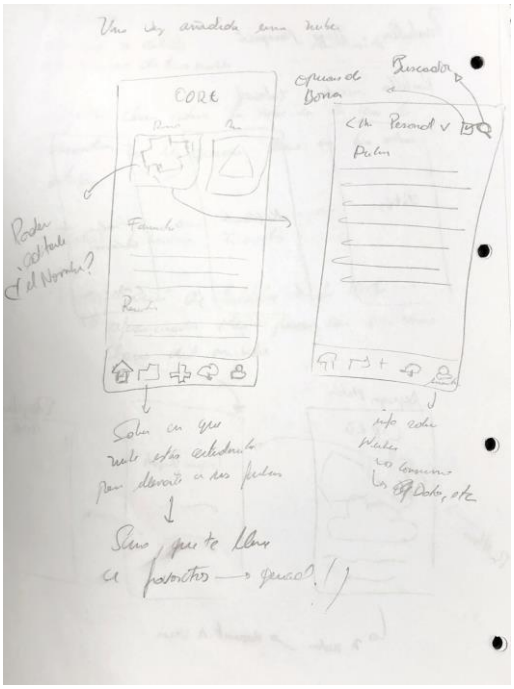

Ilustración 43 Bocetos - Pantallas iniciales

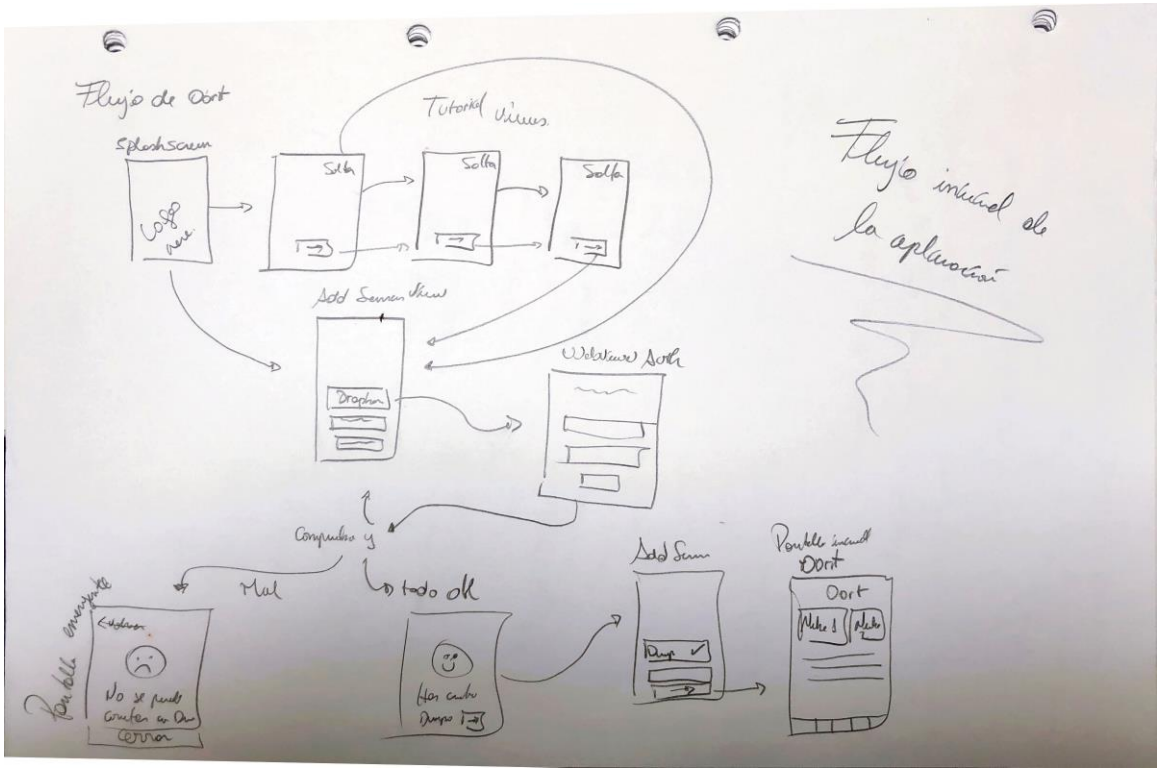

Ilustración 44 Bocetos - Borrador de posible flujo de inicio de la aplicación

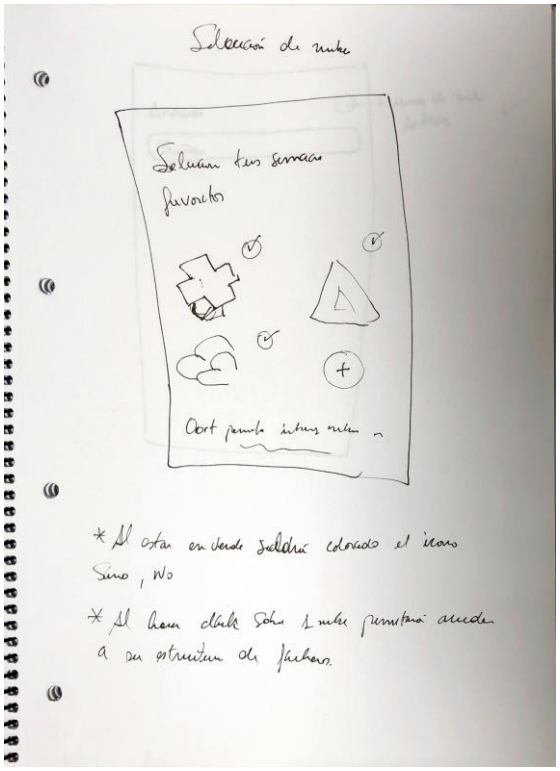

Ilustración 46 Bocetos - Selección de servicios de almacenamiento

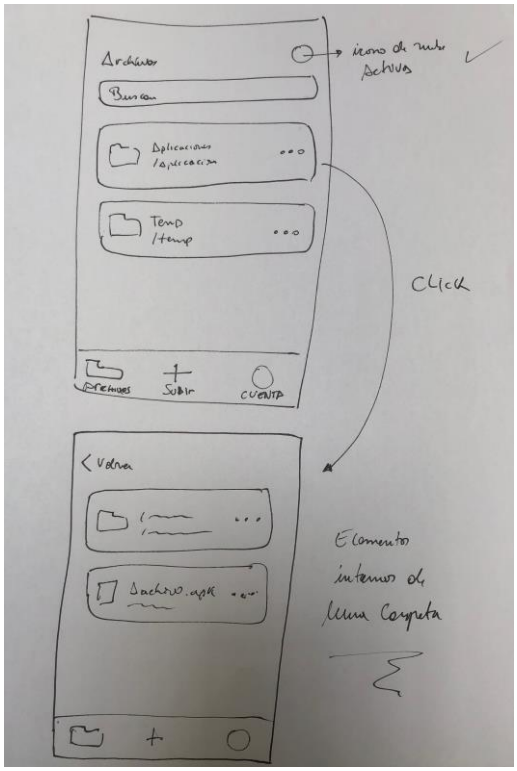

Ilustración 45 Bocetos - Selección de carpeta

Estas ilustraciones (46 a 48) muestran las pantallas principales del prototipo de aplicación de gestión de nubes, entre ellas se puede ver la selección de las nubes (ilustración 46) , la forma de visualizar carpetas (ilustración 45), perfil de usuario con la cantidad de espacio utilizado en sus nubes(ilustración 48), Añadir archivos desde múltiples opciones, crear carpeta(ilustración 47), etc.

