Universitat Oberta de Catalunya

Faculty of Computer Science, Multimedia and Telecommunications

Design and implementation of an end-to-end encrypted cloud backup service for disk partitions

Bachelor thesis written and presented by
Xavier Velàsquez Melenciano

Supervised by
Oriol Martí Girona & Sergi Caballé Llobet

Grau en Enginyeria Informàtica

Delivered on June 2020

License

Design and implementation of an end-to-end encrypted cloud backup service for disk partitions

Design of a potential commercial solution and implementation of a prototype of the service

Xavier Velàsquez Melenciano

<xvelasqu@uoc.edu>

Abstract

The market trend in the latest years regarding data backup solutions for organizations has been the massive adoption of cloud drive services like Dropbox or Google Drive. These systems have proved to be very powerful productivity tools for day to day operations by enabling data ubiquity, real-time collaboration, and version control for any kind of file.
Nevertheless, most of these services have raised many concerns regarding safety and data protection. Recurrent data breaches, insufficient protection against legal subpoenas and summons, and the impossibility of enabling verifiable encryption out of the box have brought the spotlight to innovative cloud services offering end-to-end encryption where all cryptographic operations are claimed to take place in the user's device — like MEGA and ProtonDrive.

The goal of this project is to bring this approach to a cloud storage service where instead of hosting single files, it will store an encrypted image of a whole disk partition. Users will install an application in their devices which will allow them to schedule backups, execute the encryption/decryption processes, and synchronize these disk snapshots with a cloud service. This cloud service will also offer a web interface with options for users to retrieve their backups. The need for a service like this is justified by the increase of data corruption threats that malware, especially these known as ransomware, has posed to corporations with a big number of interconnected workstations. File cloud drives generally fail both in terms of reliability and efficiency for data recovery in cases of a total loss, while partition-level backups excel in these use cases.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context and Project justification

The goal of this project is to design a multi-platform application for performing client-side encrypted full and differential backups of all desired disk partitions in a user workstation and store them remotely in the cloud. It is meant to cover the data backup needs of small and medium companies and individuals, who generally find existing solutions confusing and many don't even realize that they have this need.

Certainly, the spread of online cloud services replacing desktop productivity tools like Google Docs, Gmail, and Dropbox has substantially reduced the damage derived from data loss events in the user segments previously mentioned; but at the same time has risen new concerns regarding privacy and additional potential forms of data leaks [24]. This project aims to develop a cloud backup service that is easy to understand and use, that does not interfere with the way that users use productivity software of their choice, and where data confidentiality is uncontested.

Key characteristics of this service will be ease of use and painless setup, lowest consumption of compute and network resources as possible, and full privacy due to end-to-end encryption of backups with one-time keys.

1.2 Project objectives

In the first place, the main goal is to design the service previously introduced and provide a full software specification that covers all the components involved. The main functionalities of this service shall be:

- Scheduling of unattended full-disk backups with differential snapshots, similar to Apple's Time Machine or Microsoft Windows System Restore.
- Backup encryption taking place in the user's device with one-time randomly generated encryption keys per each file.
- Background upload of backups to the cloud via a secure protocol.
- Multi-platform availability of the client applications, initially for user workstations using Windows, macOS, and Linux-based operating systems [*Figs.* 1.1 & 1.2].
- Recovery options of the disk images via direct download or physical remittance in removable media, to be decrypted on the final user's device using the encrypted keys stored in the cloud service and the user's master password.

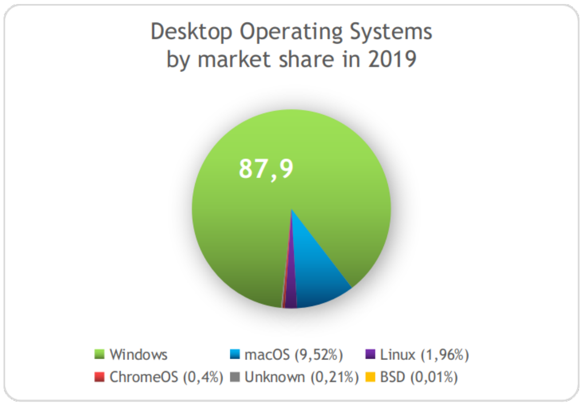


Figure 1.1: Source: NetMarketShare [10]

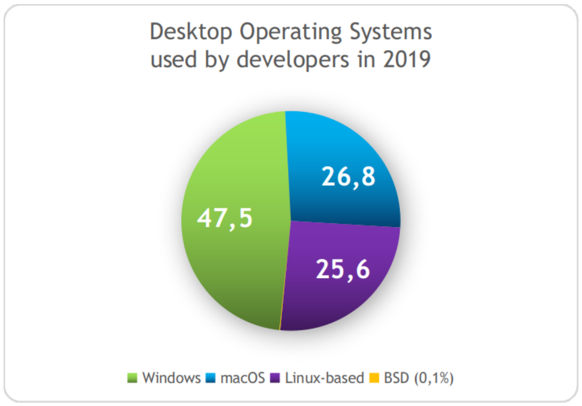


Figure 1.2: Source: Stack Overflow [11]

Then, the second objective is to develop, implement, and install a prototype of this service which shall cover, at least, one major use case of the future whole implementation.

1.3 Scope and Work methodology

Although the design of the described service involves high complexity disciplines like cryptography, file systems, and computer memory architecture; the implementation of *crypto* algorithms and the internals of disk partitions and their file systems are out of the scope of this project.

Instead, the focus will be set on the software engineering aspect of every component in the platform, and the potential of a service like the described one to become a viable commercial product. Because of that, the reader will notice that many design choices and decisions regarding the technology stack will be based on the efficiency, ease of use, maintenance complexity, and cost-effectiveness of the solution.

Project planning and execution followed the Waterfall model [8], up to the implementation phase, at all times. Although given the distributed aspect of this service, anyone willing to continue working on the achieved prototype towards an MVP should consider following an iterative approach; for example, one that follows Agile principles [7].

1.4 Work breakdown and Schedule

The initial planning contained the project goals with forecasted effort and completion deadline for every aspect of the design and the prototype implementation and was formalized as a Gantt chart [Annex A]. It relied on the expected partial deliverables expected by this thesis' supervisors and the objectives for each one of them, and mere effort estimation for every project aspect during the initial planning phase.

Although it lacks effort estimations for formal testing of the components implemented and the fact that it was affected by a posterior reduction of features of the prototype system, which will be elaborated further in this document; it could serve as a template for any party willing to bring this service to reality.

1.5 Summary of achieved outcomes

- Use cases and Requirements definition
- Class hierarchy and Entity relations for every application
- Selection of a commercially viable technology stack
- Testing, CI/CD, and QA strategy proposal
- Implementation of a prototype of the service with in-house hardware, fully covering a major use case

1.6 Briefing of the upcoming chapters

In the following sections, the reader will find an extended requirement specification of the software solution; structured following IEEE's Software Requirements Specification [2] [6].

In addition to that, the characteristics and implementation process of the prototype system will be described.

Chapter 2

Overall Description

2.1 Product Perspective

This chapter aims to describe the architecture, functionality, and other relevant characteristics of the distributed system, both software and hardware components, required to bring to life the desired service.

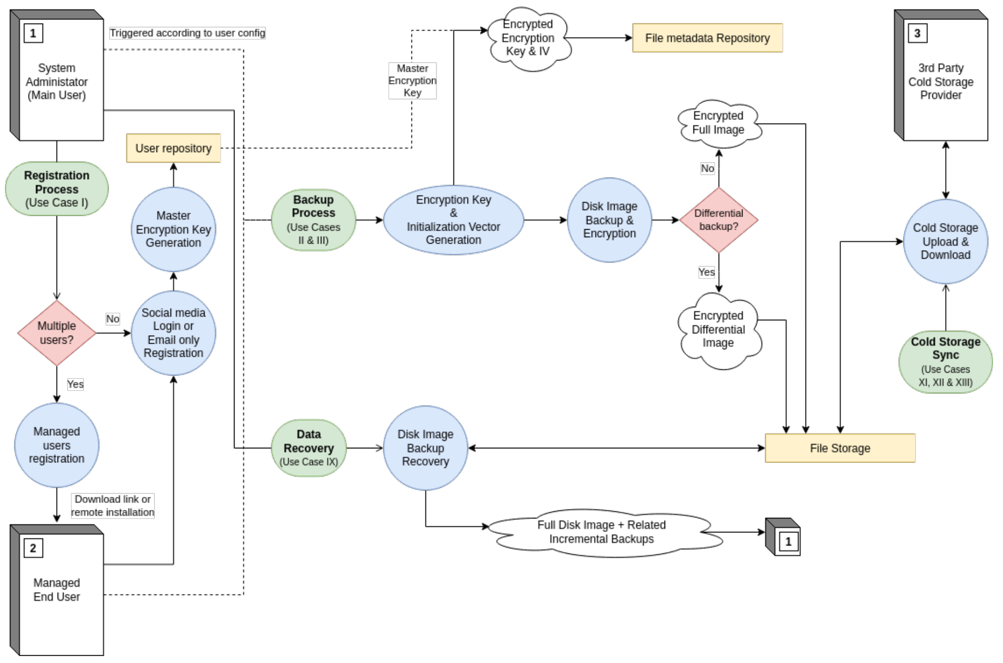

1
System Administator (Main User)
Triggered according to user config
Registration Process (Use Case I)
Multiple users?
No
Yes
Managed users registration
Download link or remote installation
2
Managed End User
Social media Login or Email only Registration
Master Encryption Key Generation
User repository
Master Encryption Key
Encrypted Encryption Key & IV
File metadata Repository
Backup Process (Use Cases II & III)
Encryption Key & Initialization Vector Generation
Disk Image Backup & Encryption
Differential backup?
No
Encrypted Full Image
Yes
Encrypted Differential Image
Data Recovery (Use Case IX)
Disk Image Backup Recovery
File Storage
Full Disk Image + Related Incremental Backups
1
3
3rd Party Cold Storage Provider
Cold Storage Upload & Download
Cold Storage Sync (Use Cases XI, XII & XIII)

Figure 2.1: Data Flow Diagram contextualizing use cases I, II, III, IX, XI, XII, & XIII [2.2]

It will also describe the 3rd party technologies and partners required for its operation, which will be elaborated on the external interfaces chapter [3], and a simplified version of a potential business plan and an outlook for its possible commercialization.

2.2 Product Functionality: Main Use Cases

Categorized according to the three main actors in the service, a high-level description of the main use cases follows:

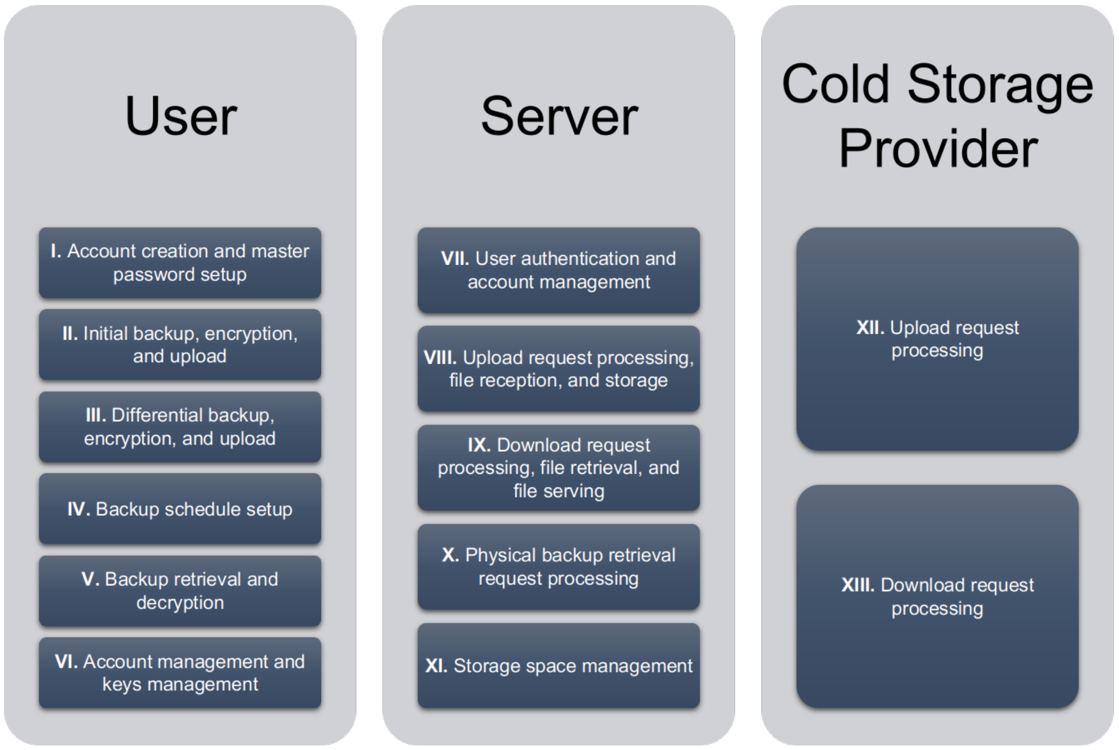

User
I. Account creation and master password setup
II. Initial backup, encryption, and upload
III. Differential backup, encryption, and upload
IV. Backup schedule setup
V. Backup retrieval and decryption
VI. Account management and keys management
Server
VII. User authentication and account management
VIII. Upload request processing, file reception, and storage
IX. Download request processing, file retrieval, and file serving
X. Physical backup retrieval request processing
XI. Storage space management
Cold Storage Provider
XII. Upload request processing
XIII. Download request processing

Figure 2.2: Main Use Cases of the service [2.2]

2.2.1 User-Initiated Use Cases

I. **Account creation and master password setup:** A new user wants to create an account in the service and a master password to protect his/her master encryption key.