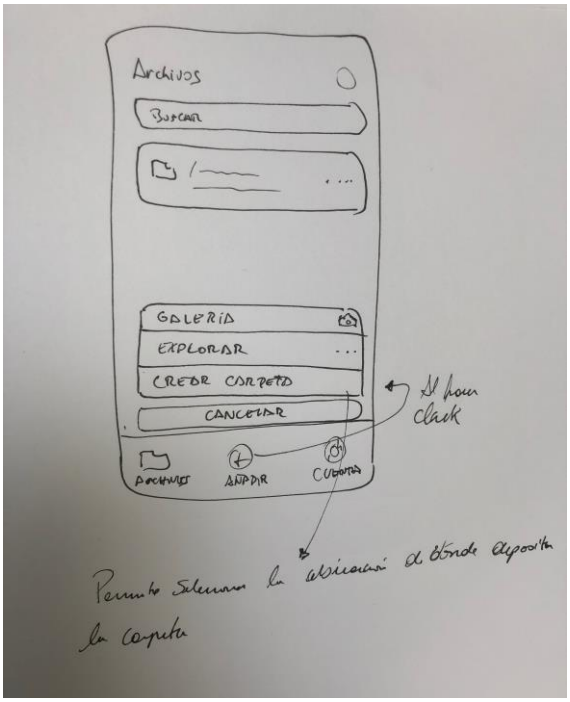

Ilustración 47 Bocetos - Añadir elementos

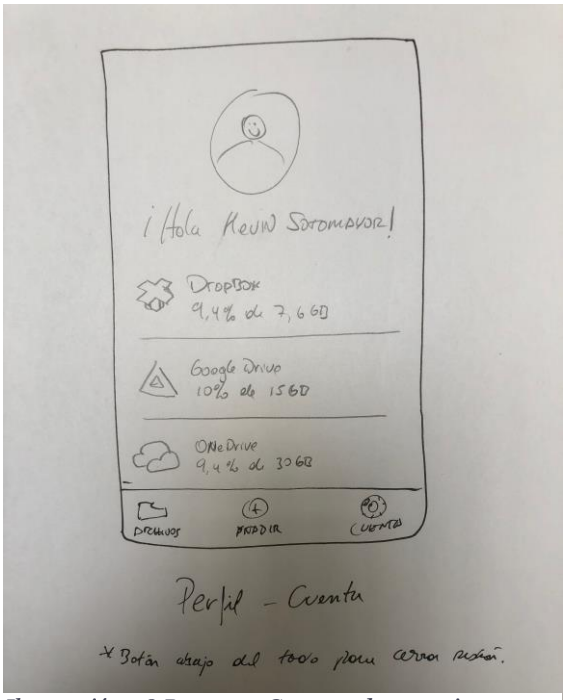

Ilustración 48 Bocetos - Cuentas de usuario

6.3.3.2. Interfaz final

La interfaz final es la solución del prototipo de aplicación de gestión de nubes en la cual se aprecia el contenido realizado en este documento para dispositivos móviles con iOS. Desde la ilustración 49 hasta la 54 se puede apreciar capturas de pantalla de la ejecución del prototipo sobre un dispositivo iPhone X el cual muestra la interacción del usuario con ciertos apartados que se describen a continuación

Introducción

Al inicio, la aplicación antes de mostrar la pantalla para la selección de nubes indica al usuario qué servicios ofrece (ilustración 49)

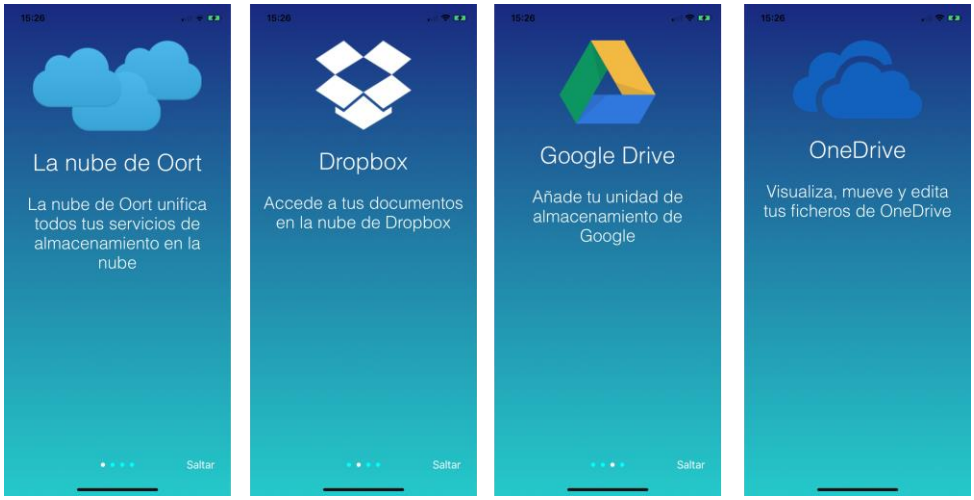

Ilustración 49 Prototipo - Pantalla de Introducción

Autorización

La ilustración 50 muestra la selección de las nubes que permite el prototipo y el inicio de sesión con una de ellas, en este caso Dropbox, la cual muestra su pantalla de autorización de uso de los datos.

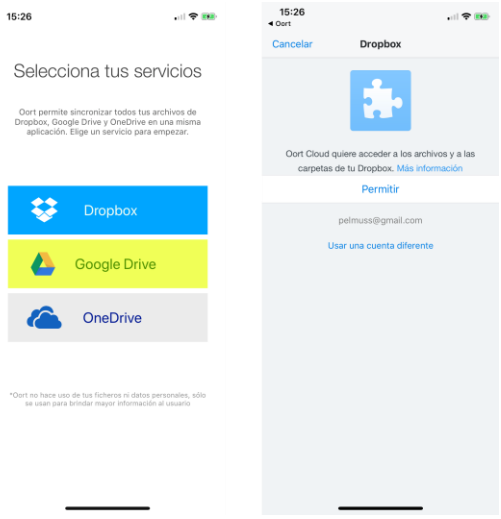

Ilustración 50 Prototipo - Pantalla de selección y autorización

Gestión de archivos

La ilustración 51 muestra el listado de archivos, el menú contextual para editar, mover o eliminar y el apartado de añadir ficheros, el cual permite agregarlos desde diferentes ámbitos, ya sea la galería, los servicios de iCloud en explorar, y crear carpeta.

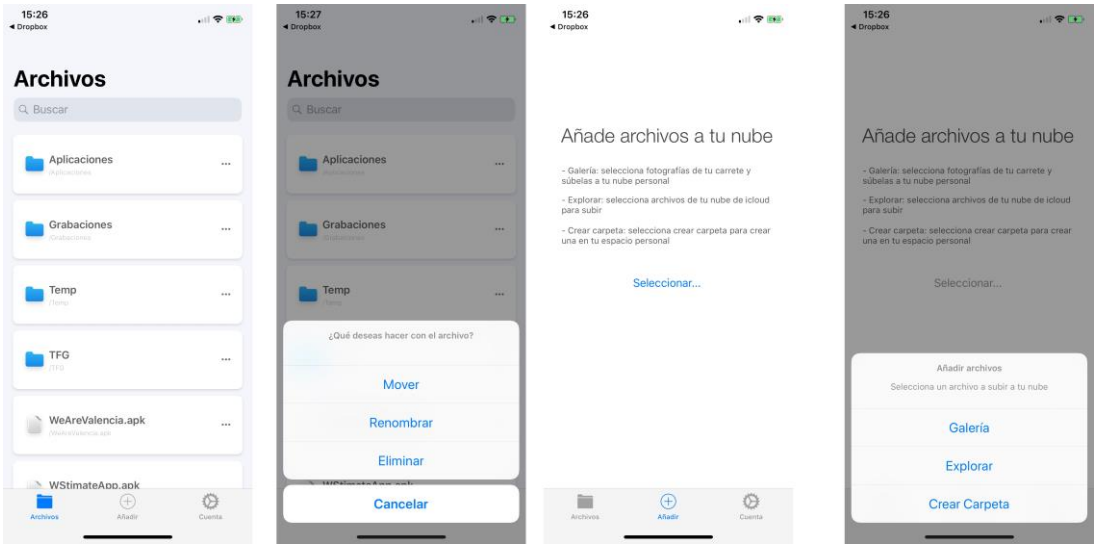

Ilustración 51 Prototipo - Gestión de archivos

Cuenta

La ilustración 52 muestra un resumen del uso de las nubes, en este caso cuando el usuario ha accedido a las diferentes nubes, se muestra el correo asociado a la misma y el espacio consumido en total de cada una de ellas. Cuando la nube aún no está integrada en Oort, éste no la muestra.