II. **Initial backup, encryption, and upload:** A user wants to generate a full backup of a disk partition of his/her system, encrypt it with a randomly generated one-time key, encrypt this key with his/her master encryption key, and upload both the encrypted payload and its key to the server.

III. **Differential backup, encryption, and upload:** A user wants to generate a differential backup based on a previous full backup of a disk partition of his/her system, encrypt it with a randomly generated one-time key, encrypt this key with his/her master encryption key, and upload both the encrypted payload and its key to the server.

IV. **Backup schedule setup:** A user wants to choose which disk partitions will be automatically backed up, encrypted, and uploaded to the server; and the frequency of these scheduled operations.

V. **Backup retrieval and decryption:** A user wants to download or order physical delivery of a previously uploaded backup and, when received, decrypt it.

VI. **Account management and keys management:** A user wants to modify his/her private data (including master password) and/or subscription, add or remove managed users to the account, or just delete the account.

2.2.2 Server-initiated Use Cases

VII. **User authentication and account management:** The server wants to fulfill an authorization request or account CRUD request.

VIII. **Upload request processing, file reception, and storage:** The server wants to register a user file upload request, including its encryption keys, and receive and store the file.

IX. **Download request processing, file retrieval, and file serving:** The server wants to retrieve a file requested by a user from local or external cold storage and then digitally transfer it to the user.

X. **Physical backup retrieval request processing:** The server wants to record a physical file delivery request, which will be manually processed by the service staff, to fulfill a user-initiated request.

XI. **Storage space management:** The server wants to transfer backup files that are not accessed frequently to the cold storage, to free space for new user files.

2.2.3 3rd party provider-initiated Use Cases

XII. **Upload request processing:** The 3rd party provider wants to receive and store a file sent by the server.

XIII. **Download request processing:** The 3rd party provider wants to digitally deliver a file requested by the server.

2.3 User Classes and Characteristics

2.3.1 End Users

- **System Administrator:**

  They are either individuals who sign up in the service or IT administrators in an organization. They can link new devices to their account and invite managed users whose storage devices will be backed up. They're the only end-users with permission to retrieve backups stored in the cloud and with access to the account's encryption private keys.

- **Managed User:**

  These users' accounts are managed by a system administrator. They are allowed to install the client software and schedule backups for their devices, but they need to contact their sysadmin to perform recovery tasks.

2.3.2 Service Operations

- **Cloud Service Administrator:**

  It is the engineer in charge of supervising the correct function of the backups cloud storage service. He/she will make sure that the service complies with the SLA acquired with the end-users and that a secure and cost-effective Cold Storage backup strategy is being executed together with the more appropriate 3rd party providers.

- **Operations Technician:**

  These technicians support customer service staff and end-users in data recovery processes, giving advice on the most appropriate backup retrieval way according to the budget and time restrictions (e.g. direct download, P2P technologies, hard disks shipment, tape shipment, ...) and assisting in the disk image decryption and system recovery processes.

They can also advise customers on backup strategies in terms of frequency and device coverage scope. They can be considered Tier 3 support and most of them should have the ability to troubleshoot and fix issues in the systems and/or software.

- **Customer Service Representative:**

  Customer Service Representatives comprehend the first two tiers of support to end-users:

  Tier 1 greet customers in the available support platforms (e.g. ticketing system, live chat, call center, support forums, ...) and perform the first triage of incidents.

  Tier 2 agents assist customers and have a technical background sufficient to properly identify bugs and report them to the development team, to identify a total or partial system outage and report it to the Cloud Service Administrator, and to assist customers setting up their backup schedule strategy or taking actions in front of a data loss situation; always assisted by Operations Technicians.

2.3.3 3rd Party Provider

- **Cold Storage Service Representative:**

  This party can be either an Account Manager with a technical background or an IT expert. Should act as a SPOC and be able to provide immediate assistance in case of outage of the Cold Storage service and assist with heavy or complex data retrieval operations. Any situation not complying with the SLA established with the provider must be reported to this representative immediately.

2.4 Software Components Formalization

2.4.1 Service Components

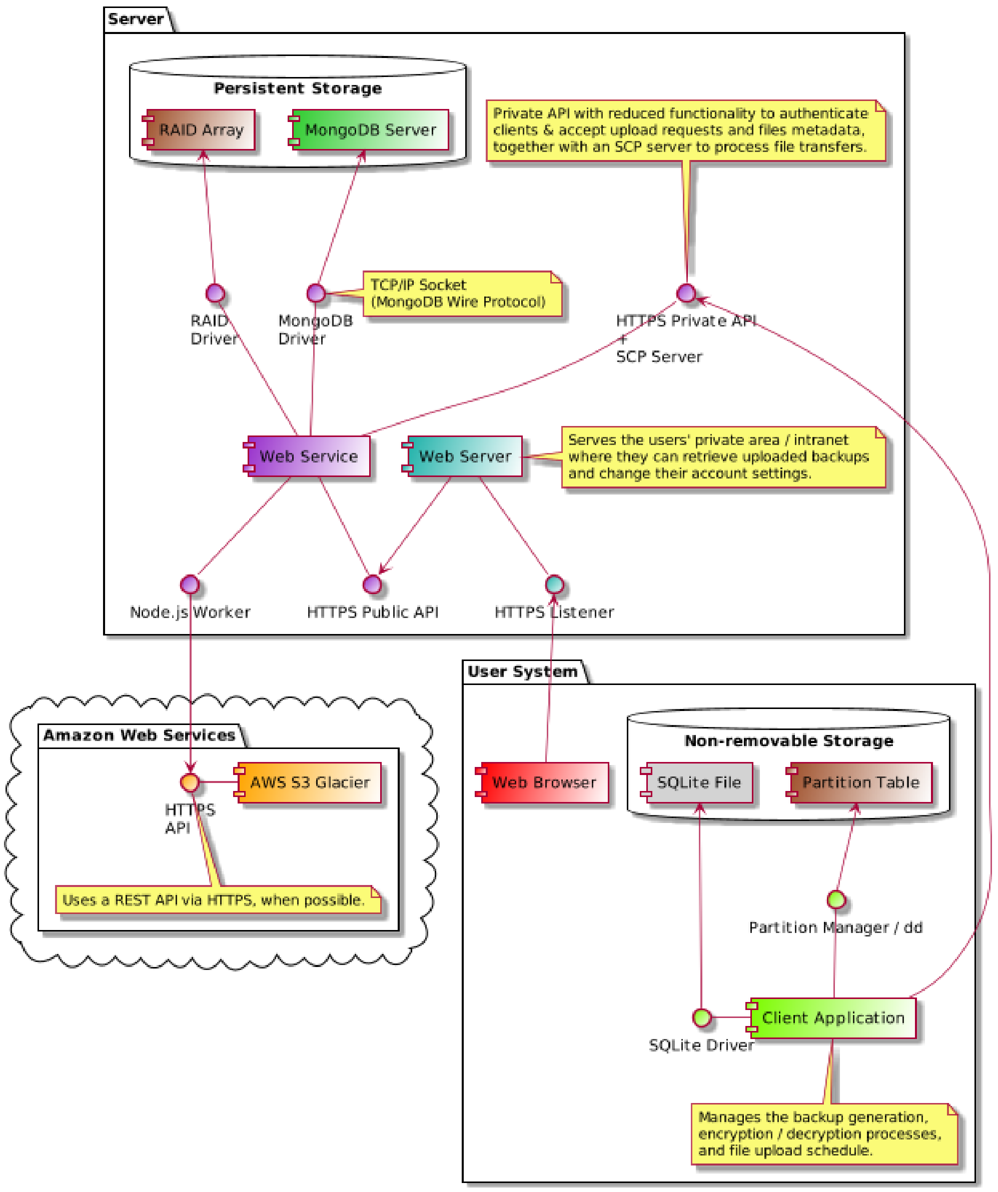

Server
Persistent Storage
RAID Array
MongoDB Server
Private API with reduced functionality to authenticate clients & accept upload requests and files metadata, together with an SCP server to process file transfers.
TCP/IP Socket (MongoDB Wire Protocol)
RAID Driver
MongoDB Driver
HTTPS Private API + SCP Server
Web Service
Web Server
Serves the users' private area / intranet where they can retrieve uploaded backups and change their account settings.
Node.js Worker
HTTPS Public API
HTTPS Listener
Amazon Web Services
AWS S3 Glacier
HTTPS API
Uses a REST API via HTTPS, when possible.
User System
Non-removable Storage
Web Browser
SQLite File
Partition Table
Partition Manager / dd
Client Application
SQLite Driver
Manages the backup generation, encryption / decryption processes, and file upload schedule.

Figure 2.3: Components Diagram of the service

2.4.2 Class Hierarchy Proposals

The class hierarchy proposals presented above consist in a Model-View-Controller [1] structure for both the client and server applications. This paradigm allows for an effective decoupling between the presentation layer and the business logic and database driver, which is exactly what we are aiming for by using Qt; sharing a core business logic that is isolated from other components while compatible with every target operating system.

Client Application

The following diagram is an abstraction of what the final class hierarchy would look like, as Qt specific types and controllers are not specified. *Getter* and *setter* methods are not displayed either. All of this is done on purpose, so the reader can get a better idea of the actual data entity relations.
Also, decoupling of the view works in a particular fashion in the Qt framework. The developer will normally design views by defining an XML layout and have the view widgets communicate with the application controllers via events, known as signals [43]. Because of this, view controllers are not specified in the diagram.