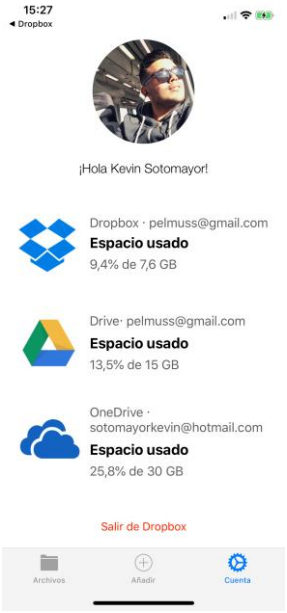

Ilustración 52 Prototipo - Cuenta de usuario

7. Conclusiones

En este apartado del documento se presenta una evaluación global de los resultados del estudio y análisis de la integración de los diferentes servicios de almacenamiento en la nube, una visión de futuro del prototipo acompañada de una valoración final por parte del autor.

En este documento se ha llevado a cabo el análisis de los diferentes servicios de almacenamiento en la nube los cuales permiten que muchos usuarios puedan ahorrar espacio de almacenamiento local y tenga accesible sus archivos en cualquier momento. Entre los servicios más conocidos se han elegido entre los más descargados a Google Drive, Dropbox y OneDrive, de los cuales se ha podido realizar un estudio de cómo está estructurado internamente cada elemento que se almacena en las unidades de almacenamiento. Este estudio ha permitido extraer propiedades que comparten en común estos servicios para poder realizar el desarrollo de un prototipo de aplicación de gestión de nubes. El desarrollo del prototipo implementa un conjunto de buenas prácticas y soluciones de software de gran calidad para poder integrar diferentes servicios en una aplicación móvil para el sistema operativo iOS.

En general la solución que se presenta para la integración de datos virtual aplicada en el prototipo descrito en este documento es una solución más que viable para poder llevar más lejos el desarrollo del prototipo expuesto, ya que, la aplicación que se puede ver en este documento cuenta con las funcionalidades básicas para gestionar archivos de los distintos servicios en la nube.

Como trabajos futuros de la aplicación de gestión de nubes se pueden añadir diferentes extras que puede hacer una aplicación muy destacable entre las aplicaciones de la App Store, añadiendo así funcionalidades como la de generar enlaces compartidos de manera sencilla, permitir compartir ficheros desde la propia aplicación, añadir múltiples cuentas para un mismo servicio, etc. Las bases del prototipo son bastante sólidas arquitectónicamente como para poder escalar a una aplicación más compleja y completa, ya que la inclusión de ciertos patrones ha hecho que el desarrollo sea más fácil y a la vez más eficiente y por tanto se puedan añadir funcionalidades como las mencionadas anteriormente.

Como conclusión este trabajo a nivel personal me ha ayudado a aprender de manera autodidacta a través de libros, videos, tutoriales, foros, etc. un nuevo lenguaje de programación móvil como lo es Swift. Integración de datos (IEI) fue una de las asignaturas, junto a otras, de las más interesantes de la carrera y los conocimientos aprendidos a lo largo de estos años me han permitido abordar la problemática con soluciones elegantes y robustas lo cual hace que los últimos 5 años en la carrera hayan merecido totalmente la pena.

8. Bibliografía

1. Interfaces - How to Model a simple file-system by UML class diagram [Internet]. Software Engineering Stack Exchange. [citado 9 de julio de 2018]. Disponible en: https://softwareengineering.stackexchange.com/questions/146286/how-to-model-a-simple-file-system-by-uml-class-diagram

2. Developers [Internet]. Dropbox. [citado 9 de julio de 2018]. Disponible en: https://www.dropbox.com/developers/apps/create

3. Pavel Bogart. Installing and Setting Realm Database - How to build Apps with Realm Database (Swift 4, Xcode 9) [Internet]. [citado 11 de julio de 2018]. Disponible en: https://www.youtube.com/watch?v=K_lWru2hmXA&frags=pl%2Cwn

4. Alexandra Feldman. API v1 is now deprecated [Internet]. Dropbox Developer Blog. [citado 11 de julio de 2018]. Disponible en: https://blogs.dropbox.com/developers/2016/06/api-v1-deprecated/