Diagram at the next page

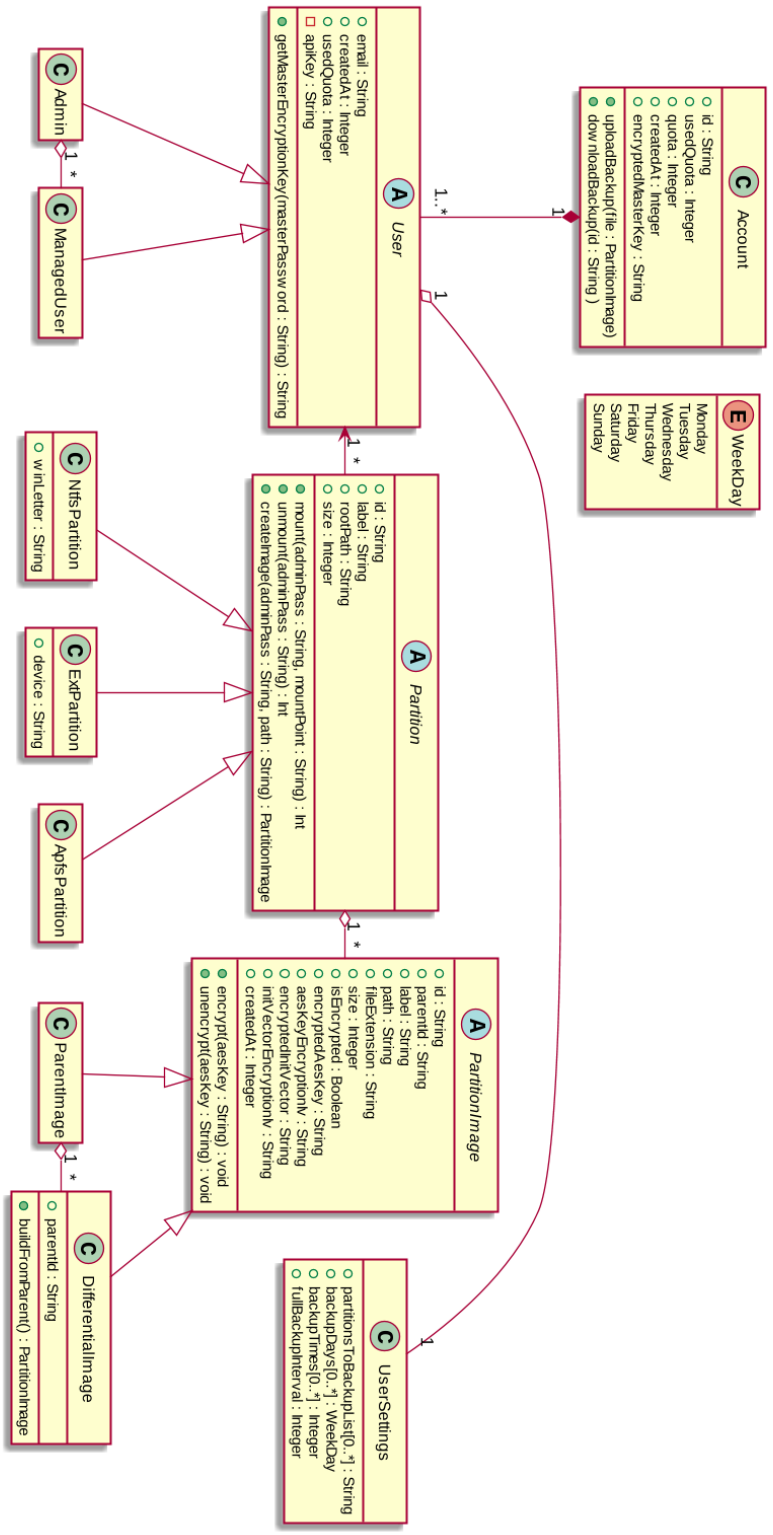

Account
id : String
usedQuota : Integer
quota : Integer
createdAt : Integer
encryptedMasterKey : String
uploadBackup(file : PartitionImage)
downloadBackup(id : String)
WeekDay
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
User
email : String
createdAt : Integer
usedQuota : Integer
apiKey : String
getMasterEncryptionKey(masterPassword : String) : String
Admin
ManagedUser
Partition
id : String
label : String
rootPath : String
size : Integer
mount(adminPass : String, mountPoint : String) : Int
unmount(adminPass : String) : Int
createImage(adminPass : String, path : String) : PartitionImage
NtfsPartition
winLetter : String
ExtPartition
device : String
ApfsPartition
PartitionImage
id : String
parentId : String
label : String
path : String
fileExtension : String
size : Integer
isEncrypted : Boolean
encryptedAesKey : String
aesKeyEncryptionIv : String
encryptedInitVector : String
initVectorEncryptionIv : String
createdAt : Integer
encrypt(aesKey : String) : void
unencrypt(aesKey : String) : void
ParentImage
DifferentialImage
parentId : String
buildFromParent() : PartitionImage
UserSettings
partitionsToBackupList[0..*] : String
backupDays[0..*] : WeekDay
backupTimes[0..*] : Integer
fullBackupInterval : Integer
1
1..*
1
1
1
*
1
*
1
*
1
*

Figure 2.4: Client application - Class Hierarchy Diagram

Web Service Application

In the case of the web service, the class hierarchy diagram can also serve as an entity relation service as Object-relational mapping is the approach chosen to interact with the persistence layer.

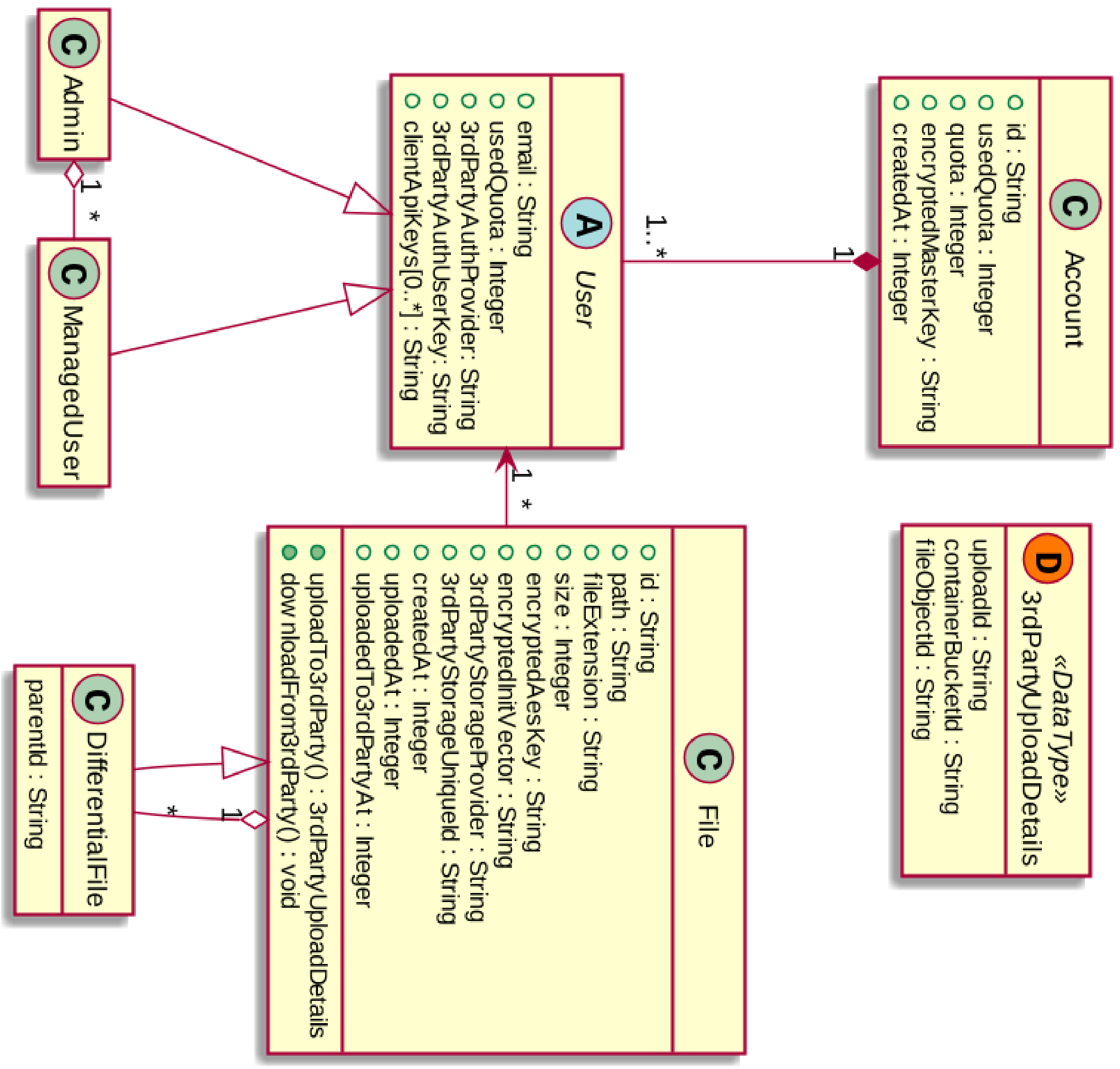


Figure 2.5: Web service - Class Hierarchy Diagram

2.4.3 Client-side Encryption Strategy

Following the market trend of end-to-end encrypted SaaS services like ProtonMail [33] and MEGA [13], all cryptographic operations will be performed on the client applications and their source code will be available to the public for inspection, audit, and even to build their own binaries.

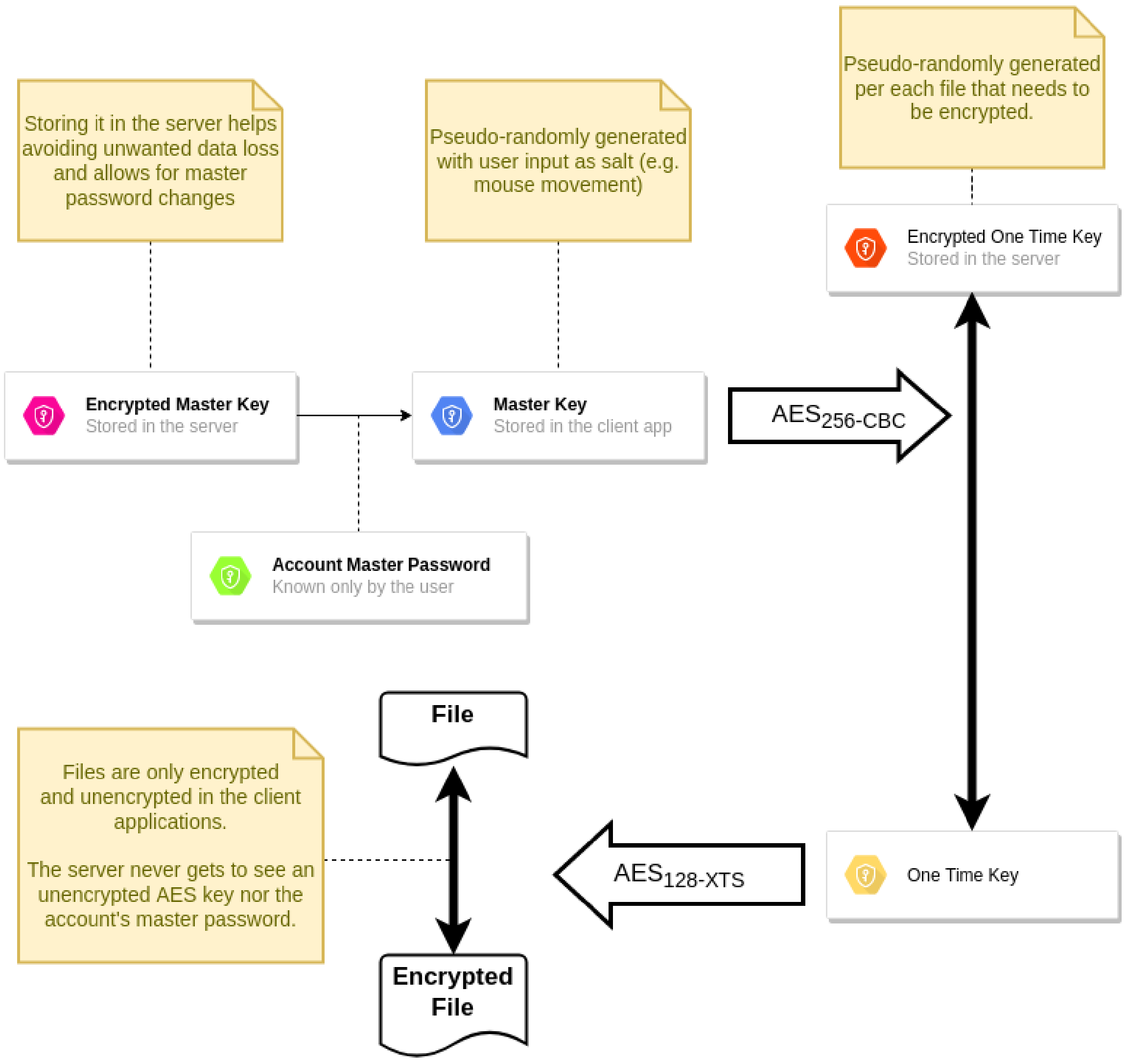

Storing it in the server helps avoiding unwanted data loss and allows for master password changes
Pseudo-randomly generated with user input as salt (e.g. mouse movement)
Pseudo-randomly generated per each file that needs to be encrypted.
Encrypted One Time Key
Stored in the server
Encrypted Master Key
Stored in the server
Master Key
Stored in the client app
AES256-CBC
Account Master Password
Known only by the user
File
Files are only encrypted and unencrypted in the client applications.
The server never gets to see an unencrypted AES key nor the account's master password.
AES128-XTS
One Time Key
Encrypted File

Figure 2.6: Client-side Encryption Strategy Overview

The AES XTS implementation is specifically tailored for disk partitions encryption [26]. In order to ensure a relatively fast encryption process of the disk partition files, the application will use the implementation for 128-bit keys. These keys are way less secure than 256-bit keys, but still impossible to brute force by nowadays computing power [5].

For the one-time keys encryption a more simple AES CBC implementation is used, this time using the 256-bit master key of the account. Given that the account's master key is meant to have a longer life span than one-time keys and that the payload to cipher in this case requires a negligible amount of computing time, it is appropriate to use the more secure 256-bit keys.

Nevertheless, it is important to mention that both ciphers are prone to Man-in-the-middle attacks. A malicious adversary could make minor modifications in the encrypted files, in a way that once the owner decrypts them an apparent working disk partition is generated containing with corrupted data [4]. For this reason, checksum hashes must be generated before and after encrypting every file and then stored together with in the files' metadata repository.

2.5 Operating Environment

The hardware and software platform hosting the service must be able to accept an undetermined amount of data, thus it shall scale automatically to accommodate any user upload request. This scaling necessity must be anticipated and can be performed either at the server's storage or by requesting additional resources to the 3rd party provider.
That said, find a high-level proposal of the desired system environment:

2.5.1 Server Hardware

- One or several arrays of SSD or Hard disk drives in a RAID5 redundant configuration [9].
- One or several load-balanced web servers with sufficient computing power and random access memory resources to handle multiple client connections and I/O operations with the disk arrays.

2.5.2 Server Operating System

- Linux-based distribution compatible with Docker containerization.

2.5.3 Software Stack

- **Web Service:** Express 4.17.1 [22] on top of Node.js 12.16.3 LTS [28]
- **Client Applications Framework:** Qt 5.14.2 for Windows, macOS, and Linux [17]
- **Orchestration Service:** Docker with verified Docker Hub ARM images (`nginx`, `node`, `mongo`) [21]
- **Cold Storage 3rd Party Service:** Amazon Web Services S3 Glacier [16]

C++ and Qt were chosen as the programming language and application framework, respectively, due to their speed for processing I/O and compute-intense operations (cryptographic calculations, in this case) and the ability to work on a single code base that can make builds for multiple operating systems.
Regarding the web service stack, Node.js was the choice because of its asynchronous approach; which becomes really useful in I/O-intensive programs like this one. Docker was chosen as a way to facilitate development of the prototype

2.6 Design and Implementation Constraints

2.6.1 Legal Environment & Compliance

- **GDPR compliance** can require changes and platform upgrades as the size of the operations grow, due to organization size-related rules or legislation changes.
- Depending on where operations are run from it might be required to follow strict **procedures on data destruction**.
- There should be a way to handle subpoenas and data requests from justice and legal authorities and **users should know in advance how much of their data would be leaked** in such a situation.
- There might be additional constraints imposed by **mandatory audits** on data handling and systems security

2.6.2 Software Licensing

- Several components of the Qt framework have not been released in a fully open source license, therefore it might be required to purchase licenses or adopt an alternative if the service ever becomes "for-profit".

2.6.3 Software Limitations

- The Qt framework does not cover all the disk imaging functionality required for the client applications, therefore this part of the application features will be depending on platform dependant solutions and a fully multi-platform match of functionality might not be feasible.
- Any SLA contracted with end-users will be limited by the SLAs agreed with the 3rd party cold storage provider.

2.7 User Documentation

The prototype will not include additional documentation, other than those aimed at the application developers. The first iteration of the product should be easy to use with a trivial minor setup. Future iterations of this product, and certainly after the service Go-Live, will include a website with FAQ and customer support.

2.8 Risks, Assumptions, and Dependencies

2.8.1 Risk Analysis

Several aspects can be considered mid/long term risks for a service like this:

- Many industries, like the film and videogames sectors, are massively adopting cloud computing as a way to serve content via streaming technologies; deeming user devices as simple thin clients. If this trend keeps covering more industries, backups of client storage devices would eventually become irrelevant.
- There are many reasons to believe that cloud object storage prices will go up. One of them is that right now there is a pseudo-monopoly between Amazon, Microsoft, and Google; which can lead to the destruction of other competitors and eventual price abuse. Also, the fact that the carbon footprint of the datacenters sustaining these servers in massive will most likely result in future taxes and fines for these companies; which will eventually increase the retail prices.
- Many organizations already have several active SaaS subscriptions, so every time they are more reluctant to consider adding new services to their stacks. If this service ever becomes a commercial product, appropriate monetization and pricing strategies will likely be key for its success.
- The fact that many organizations have already put in place procedures to store business-critical data in cloud services might make a service like this a harder sell. At the end of the day, businesses whose key digital assets are office suite documents will probably already have them in the cloud; making user workstation potential data losses less of a concern.
- As we rely on the AES algorithm implementations using 128 and 256 bits keys, we are supposed to be covered against brute force attacks for many years to come. In any case, new computing paradigms like quantum computing are evolving fast and may eventually allow ways to make more time-efficient attacks on these cryptographic algorithms.

2.8.2 Initial Assumptions

- All the open-source development tools used will be supported by the FOSS community during the whole service life cycle.
- The cryptography libraries employed are fully secure and not prone to brute force attacks given nowadays achievable computing power; in a reasonable time frame.
- Amazon Web Services S3 or any replacement for cloud cold storage will, at the same time, perform reliable backups in a consistent way of all the data we upload there.
- Both our communications, distributed cloud systems, or 3rd party providers are prone to both intentional or unintentional data leaks and unauthorized access to customer data. Consequently, there is full reliance on client-side encryption, user custody of encryption private keys, and leverage of user authentication to third parties (e.g. social media log in, one-time access links sent via email, ...).
- As deterministic backups when dealing with disk partitions are not feasible [3] and in order to avoid Man-in-the-middle attacks that could affect data integrity, checksum hashes of files, both pre and post encryption, will be stored in the file metadata repository of the server database.

2.8.3 Main Dependencies

- Hot Backups' consistency and availability depend on the RAID array reliability.
- The ability to scale, regarding the number or profile of users, will directly depend on the cloud service resources and storage capacity.
- Disk image creation and restore performance will depend on 3rd party FOSS, as the development of these tools is not in the scope of this project.
- As previously mentioned, all cryptography operations will depend on 3rd party FOSS, as the development of these tools is not in the scope of this project.
- Data recovery and user privacy will completely depend on the ability of the end-users to store their master password confidentially and securely.

Chapter 3

External Interface Requirements

3.1 User Interfaces

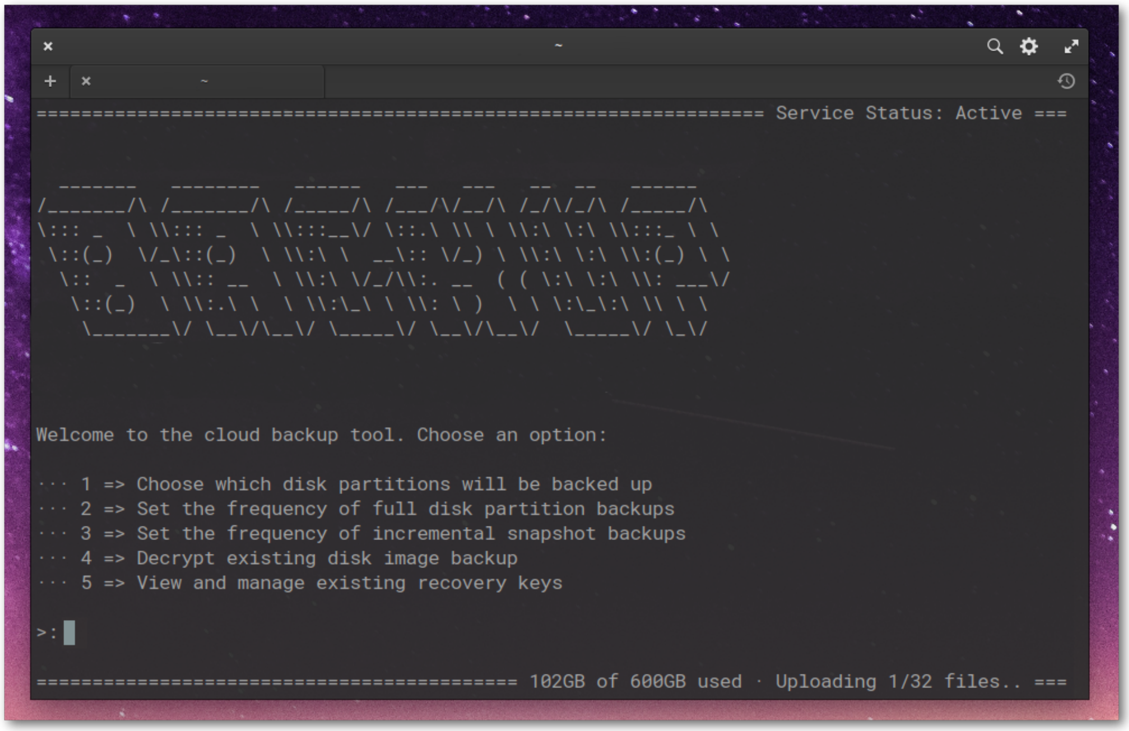
3.1.1 Main Client Interface

The client applications must offer a CLI [3.1] aimed for technical users and system administrators, and a user-friendly GUI, taking advantage of the multi-platform widgets available in the Qt framework. This graphical interface could be inspired in Apple's Time Machine [3.2], in the sense that it should not require prior knowledge of the file system or the disks partition tables for users being able to set up a backup schedule

3.1.2 Backup Recovery Site

The MVP must come with a user private area accessible on the Internet with a web browser [3.3], provided with the same authentication methods as the client applications. In this site, the users with sufficient privileges and not organization-linked users can follow-up the status of all cloud-synced backups and request the retrieval of available backups.

As some backups residing in the cold storage may take some time to be transferred to the cloud service, the user will be shown estimated times for the different retrieval options provided (e.g. direct download, P2P download, hard disks shipment, tape shipment, ...).



```
================================================================= Service Status: Active ===



  _______   ________   ______   ___   ___   __  __   ______
/_______/\ /_______/\ /_____/\ /___/\/__/\ /_/\/_/\ /_____/\
\::: _  \ \\::: _  \ \\:::__\/ \::.\ \\ \ \\:\ \:\ \\:::_ \ \
 \::(_)  \/_\::(_)  \ \\:\ \  __\:: \/_) \ \\:\ \:\ \\:(_) \ \
  \::  _  \ \\:: __  \ \\:\ \/_/\\:. __  ( ( \:\ \:\ \\: ___\/
   \::(_)  \ \\:.\ \  \ \\:\_\ \ \\: \ )  \ \ \:\_\:\ \\ \ \
    \_______\/ \__\/\__\/ \_____\/ \__\/\__\/  \_____\/ \_\/



Welcome to the cloud backup tool. Choose an option:

··· 1 => Choose which disk partitions will be backed up
··· 2 => Set the frequency of full disk partition backups
··· 3 => Set the frequency of incremental snapshot backups
··· 4 => Decrypt existing disk image backup
··· 5 => View and manage existing recovery keys

>:

=========================================== 102GB of 600GB used · Uploading 1/32 files.. ===
```

Figure 3.1: Mock-up of the CLI client

3.1.3 System Restore Utility

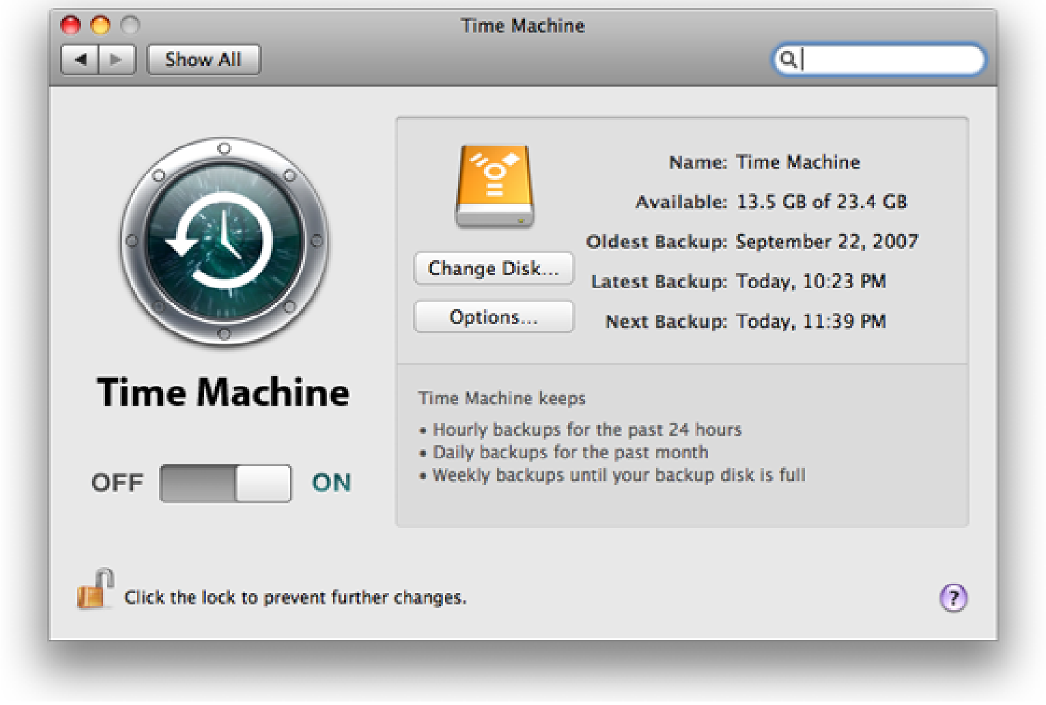
It is not in the scope of this project, but users involved in a data recovery process will require the use of a disk image restore utility to mount recovered disk partitions in their systems. A FOSS option for this scenario could be Clonezilla [3.4], which is a popular safe option that runs from a Live CD but lacks a friendly user interface.

3.2 Hardware Interfaces

3.2.1 Storage Redundancy

It is suggested to employ a RAID 5 redundant array for storage, which means that with a minimum number of three drives it will only sacrifice one disk per array for redundancy handling and gain fault tolerance for one disk failure.

$$RAID5_{capacity} = (\# \texttt{ of drives} - 1) * \texttt{Size of the smallest drive}$$


Name: Time Machine
Available: 13.5 GB of 23.4 GB
Oldest Backup: September 22, 2007
Latest Backup: Today, 10:23 PM
Next Backup: Today, 11:39 PM

Time Machine keeps
- Hourly backups for the past 24 hours
- Daily backups for the past month
- Weekly backups until your backup disk is full

Click the lock to prevent further changes.

Figure 3.2: Apple^{TM} Time Machine's User Interface (©*pngkey.com*)

Anyone involved in the materialization of this project must analyze if this level of redundancy still applies appropriate for a production environment. In case of requiring higher read/write performance, RAID10 would be a better option, and in case of requiring raising disk failure tolerance to at least 2 then RAID6 could be a good alternative at a similar performance level.

3.2.2 Data Retrieval in the form of Removable Storage

The production service shall offer the possibility of retrieving data in some form of removable storage via a courier/delivery service. These could be external hard drives, tape or optical disks; and, if viable, the user should be able to return the hardware once the data recovery process is finished and get a partial refund.