5. SwiftyDropbox Reference [Internet]. [citado 11 de julio de 2018]. Disponible en: http://dropbox.github.io/SwiftyDropbox/api-docs/latest/

6. The Swift Guy. How To Create Custom Tableview Cells In Xcode 8 (Swift 3.0) [Internet]. [citado 11 de julio de 2018]. Disponible en: https://www.youtube.com/watch?v=uBesaTUJZi0

7. The Swift Guy. How To Create A TableView In Xcode 8 (Swift 3.0) [Internet]. [citado 11 de julio de 2018]. Disponible en: https://www.youtube.com/watch?v=fFpMiSsynXM

8. Brian Advent. iOS Swift Tutorial: UINavigationBar Customization - Tips & Tricks [Internet]. [citado 11 de julio de 2018]. Disponible en: https://www.youtube.com/watch?v=gUhhFPTKCrE

9. UISearchController Tutorial: Getting Started [Internet]. Ray Wenderlich. [citado 11 de julio de 2018]. Disponible en: https://www.raywenderlich.com/157864/uisearchcontroller-tutorial-getting-started

10. System folder blue Icon | Plex Iconset | Cornmanthe3rd [Internet]. [citado 12 de julio de 2018]. Disponible en: http://www.iconarchive.com/show/plex-icons-by-cornmanthe3rd/System-folder-blue-icon.html

11. Network S. Royal blue folder 7 icon - Free royal blue folder icons [Internet]. [citado 12 de julio de 2018]. Disponible en: https://www.iconsdb.com/royal-blue-icons/folder-7-icon.html

12. Swift - UITableViewCell: rounded corners and shadow [Internet]. Stack Overflow. [citado 13 de julio de 2018]. Disponible en: https://stackoverflow.com/questions/37645408/uitableviewcell-rounded-corners-and-shadow

13. ios11 - Xcode 9 «iPhone is busy: Preparing debugger support for iPhone» [Internet]. Stack Overflow. [citado 14 de julio de 2018]. Disponible en: https://stackoverflow.com/questions/46316373/xcode-9-iphone-is-busy-preparing-debugger-support-for-iphone

14. ios - UITableView with sections and custom Objects in Swift [Internet]. Stack Overflow. [citado 23 de julio de 2018]. Disponible en: https://stackoverflow.com/questions/39920005/uitableview-with-sections-and-custom-objects-in-swift

15. Romero B. Two Basic Ways To Populate Your UITableView | Codementor [Internet]. [citado 23 de julio de 2018]. Disponible en: https://www.codementor.io/brettr/two-basic-ways-to-populate-your-uitableview-du107rsyx

16. Core Data: Introducción | Curso iOS 8 [Internet]. [citado 23 de julio de 2018]. Disponible en: https://www.pixybit.es/curso/desarrollo-apps-ios/capitulo-cd1.html

17. Souza V. awesome-ios: A curated list of awesome iOS ecosystem, including Objective-C and Swift Projects [Internet]. 2018 [citado 23 de julio de 2018]. Disponible en: https://github.com/vsouza/awesome-ios

18. HTTP - Developers [Internet]. Dropbox. [citado 23 de julio de 2018]. Disponible en: https://www.dropbox.com/developers/documentation/http/documentation

19. Teoría de la Programación Orientada a Protocolos en Swift 2 [Internet]. Apple Coding. 2015 [citado 24 de julio de 2018]. Disponible en: https://applecoding.com/analisis/programacion-orientada-protocolos-evolucion-swift-2

20. Guía para Swift 2: extensiones de protocolos [Internet]. Apple Coding. 2015 [citado 24 de julio de 2018]. Disponible en: https://applecoding.com/guias/swift-2-extensiones-protocolos

21. Tutorial Swift – ¿Qué es un Closure? [Internet]. KodigoSwift. 2017 [citado 24 de julio de 2018]. Disponible en: https://kodigoswift.com/tutorial-swift-closures/

22. What Is a Singleton and How To Create One In Swift [Internet]. [citado 25 de julio de 2018]. Disponible en: https://cocoacasts.com/what-is-a-singleton-and-how-to-create-one-in-swift