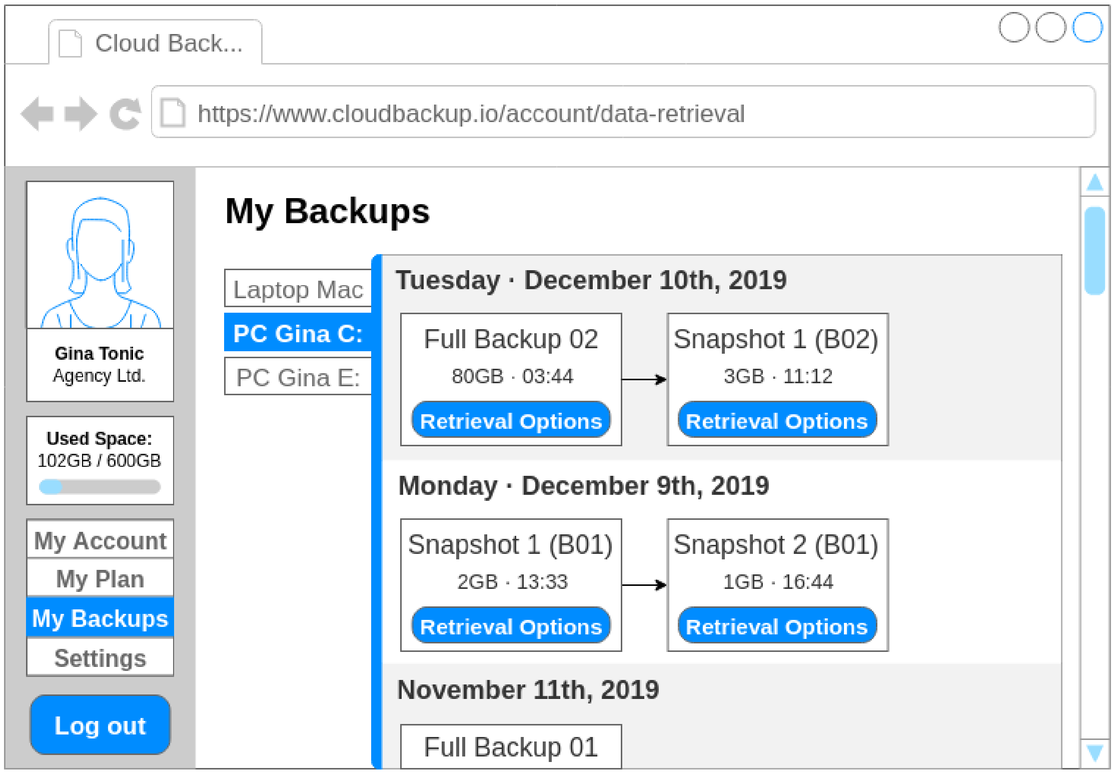


Figure 3.3: Mock-up of the backup recovery section of user private area

3.3 Software Interfaces

3.3.1 Software Libraries

- **Qt Framework** [17]

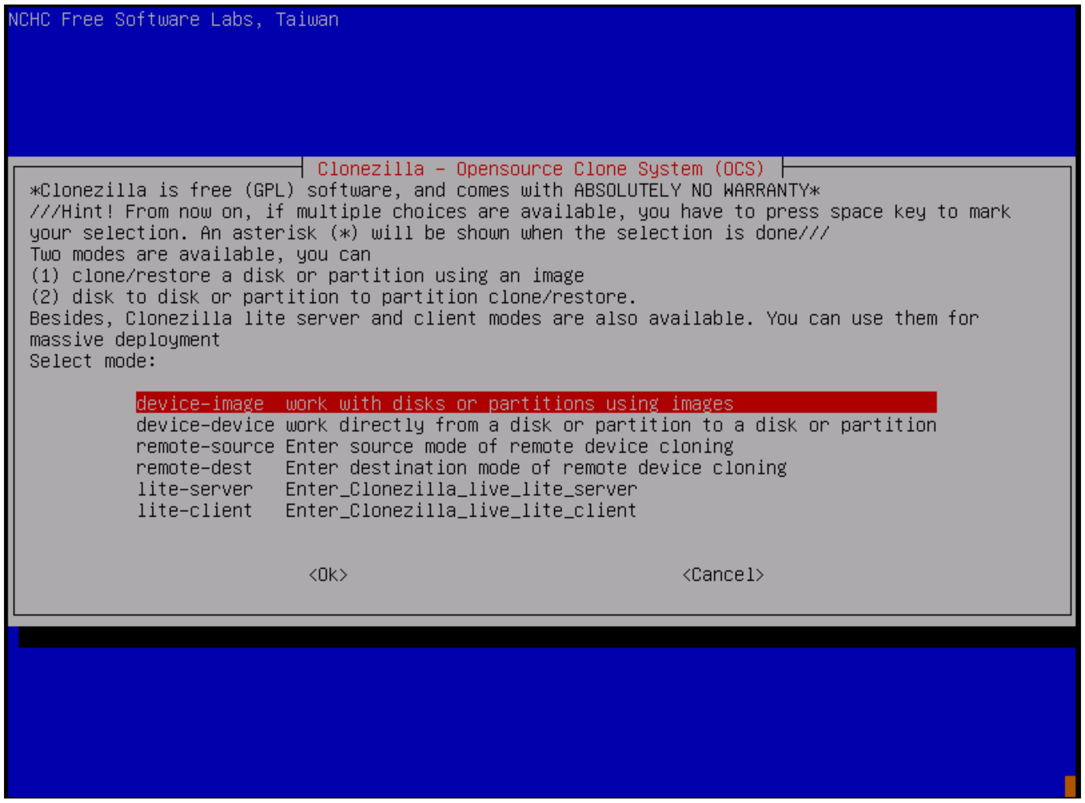
  A framework that facilitates creating multi-platform builds and user interfaces in the C++ programming language. It also includes I/O and file integrity checking functionality that will be handy for this project.

- **Botan** [40]

  C++ library with implementations for all the cryptographic operations necessary for the service functionality, in this case, several implementations of AES-based algorithms.

- **Express.js** [22]

  Library for building web services and REST APIs with Node.js. It will be the core



```
NCHC Free Software Labs, Taiwan

Clonezilla - Opensource Clone System (OCS)
*Clonezilla is free (GPL) software, and comes with ABSOLUTELY NO WARRANTY*
///Hint! From now on, if multiple choices are available, you have to press space key to mark
your selection. An asterisk (*) will be shown when the selection is done///
Two modes are available, you can
(1) clone/restore a disk or partition using an image
(2) disk to disk or partition to partition clone/restore.
Besides, Clonezilla lite server and client modes are also available. You can use them for
massive deployment
Select mode:

          device-image  work with disks or partitions using images
          device-device work directly from a disk or partition to a disk or partition
          remote-source Enter source mode of remote device cloning
          remote-dest   Enter destination mode of remote device cloning
          lite-server   Enter_Clonezilla_live_lite_server
          lite-client   Enter_Clonezilla_live_lite_client

                         <Ok>                              <Cancel>
```

Figure 3.4: Clonezilla (*GNU GPL License*)

of the cloud service API.

3.3.2 3rd Party Software

- **Partclone** [32]

  Open-source utility for easy creation of disk partition images in Linux-based operating systems.

- **dd (MacOS Mojave)** [18]

  UNIX utility that provides several ways to copy files. The corresponding client application it to create disk partition images in macOS systems.

- **ODIN** [29]

  Open-source application for easy creation of disk partition images in Windows operating systems.

- **OSFMount** [31]

  This application can be used by users of the platform to mount their disk images in Windows systems.

- **Clonezilla** [15]

  A Live CD that boots into an open-source disk partition manager with the ability to create disk partitions from image files. It can be recommended to users of the platform, independently of the operating system that they normally use, to perform a system restore from a disk partition image backup of their own.

- **HAProxy** [41]

  High availability load balancer. It allows the web service to distribute incoming requests among the spawned servers, allowing for better response times and service availability.

3.3.3 3rd Party Cloud Services

- **Amazon Web Services S3 Glacier** [16]

  Cloud storage provider for the backup and/or Cold Storage needs of the platform. It allows object direct uploads and downloads via a REST HTTP/HTTPS API, and also physical pickup and delivery of tape.

- **Google Firebase Authentication** [23]

  Cloud service that integrates several social media authentication providers in a single API. They provide SDKs for many programming languages and a REST API.

- **Docker Hub** [20]

  The official online repository of Docker images.

3.3.4 Database & Persistence Software

- **mdadm** [27]

  RAID array controller software for Linux-based operating systems.

- **rclone** [35]

Fork of the popular file and folder synchronization application, *rsync*, that facilitates integration with several cloud object storage providers out of the box.

- **SQLite** [38]

  A lightweight relational database contained within a single file. To be used in the client applications.

- **MongoDB** [42]

  An easy to use NoSQL Document-oriented Database management system. To be used in the web service.

3.4 Communications Interfaces

- **HTTP/HTTPS**

  Most communication between systems will be carried over HTTP connections, some of them SSL secured, since both the cloud service and the 3rd party providers will interact with other components of the system via REST API.

  User personal data exchanges and authentication must be performed over secure connections, while this will not be necessary for backups uploads/downloads given that the payload encryption method used to be sufficient to guarantee data confidentiality.

- **BitTorrent**

  In future iterations of the product and use cases where it is required to serve the same backup to several clients, it would be interesting to consider using P2P technology, to increase transfer rates and reduce server load. The BitTorrent protocol has several implementations for this purpose.

Chapter 4

System Features

4.1 Client Applications Features

This section contains the common features and requirements among the client-side applications of the service, regardless of the platform/operating system being used.

4.1.1 Disk Partition Backup & Encryption

Description and Priority

Description: Users will be able to select which disk partitions will be copied as a disk image, encrypted within the client application, and uploaded to the cloud service in the background. They will be able to choose when full backups will take place and when will it be a differential snapshot instead. It will also be possible to set the schedule and/or frequency when these will happen.

Priority: **High**

Functional value added: ●●●●●●●●○

Operational cost: ●●○○○○○○○
Level of risk involved: ●●●●●●●○○
Development effort required: ●●●●●●●●●

Stimulus/Response Sequences

Stimulus: The user selects a supported disk partition in the list.
Response: The application shows the backup strategy and scheduling settings available.

Stimulus: The user chooses a new backup strategy and/or scheduling setting for a certain disk partition.
Response: The application shows the backup strategy and scheduling settings available.

Stimulus: User toggles the general switch for enabling / disabling all backup operations.
Response: The application will execute or ignore all scheduled tasks, when required, according to the state of the switch.

Functional Requirements

- **BKE-1:** The application shall be able to scan, list, and show to the user all the available compatible disk partitions ready to be backed up.
- **BKE-2:** The application shall be able to generate disk images from the partitions selected by the user and immediately encrypt them using the 128-bit XTS [26] implementation of the AES algorithm.
- **BKE-3:** The application shall be able to upload files to the cloud service together with an encrypted version of its encryption key, initialization vector, and checksum calculation or equivalent; while making sure that all the metadata gets related to the user's account in the user repository.
- **BKE-4:** The application will upload a disk image to the cloud service if, and only if, it is properly encrypted.
- **BKE-5:** The application must execute any tasks scheduled by the user at proper times as long as its daemon process is running.
- **BKE-6:** The application must allow the user to set up the backup strategy regarding the frequency of full disk partition backups and differential backups for every partition available.

- **BKE-7:** The application must allow the user to define a schedule for backups to be automatically generated, encrypted, and uploaded.
- **BKE-8:** The application must allow the user to disable and re-enable backups completely by triggering an easy to access switch.
- **BKE-9:** The application shall abort the operation and notify the user when a task cannot be performed due to lack of disk space, unavailable Internet connection, or bad response from the cloud service.
- **BKE-10:** In case a disk partition is no longer accessible, its configuration will remain stored and visible to the user but no tasks will be executed and the user will be informed of the situation.
- **BKE-11:** The application must allow the user to generate, encrypt, and upload a disk partition backup (full or differential snapshot) at will and with immediate processing; regardless of the scheduled ones.

4.1.2 Encryption keys management & Decryption

Description and Priority

Description: The approach regarding encryption keys management will be based in reliance on randomly generated master keys - one per account - which will be encrypted with the account's master password and uploaded and stored in the cloud service [4].

The process will consist of:

1. A random master key generated at the account's creation time, which will be encrypted with the master password chosen by the user and uploaded to the server together with a hash of the user's master password.

   Master Key = $Rand_{256\text{-bit}}(\textit{From user-inputted noise})$
   Encrypted Master Key = $AES256_{ECB}(\text{Master Key}, \text{Master Password})$

2. Every time a backup is created a new unique encryption key and a nonce (initialization vector) will be generated. These keys will then be encrypted using the master key and uploaded to the server together with the encrypted backup payload and checksum calculations; prior and posterior to the encryption process.

One-time Key = $Rand_{\text{128-bit}}$
IV (Nonce) = $Rand_{\text{64-bit}}$
Encrypted Backup = $AES128_{XTS}$(Backup Payload, One-time Key, IV)

3. The server will deliver its copy of the encrypted master key to any client application that can guess the password's hash. The client application can then decrypt it with the user's master password and store it locally, in case of performing a fresh install of the application.

4. In case that a client application requests a master password change to the server, the latter will allow it to update the encrypted master key; this time encrypted with the new password. This allows the user to secure existing backups in case his/her master password is compromised.

Priority: High

Functional value added: ●●●●●●●○○

Operational cost: ●●●○○○○○○
Level of risk involved: ●●●●●●●●●
Development effort required: ●●●●●●●●●

Stimulus/Response Sequences

Stimulus: The user clicks on the login button.
Response: The application opens a web view and redirects the user to the authentication site of the web service.

Stimulus: The user completes the authentication flow in the web service site successfully.
Response: The client application stores the newly generated auth token and the existing encrypted master key; both sent from the web service. It then prompts the user to enter the account's master password to decrypt the master encryption key.

Stimulus: The user selects the option to decrypt a backup retrieved from the cloud service and provides the payload and its associated encrypted key and nonce.
Response: The application decrypts the provided keys using the locally stored master

key and uses the result to decrypt the payload and make it available to the user.

Stimulus: The user selects the option to see the encryption master key.
Response: The application displays the key to the user in plain text. (i.e. if this master key is compromised, every one-time-key encrypted with it and all backups associated must be considered compromised).

Functional Requirements

- **ENC-1:** The client application will use the authentication service provided by the product's web service to identify the user. It will provide the application with a copy of the encrypted master key and a token to keep the user logged in indefinitely; until there is a logout request or until there is a password change.
- **ENC-2:** The application must be able to decrypt any previously encrypted backup with its unique (encrypted) pair of encryption key and nonce, and the encryption master key stored locally.
- **ENC-3:** The user must be able to see his/her locally stored encryption master key in plain text at any moment without requiring an Internet connection.

4.2 Web Portal Features

4.2.1 User Accounts Management

Description and Priority

Description: User accounts creation, management, and authentication will be performed through a web interface on the Internet. Even while logging in from a standalone client, a web view or browser window will be open to complete the login flow and provide the native application with an auth token.

Priority: High

Functional value added: ●●●●●○○○○

Operational cost: ●●●●○○○○○
Level of risk involved: ●●●●●●●○○
Development effort required: ●●●●●●○○○

Stimulus/Response Sequences

Stimulus: The user selects the option to log in to a social media site to create an account.
Response: The site attempts to log in using the 3rd party provider API for social login. If successful, then prompts the user to provide any required personal data that could not be fetched from the user's social profile, asks him/her to confirm, and creates the account. The system then sends an email to the user requesting action to confirm that the address is correct.

Stimulus: The user selects the option to create an account.
Response: The site prompts the user to provide an email address and other required personal data, asks him/her to confirm, and creates the account. The system then sends an email to the user requesting action to confirm that the address is correct.

Stimulus: The user confirms his/her email address.
Response: The system prompts the user to provide randomized input to generate a pseudo-random master encryption key and to choose a master password to encrypt and store the newly created key in the server.

Stimulus: The user with existing account requests to login with a one time link.
Response: The system voids any existing active temporary access link and sends a new one-time access link to the user's private area via email.

Stimulus: The user with an existing account selects the option to log in to a social media site, to then log in to his/her private area.
Response: The site attempts to log in using the 3rd party provider API for social login. If successful, then tells the web service that access was granted and the user is redirected to his/her private area.

Functional Requirements

- **UAM-1:** The portal must allow creating an account by asking for an email address and the minimum amount of personal data as possible.
- **UAM-2:** The portal shall allow creating an account by using a social media login.
- **UAM-3:** The portal will send an email verification link to the user after creating a new account.
- **UAM-4:** Once the user validates the email account of a newly created account, the site will prompt the user to input randomized data (e.g. by moving the cursor, ..) to generate a pseudo-random master encryption key on the client-side.
- **UAM-5:** Once a user's master encryption key is generated, the user will be prompted to input a secure master password which will be used to encrypt the key on the client-side, and then will be uploaded to the server.
- **UAM-6:** The portal must allow users to login without requiring a password, by sending a one-time access link to the user's email account.
- **UAM-7:** One-time access links must expire when used, when another link is generated, or after a reasonable amount of time, according to security concerns.
- **UAM-8:** The portal shall allow users who created their account with the social media credentials, to authenticate themselves by logging in that social media provider.
- **UAM-9:** The portal must allow the user to delete all his / her private data, backups, and/or the whole account.
- **UAM-10:** The portal will eventually allow users of organizations to invite additional managed users to share their accounts.
- **UAM-11:** The user must not be able to interact with his / her newly created account until the associated email account is validated.
- **UAM-12:** If an email address is not validated in a reasonable amount of time, according to security standards, the newly created account associated will be completely removed.
- **UAM-13:** The user must be able to modify his / her private data and email address, and the latter case will initiate a new email address validation process.

- **UAM-14:** The user must be able to change his / her master password, in which case the encryption master key will be encrypted with the new password on the client-side and then replaced in the server.
- **UAM-15:** The user master password must be considered secure according to current cryptography standards in regards to tolerance to brute force attacks for the web service to allow it in the first place.

4.2.2 Cloud Backup Management & Data Retrieval

Description and Priority

Description: The web portal will be the interface that users will use to manage and retrieve their cloud-synced backups. It will feature a chronological view [3.3] where users will be able to easily find their backups and choose different retrieval options.

Priority: **Medium**

Functional value added: ●●●●●●●○○

Operational cost: ●●●●●●●●●
Level of risk involved: ●●●●●○○○○
Development effort required: ●●●●●●○○○

Stimulus/Response Sequences

Stimulus: The user chooses to retrieve a certain full backup or snapshot.
Response: The server will fetch all dependencies, calculate the total size, and provide the user with the available options for data retrieval.

Stimulus: The user requests to download a certain backup.
Response: If the files are available, a direct download will begin immediately. If files need to be fetched from the cold storage provider, the user will receive an alert via email when files are ready to download.

Stimulus: The user requests to retrieve a backup via the physical delivery of a removable storage drive.

Response: The portal will log the request and send a confirmation to the user via email.

Functional Requirements

- **CBK-1:** The user must be able to see all his / her full and partial backups and related metadata; including file system type, date of creation, date of upload, device of origin identifier, encrypted the one-time encryption keys, and the user who uploaded.
- **CBK-2:** The user must be able to delete any full or partial backup, together with all its dependencies.
- **CBK-3:** The user must be able to download any full or partial backup, together with all its dependencies.
- **CBK-4:** In case that a backup requested by the user for direct download is not available at the moment, the system will queue any tasks required to make it available and will send an alert to the user via email when it becomes available.
- **CBK-5:** The user must be able to log a request for physical delivery of any full or partial backup in removable storage, and get a confirmation and follow-ups via email.

4.3 Server Features

4.3.1 Cloud Backup Service

Description and Priority

Description: The cloud backup service will store and serve user backups, to the web service and standalone clients of the platform, together with their metadata and encrypted versions of their unique encryption keys. It will also check file integrity whenever a checksum is available.

Priority: **High**

Functional value added: ●●●○○○○○○

Operational cost: ●●●●●●●●○
Level of risk involved: ●●●●●●●●●
Development effort required: ●●●●●●●○○

Stimulus/Response Sequences

Stimulus: The client application uploads a backup.
Response: The server stores the payload and adds an entry to the the database linked to the user's account containing the file's metadata and unique encryption key encrypted with the account's master key.

Stimulus: The client or web service requests a certain backup.
Response: The server looks for it in the hot storage and serves it directly, or requests it to the cold storage and returns an estimation for its availability.

Stimulus: The client or web service issues a delete request.
Response: The server looks for the requested backup both in the hot storage and cold storage and deletes all copies.

Functional Requirements

- **BSV-1:** The server will be constantly listening to HTTP/HTTPS requests from standalone clients and web service.

- **BSV-2:** The server will only process requests that contain a valid API key for an active account, and will limit the scope of data they can acquire and manipulate to that account's records.

- **BSV-3:** The server must implement RAID5 array redundancy in its hot storage disks.

- **BSV-4:** The server will only accept incoming backup uploads consisting of the encrypted payload, metadata, checksum, and unique encryption keys; the latter themselves encrypted with the account's master key.

- **BSV-5:** The server will initially store the payload of backups in its hot storage file system and save the metadata in the service database.
- **BSV-6:** The server will serve files requested by a client/web service together with their metadata directly from the hot storage.
- **BSV-7:** If the server does not find a requested file in the hot storage, it will issue a download request from the cold storage provider to the hot storage and estimate the remaining time to complete the operation.
- **BSV-8:** The server will delete files from the hot storage and issue delete requests to the cold storage provider whenever instructed by the web service.
- **BSV-9:** The server will deny uploads if the user account available space usage has been exceeded.

4.3.2 Cold Storage Backups Sync

Description and Priority

Description: This service will proactively move backups between hot and cold storage (3rd party) according to security, efficiency, and availability criteria.

Priority: Low

Functional value added: ●●●○○○○○○

Operational cost: ●○○○○○○○○
Level of risk involved: ●●●●●●●○○
Development effort required: ●●●●●●●●○

Functional Requirements

- **CSS-1:** The server shall make sure that no storage request to the hot storage will be denied by a lack of disk space by uploading existing files to the cold storage provider and liberating space.
- **CSS-2:** The server criteria for choosing files to be moved to cold storage must favor service efficiency according to access patterns and user activity.

- **CSS-3:** The server will permanently delete long term files according to the user accounts plans/agreements.

Chapter 5

Other Nonfunctional Requirements

5.1 Performance Requirements

The web service must adhere to the SLA terms negotiated with end-users, which will always be less restrictive than the SLA signed with the cold storage 3rd party provider.

5.2 Safety Requirements

Independently of the cloud service operation, it will be necessary to perform recurring backups of the whole platform databases and user files stored either in hot or cold storage. These will be the last resource of data recovery in case of major systems failure, so it can rely on long-term bulk storage technologies like tape or optical disks.

5.3 Security Requirements

The service must comply with GDPR and other data protection laws applicable wherever the server is located and the clients are invoiced.

The organization running operations will also be required to comply with any mandatory audit regarding data protection in the server's jurisdiction before going live.

It cannot be possible for the service administrators to retrieve any non-encrypted user encryption key or master password, and they will never facilitate ways to force users to reveal their not-encrypted encryption keys or master passwords to law enforcement or justice authorities. At the same time, software developers involved in the product creation or maintenance will never implement back-doors or spyware in any of the client or web portal releases.

5.4 Software Quality Assurance and Testing

Both the standalone client applications and the web portal must be easy to use by users with basic computer skills without the need for extensive documentation. In the GUI desktop versions, setup shall be similar to Apple's Time Machine [3.2] and the application itself shall be lightweight in terms of storage and memory usage (without taking space used by backup files into account) when compared with similar applications.

Every graphical interface must be clean and be transparent and informative regarding the handling of user data, level of confidentiality achieved, and risks involved in every data transfer or data manipulation operation.

5.4.1 Testing Strategy

Given that the service runs in a sort of distributed system and client applications compatible with different operating systems and device types, it is critical to maintain a comprehensive test suite that guarantees permanent service availability and system integrity during new releases.
The proposed strategy consists of two different approaches to **automated testing**:

1. **End-to-end Testing:** Since a critical aspect of the distributed system operations is communication between components, it will be mandatory to have automated tests that can directly interact with the applications' UIs and make sure that every functionality is working correctly.

   These should cover, at least, every major use case of the service and run automatically in a schedule and, in addition to that, every time that a new software release is about to be deployed.

Solutions like Katalon [25], TestComplete [37], and Selenium [36] can be used to develop these tests and in different scenarios that cover all the target operating systems and platforms of this project.

2. **Unit Testing:** Maintaining applications that support so many different operating systems and the fact that several programming languages are involved, implies that some or all developers will work in several codebases.

   In these situations, it almost mandatory to implement and execute automated unitary tests before every commit to the development code repository. Doing so will minimize the risk of bugs and can help anticipate errors that may affect more than one component of the distributed system before end-to-end tests kick in; which are generally much slower and resource-consuming than unit tests.

   For unit tests coding it is advised to use the standard or *de facto* standard tools provided by the programming language or application framework that the code is written for.

5.5 CI/CD Requirements

Deploy of new releases will involve the provision of several distributed systems, execution of containerized applications, and execution of automated unitary and end-to-end tests; which raises the necessity for automated software deployment workflows.

Since the suggested approach for the evolution of this service from the current prototype to the MVP is an iterative approach in an Agile fashion, it would be interesting to consider applying DevOps principles to the continuous integration and continuous deployment strategies. Another fact that supports this approach is the requirement for automated scaling of the system, which will require setting up rule sets and automated workflows in the production environment so new web service instances and storage resources are provisioned just in time.

5.6 Business Rules

- Users will be limited to access data created and uploaded by them. Initially, there will be no feature that allows sharing between accounts, therefore each account operation will be limited to its data domain.

- Service will be interrupted once an account reaches the storage limit set for it. Only data retrieval will be possible, which may incur additional costs payable by the user according to the plan subscribed.