23. Networking with RxSwift | Netguru Blog on iOS [Internet]. [citado 26 de julio de 2018]. Disponible en: https://www.netguru.co/codestories/networking-with-rxswift

24. Azam M. MVVM in iOS [Internet]. Mohammad Azam. 2017 [citado 26 de julio de 2018]. Disponible en: https://medium.com/@azamsharp/mvvm-in-ios-from-net-perspective-580eb7f4f129

25. Belatrix Software. iOS: Optimizando código con RxSwift [Internet]. [citado 26 de julio de 2018]. Disponible en: https://www.youtube.com/watch?v=AMYeykKWe3w&frags=pl%2Cwn

26. RxSwift: Reactive Programming in Swift [Internet]. ReactiveX; 2018 [citado 26 de julio de 2018]. Disponible en: https://github.com/ReactiveX/RxSwift

27. RxSwift by Examples #3 - Networking. - Mobile & Web Development Company Poland - Droids On Roids [Internet]. iOS & Android Mobile App Development Company - Droids On Roids - Poland. 2016 [citado 26 de julio de 2018]. Disponible en: https://www.thedroidsonroids.com/blog/ios/rxswift-examples-3-networking/

28. The introduction to Reactive Programming you've been missing [Internet]. Gist. [citado 26 de julio de 2018]. Disponible en: https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

29. google-api-objectivec-client-for-rest: Google APIs Client Library for Objective-C for REST [Internet]. Google; 2018 [citado 28 de julio de 2018]. Disponible en: https://github.com/google/google-api-objectivec-client-for-rest

30. G Suite APIs for iOS | G Suite Developer [Internet]. Google Developers. [citado 28 de julio de 2018]. Disponible en: https://developers.google.com/gsuite/guides/ios

31. Ortiz FM. Managing async code in Swift [Internet]. iOS App Development. 2017 [citado 28 de julio de 2018]. Disponible en: https://medium.com/ios-os-x-development/managing-async-code-in-swift-d7be44cae89f

32. Lee B. The Complete Understanding of Swift Delegate and Data Source [Internet]. Bob the Developer. 2017 [citado 28 de julio de 2018]. Disponible en: https://blog.bobthedeveloper.io/the-complete-understanding-of-swift-delegate-and-data-source-9c91ecd7f1

33. Culver L. Try out SwiftyDropbox, the new Swift SDK for Dropbox API v2! [Internet]. Dropbox Developer Blog. [citado 28 de julio de 2018]. Disponible en: https://blogs.dropbox.com/developers/2015/05/try-out-swiftydropbox-the-new-swift-sdk-for-dropbox-api-v2/

34. Aleksandrov E. EAIntroView: Highly customizable drop-in solution for introduction views [Internet]. 2018 [citado 28 de julio de 2018]. Disponible en: https://github.com/ealeksandrov/EAIntroView

35. Builes J. file-structure: A simple file structure proposal suitable for any small or mid-sized iOS code base [Internet]. 2018 [citado 28 de julio de 2018]. Disponible en: https://github.com/jlnbuiles/file-structure

36. Ciurus M. Swift Pearls [Internet]. [citado 28 de julio de 2018]. Disponible en: http://michalciurus.github.io/RxSwift-for-dummies-1-Observables.html

37. How to Add Pull-to-Refresh to a Table View or Collection View [Internet]. [citado 6 de agosto de 2018]. Disponible en: https://cocoacasts.com/how-to-add-pull-to-refresh-to-a-table-view-or-collection-view

38. Document png image | Royalty free stock PNG images for your design [Internet]. [citado 6 de agosto de 2018]. Disponible en: http://pngimages.net/document-png-image-46

39. The Swift Guy. How To Display An Activity Indicator In xCode 8 (Swift 3.0) [Internet]. [citado 6 de agosto de 2018]. Disponible en: https://www.youtube.com/watch?v=dLfOdObZW7k

40. La AppStore cumple diez años y estos, son algunos datos curiosos [Internet]. [citado 21 de agosto de 2018]. Disponible en: http://www.milenio.com/tecnologia/10-datos-appstore-celebrar-decimo-aniversario