Chapter 6

Prototype Implementation

To illustrate the feasibility of the distributed system architecture, technology stack, and encryption and data handling strategies proposed — a prototype covering a subset of the MVP expected functionality has been implemented in a limited hardware platform and a Linux client.

Source code for this implementation should have been provided together with this document.

6.1 General Use Case: Backup and Upload Partition

The goal of the prototype is to cover one of the main use cases, as a matter of fact; one that all active users of the service will eventually execute. It is use case number II [2.2], the first backup of a disk partition where the process is user-initiated manually.

6.1.1 Description

The user wants to generate a one-time encrypted full backup of one of the disk partitions in his/her system, and then upload it to the cloud service.

This use case covers the typical scenario after the first time install of the software, where the user wants to build a first time backup of their disks. The software should encourage the user to perform it as soon as possible.

Once the case use scenario is completed the user will have the possibility to retrieve the file, and/or generate and upload differential snapshots to keep the backup up to date with the current status of the system partition.

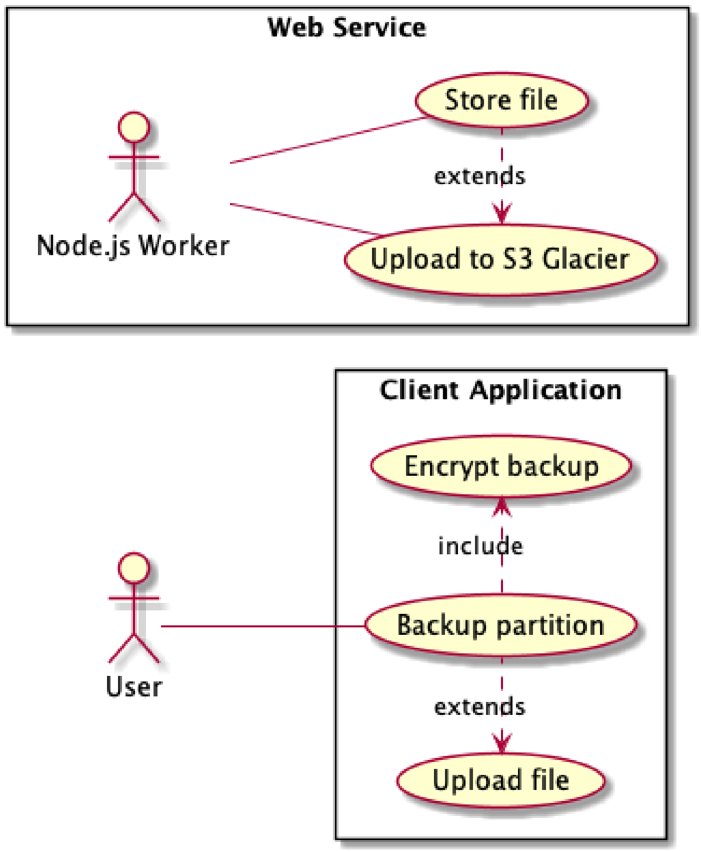


Figure 6.1: Use Case II: Actors and Dependencies [2.2]

6.1.2 Main Scenario

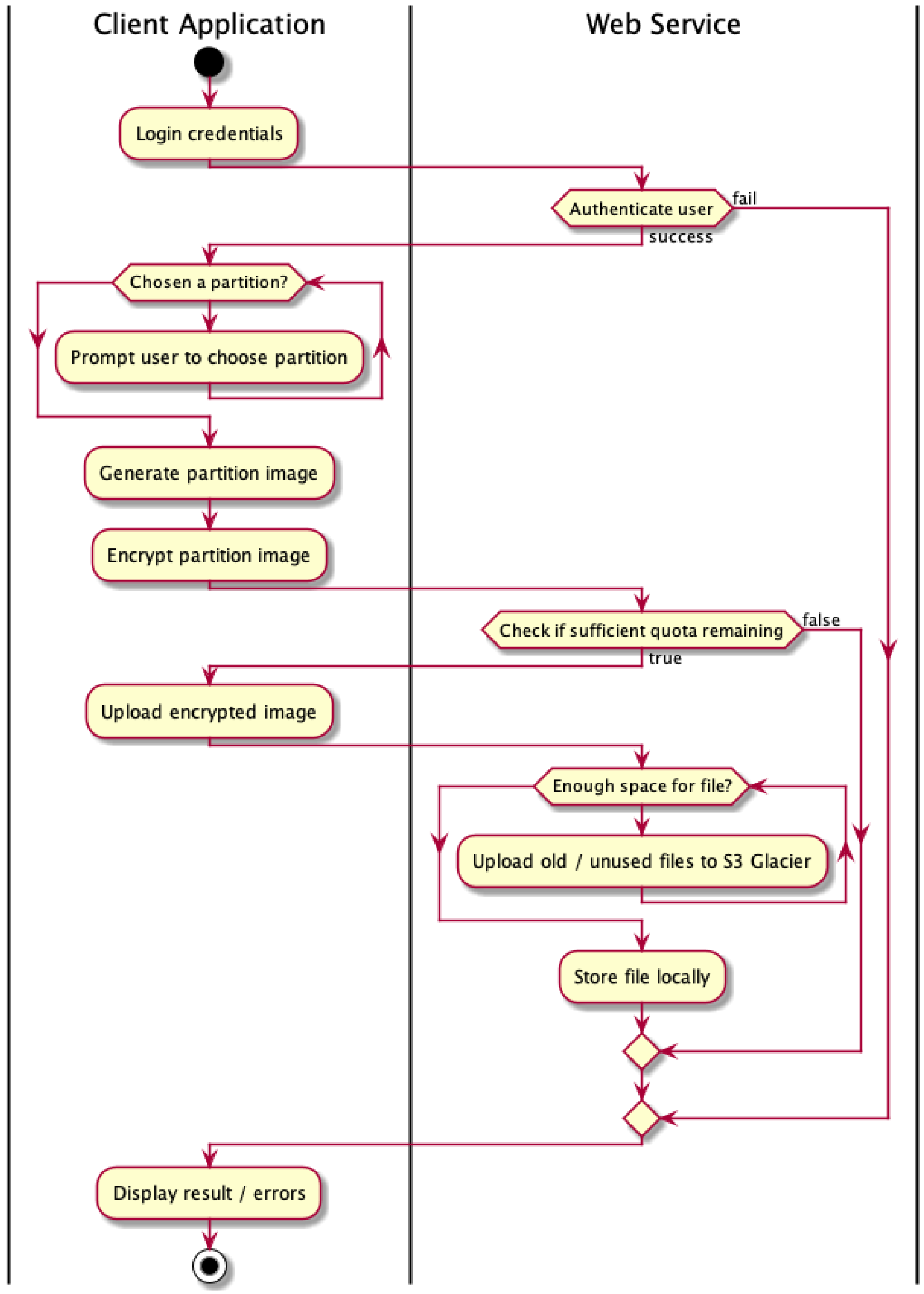


Figure 6.2: Use Case II: Flow Diagram [2.2]

6.1.3 Preconditions

1. The client application must have been installed in the user's system and been given the required privileges to access the application storage space and system partition table.
2. The user must have an active account registered in the web service with an existing pair of a master encryption key and master password, and they need to be added to the environment variables file of the client application.
3. The partition to be backed up must be of type ext3/ext4 since the prototype will only work in Linux-based operating systems.
4. The disk partition cannot contain the Linux root, as it needs to be unmounted.
5. The file system must have enough free space to accommodate a copy of the target partition.
6. The user's system has a stable connection to the Internet with sufficient bandwidth to allow the upload operations.
7. The web service must be fully operative at the address specified in the environment variables file of the client application.

6.1.4 Post-conditions

- The encrypted backup of the user's chosen partition is stored at the very own storage of our cloud service immediately after finishing upload.
- The encrypted backup will eventually be moved from the remote server to Amazon S3 Glacier, according to the server's *cron* scheduler criteria for executing *rclone* and syncing files with the S3 bucket.
- Retrieving/Downloading a file that was moved to the 3rd party cold storage will take more time to process than those stored in the cloud service itself.
- The user account's used space quota will be updated according to the final file size of the encrypted backup.
- The client application will allow the user to generate snapshots (differential backups) of the partition which will be labeled as dependant on the existing full backup.

6.2 Hardware Stack

- Raspberry Pi 4 Model B - Quad core 64bit ARM v8 processor @ 1.5GHz with 4GB SDRAM
- Toshiba Canvio Basics 1TB External Hard Drive (4x on a redundant RAID5 array)

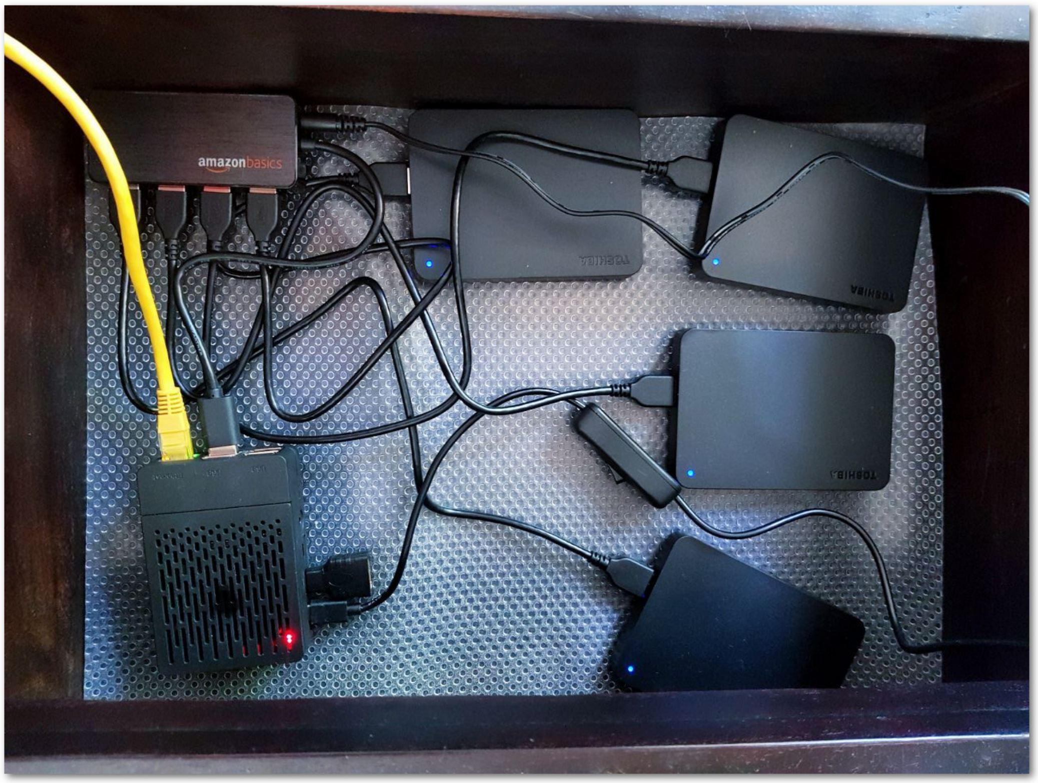


Figure 6.3: Prototype: Web Server & Hot Storage Disks Array

6.3 Specific Software Stack

- **Operating System:** ”Raspbian” Debian Buster with GNU Linux Kernel v4.19 running a SSH/SCP server with public/private key authentication [19]
- **Web Service:** Express 4.17.1 [22] on top of NodeJS 12.16.3 LTS [28]
- **Client Applications Framework:** Qt 5.14.2 for Linux [17]

- **Orchestration Service:** Docker [21] with verified Docker Hub ARM images (`node`, `haproxy`, and `mongo`) provisioned via Docker Compose [12].