41. Rojas C. El estado del arte de las Apps. [Internet]. ion-book. [citado 21 de agosto de 2018]. Disponible en: https://blog.ng-classroom.com//blog/tips/estado-de-las-apps-hibridas/

42. Document Based Apps - Apple Developer [Internet]. [citado 21 de agosto de 2018]. Disponible en: https://developer.apple.com/document-based-apps/

43. Las claves de éxito de Dropbox, que hoy debuta en el Nasdaq [Internet]. La Vanguardia. 2018 [citado 28 de agosto de 2018]. Disponible en: https://www.lavanguardia.com/economia/20180323/441824037378/dropbox-bolsa-claves-exito.html

44. López M. ¿Cuál el mejor servicio de almacenamiento en la nube? Lo buscamos entre cinco candidatos [Internet]. Genbeta. 2015 [citado 28 de agosto de 2018]. Disponible en: https://www.genbeta.com/almacenamiento/cual-el-mejor-servicio-de-almacenamiento-en-la-nube-lo-buscamos-entre-cinco-candidatos

45. Home - OneDrive Dev Center [Internet]. [citado 30 de agosto de 2018]. Disponible en: https://developer.microsoft.com/en-us/onedrive

46. G Suite Developer | G Suite Developer [Internet]. Google Developers. [citado 30 de agosto de 2018]. Disponible en: https://developers.google.com/gsuite/

47. Limitación - Documentación - Microsoft Graph [Internet]. [citado 30 de agosto de 2018]. Disponible en: https://developer.microsoft.com/es-es/graph/docs/concepts/throttling

48. Qué es la autenticación basada en Token [Internet]. Carlos Azaustre. 2015 [citado 30 de agosto de 2018]. Disponible en: https://carlosazaustre.es/que-es-la-autenticacion-con-token/

49. Una introducción a OAuth 2 [Internet]. DigitalOcean. [citado 30 de agosto de 2018]. Disponible en: https://www.digitalocean.com/community/tutorials/una-introduccion-a-oauth-2-es

50. About Files and Folders | Drive REST API [Internet]. Google Developers. [citado 31 de agosto de 2018]. Disponible en: https://developers.google.com/drive/api/v3/about-files

51. PowerData G. Data Warehouse: todo lo que necesitas saber sobre almacenamiento de datos [Internet]. [citado 1 de septiembre de 2018]. Disponible en: https://www.powerdata.es/data-warehouse

52. Alvarado P. Qué Es Xcode, Para Qué Sirve y Cómo Descargar [Internet]. iPadizate. 2014 [citado 3 de septiembre de 2018]. Disponible en: https://www.ipadizate.es/2014/07/20/xcode-93212/

53. What's new in Xcode 10? – developerinsider – Medium [Internet]. [citado 3 de septiembre de 2018]. Disponible en: https://medium.com/developerinsider/whats-new-in-xcode-10-fddeab035d05

54. Git - Acerca del control de versiones [Internet]. [citado 3 de septiembre de 2018]. Disponible en: https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones

55. About - Git [Internet]. [citado 3 de septiembre de 2018]. Disponible en: https://git-scm.com/about

56. Atlassian. Bitbucket: Funcionalidades [Internet]. Atlassian. [citado 3 de septiembre de 2018]. Disponible en: https://es-4f4071804890f12e0.getsmartling.com/product/features

57. ¿Qué es la programación reactiva? | Consultoría y Servicios IT para empresas [Internet]. Profile Software Services. 2017 [citado 4 de septiembre de 2018]. Disponible en: https://profile.es/blog/que-es-la-programacion-reactiva-una-introduccion/

58. Code T. Realm Database [Internet]. Tony Code. 2017 [citado 4 de septiembre de 2018]. Disponible en: https://medium.com/@howtocod3/realm-database-83655bb0a92a

59. Barquinero JMM. Agregación Vs Composición en diagramas de clases. UML. [Internet]. Blog SEAS. [citado 5 de septiembre de 2018]. Disponible en: https://www.seas.es/blog/informatica/agregacion-vs-composicion-en-diagramas-de-clases-uml/

60. Sergio Becerril SB. Swift 4. Aprender a crear Apps para iPhone y iPad. RC Libros 2018;