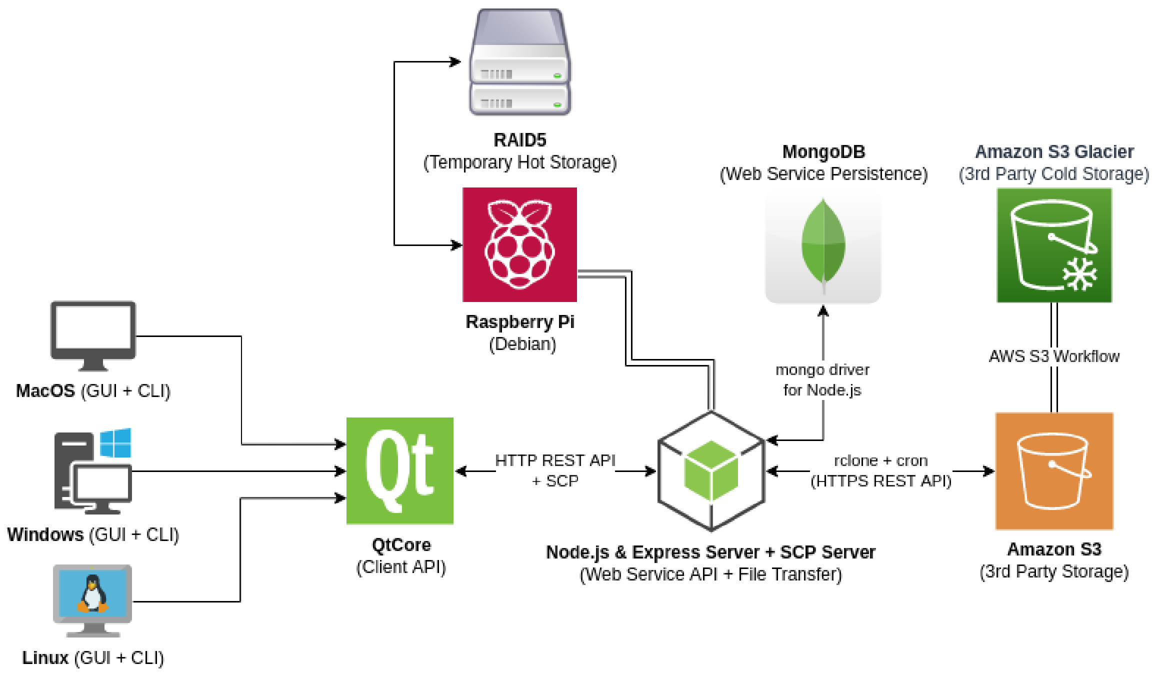


Figure 6.4: Prototype: Solution Overview

6.3.1 3rd Party Resources

- Cold Storage: Amazon Web Services S3 and S3 Glacier [16]
- Dynamic DNS: FreeDNS/Afraid.org [14]

Chapter 7

Conclusion

Thanks to all the work of the academic community on cryptographic algorithms and the work of the FOSS community on developing abstraction libraries for these algorithms, it is safe to claim that developing an end-to-end encrypted service is a pretty straightforward journey once the data securitization and credentials management strategy has been sorted out.

The abuse of user data collection by Internet companies, the increase in data loss events caused by malware, and the interest in blockchain and more transparent and decentralized technologies are a promising environment for the proliferation of SaaS services where respect for user data is the main value. Therefore, I believe this project could be a good starting point for any new Internet service that shares these values.

7.1 Project scope changes and other eventualities

As mentioned in the introduction and evidenced in the initial project roadmap, a change of scope regarding the goals of the prototype took place soon after commencing work. A more realistic analysis of the resources available and the project deadline resulted in removing graphical interface design from the prototype implementation, together with some of the main use cases of the product. At the end of the day, we believe that the resulting prototype serves its purpose and proves that a viable product can be developed with the software specification and technology stack proposed.

The other main eventuality we faced was the cryptographic algorithms library choice. Initially, the system was intended to use the Qt Cryptographic Architecture [34] library;

which proved to be quite outdated for the Qt Framework version used and to have almost no documentation available online. Then we decided to pivot to the popular OpenSSL [30] application, which is widely documented but developing the client with it was quite slow due to the low-level approach of its C++ library. And, finally, Botan [40] turned out to be the cryptography library of choice, due to its great documentation and process-oriented method.

7.2 MVP Roadmap

As next steps to iterate from the current prototype towards an MVP, one should consider the following:

1. Daemonize the application, so disk partitions that hold the operating system root directories can be unmounted and backed up during system shutdown [39] or startup.
2. Elaborate on the physical delivery options for backups and design the workflows and quality assurance procedures that the staff in charge of delivering these should follow.
3. Design and build the user interface and clients for all supported operating systems.
4. Design and build the website containing the users' private area and the user authentication flows.

7.3 Future of the project

As additional considerations, these topics deserve further investigation and elaboration for whoever aiming to turn this project into a commercial SaaS product:

- **Master encryption key compromised scenario**

  What are the options if an account's master encryption key is compromised? Shall all existing backups be re-encrypted? Would that be cost-effective and sustainable for the system?

- **Pricing structure**

  Could a freemium model work for a SaaS like this? A pay-as-you-go model maybe? Could hardware selling be a profitable source of income?

- **Business Development Strategy**

  Is this a SaaS that could work with an inbound sales approach? Which market segments could be more interesting to start targeting? B2B? B2C? Both?

Glossary

CLI Command Line Interface. 8, 28, 29

Cold Storage Storage service intended to host files, typically big in size, that are meant to stay not accessed or modified for long periods of time.. 16, 17, 33

Daemon Background non-interactive process running in an operating system.. 58

DevOps Continuous Integration and Continuous Deployment set of practices that encourages collaboration between development teams with IT system administrators in charge of software deployments.. 49

FAQ Frequently Asked Questions. 26

FOSS Free or Open Source Software. 27, 29

GUI Graphical User Interface. 28

Hot Backups Full, partial or incremental disk image backups which are stored in our in-house storage but have not been uploaded to the Cold storage service.. 27

I/O Input / Output. 24

Live CD Operating System that runs directly from removable storage without requiring installing files in the disk, only RAM usage.. 29, 33

Man-in-the-middle attack Systems penetration attack in which a third party modifies the payload of a communication between two or more parties without their knowledge nor their authorization.. 23, 27

MVP Minimum Viable Product. 11, 49, 51

NoSQL Any database management system design paradigm that is other than the relational approach. Generally refers to document-oriented databases and key-value stores.. 34

Object-relational mapping Data model management strategy in which the class data models in an application are abstractions of the underlying database structure. It allows for seamless materialization of changes in the application data model in the database, and vice-versa.. 21

SLA Service Level Agreement. 16, 17, 25, 47

SPOC Single Point Of Contact. 17

Bibliography

Journal Articles

[1] T. M. H. Reenskaug, “The original MVC reports”, 1979, Accepted: 2013-03-12T07:58:06Z. [Online]. Available: https://www.duo.uio.no/handle/10852/9621 (visited on 06/15/2020).

[2] IEEE, “IEEE guide for software requirements specifications”, *IEEE Std 830-1984*, pp. 1–26, Feb. 1984, Conference Name: IEEE Std 830-1984. DOI: 10.1109/IEEESTD.1984.119205.

[3] P. Damaschke, “Online strategies for backups”, *Theoretical Computer Science*, vol. 285, pp. 43–53, Jan. 1, 2000. DOI: 10.1016/S0304-3975(01)00289-4.

[5] A. A. Hasib and A. A. M. M. Haque, “A comparative study of the performance and security issues of AES and RSA cryptography”, *2008 Third International Conference on Convergence and Hybrid Information Technology*, 2008. DOI: 10.1109/ICCIT.2008.179.

[13] Mega Ltd., “MEGA security white paper”, no. 2, p. 42, Jan. 2020. [Online]. Available: https://mega.nz/SecurityWhitepaper.pdf (visited on 05/06/2020).

[26] S. S. Keller and T. A. Hall, “The XTS-AES validation system (XTSVS)”, p. 10,

Books

[9] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, “Redundant arrays of inexpensive disks(RAIDs)”, in *Operating Systems: Three Easy Pieces*, CreateSpace Independent Publishing Platform, Sep. 2018, ISBN: 978-1-985086-59-3.

Press Articles

[6] Donn Le Vie, Jr. (Aug. 29, 2010). Writing software requirements specifications (SRS), TechWhirl. Library Catalog: techwhirl.com Section: Tech Comm Deliverables, [Online]. Available: https://techwhirl.com/writing-software-requirements-specifications/ (visited on 04/05/2020).

[24] R. Hodge. (). Welcome to the 2019 data breach hall of shame, CNET. Library Catalog: www.cnet.com, [Online]. Available: https://www.cnet.com/news/2019-data-breach-hall-of-shame-these-were-the-biggest-data-breaches-of-the-year/ (visited on 06/02/2020).

Manuals/Guides

[18] (). Dd(1) [mojave man page]. Library Catalog: www.unix.com, [Online]. Available: /man-page/mojave/1/dd (visited on 05/05/2020).

[39] (). Systemd-halt.service, [Online]. Available: https://www.freedesktop.org/software/systemd/man/systemd-halt.service.html (visited on 06/08/2020).

[43] The Qt Company. (). Getting started programming with qt widgets — qt widgets 5.15.0, [Online]. Available: https://doc.qt.io/qt-5/qtwidgets-tutorials-notepad-example.html (visited on 06/15/2020).

Websites/Blogs

[8] Oxagile. (Feb. 5, 2014). Waterfall software development model, Oxagile Blog. Library Catalog: www.oxagile.com, [Online]. Available: https://www.oxagile.com/article/the-waterfall-model/ (visited on 06/13/2020).

[10] NetMarketShare. (2019). Desktop operating system market share in 2019, [Online]. Available: https://www.netmarketshare.com/operating-system-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Desktop%2Flaptop%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Custom%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22platform%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22platformsDesktop%22%2C%22dateInterval%22%

3A%22Monthly%22%2C%22dateStart%22%3A%222019-01%22%2C%22dateEnd%22%3A%222019-12%22%2C%22hiddenSeries%22%3A%7B%7D%2C%22segments%22%3A%22-1000%22%7D (visited on 05/25/2020).

[11] Stack Overflow. (2019). Stack overflow developer survey 2019, Developers' Primary Operating Systems 2019. Library Catalog: insights.stackoverflow.com, [Online]. Available: https://insights.stackoverflow.com/survey/2019/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2019#technology-_-developers-primary-operating-systems (visited on 05/25/2020).

[12] Docker. (Jun. 11, 2020). Overview of docker compose, Docker Documentation. Library Catalog: docs.docker.com, [Online]. Available: https://docs.docker.com/compose/ (visited on 06/14/2020).

[14] J. Anderson. (). FreeDNS - free DNS - dynamic DNS - static DNS subdomain and domain hosting, [Online]. Available: https://freedns.afraid.org/ (visited on 06/14/2020).

[15] (). Clonezilla - about, [Online]. Available: https://clonezilla.org/ (visited on 05/05/2020).

[16] (). Cloud data archiving — long-term object storage — amazon s3 glacier, [Online]. Available: https://aws.amazon.com/glacier/ (visited on 05/05/2020).

[17] T. Q. Company. (). Qt — cross-platform software development for embedded & desktop. Library Catalog: www.qt.io, [Online]. Available: https://www.qt.io (visited on 05/05/2020).

[19] (). Debian – the universal operating system, [Online]. Available: https://www.debian.org/ (visited on 05/05/2020).

[20] (). Docker hub, [Online]. Available: https://hub.docker.com/ (visited on 05/05/2020).

[21] (). Empowering app development for developers — docker, [Online]. Available: https://www.docker.com/ (visited on 05/05/2020).

[22] (). Express - node.js web application framework. Library Catalog: expressjs.com, [Online]. Available: https://expressjs.com/ (visited on 05/05/2020).

[23] (). Firebase authentication — simple, free multi-platform sign-in. Library Catalog: firebase.google.com, [Online]. Available: https://firebase.google.com/products/auth (visited on 05/05/2020).

[25] Katalon. (). Katalon — simplify web, API, mobile, desktop automated tests, Katalon Solution. Library Catalog: www.katalon.com, [Online]. Available: `https://www.katalon.com/` (visited on 06/14/2020).

[27] (). Mdadm, [Online]. Available: `http://neil.brown.name/blog/mdadm` (visited on 05/05/2020).

[28] Node.js. (). Node.js — about, Node.js. Library Catalog: nodejs.org, [Online]. Available: `https://nodejs.org/en/about/` (visited on 05/05/2020).

[29] (). ODIN - open disk imager for windows, [Online]. Available: `http://odin-win.sourceforge.net/` (visited on 05/05/2020).

[30] OpenSSL Software Foundation. (). OpenSSL - cryptography and SSL/TLS toolkit, [Online]. Available: `https://www.openssl.org/` (visited on 06/14/2020).

[31] (). OSFMount - mount disk images & create RAM drives, [Online]. Available: `https://www.osforensics.com/tools/mount-disk-images.html` (visited on 05/05/2020).

[32] (). Partclone - about, [Online]. Available: `https://partclone.org/` (visited on 05/05/2020).

[33] Proton Technologies AG. (). Secure email: ProtonMail is free encrypted email., ProtonMail. Library Catalog: protonmail.com, [Online]. Available: `https://protonmail.com` (visited on 06/14/2020).

[34] (). Qt cryptographic architecture, [Online]. Available: `https://api.kde.org/kdesupport-api/qca-apidocs/` (visited on 05/05/2020).

[35] (). Rclone - rsync for cloud storage, [Online]. Available: `https://rclone.org/` (visited on 05/05/2020).

[36] Selenium. (). SeleniumHQ browser automation, [Online]. Available: `https://www.selenium.dev/` (visited on 06/14/2020).

[37] Smartbear. (). Automated UI testing tools — TestComplete, [Online]. Available: `https://smartbear.com/product/testcomplete/overview/` (visited on 06/14/2020).

[38] SQLite Consortium. (). SQLite home page, [Online]. Available: `https://www.sqlite.org/index.html` (visited on 06/15/2020).

[40] The Botan Developers. (). Botan: Crypto and TLS for modern c++ — botan, [Online]. Available: `https://botan.randombit.net/` (visited on 06/14/2020).

[41] The HAProxy Developers. (). HAProxy - the reliable, high performance TCP/HTTP load balancer, [Online]. Available: http://www.haproxy.org/ (visited on 06/14/2020).

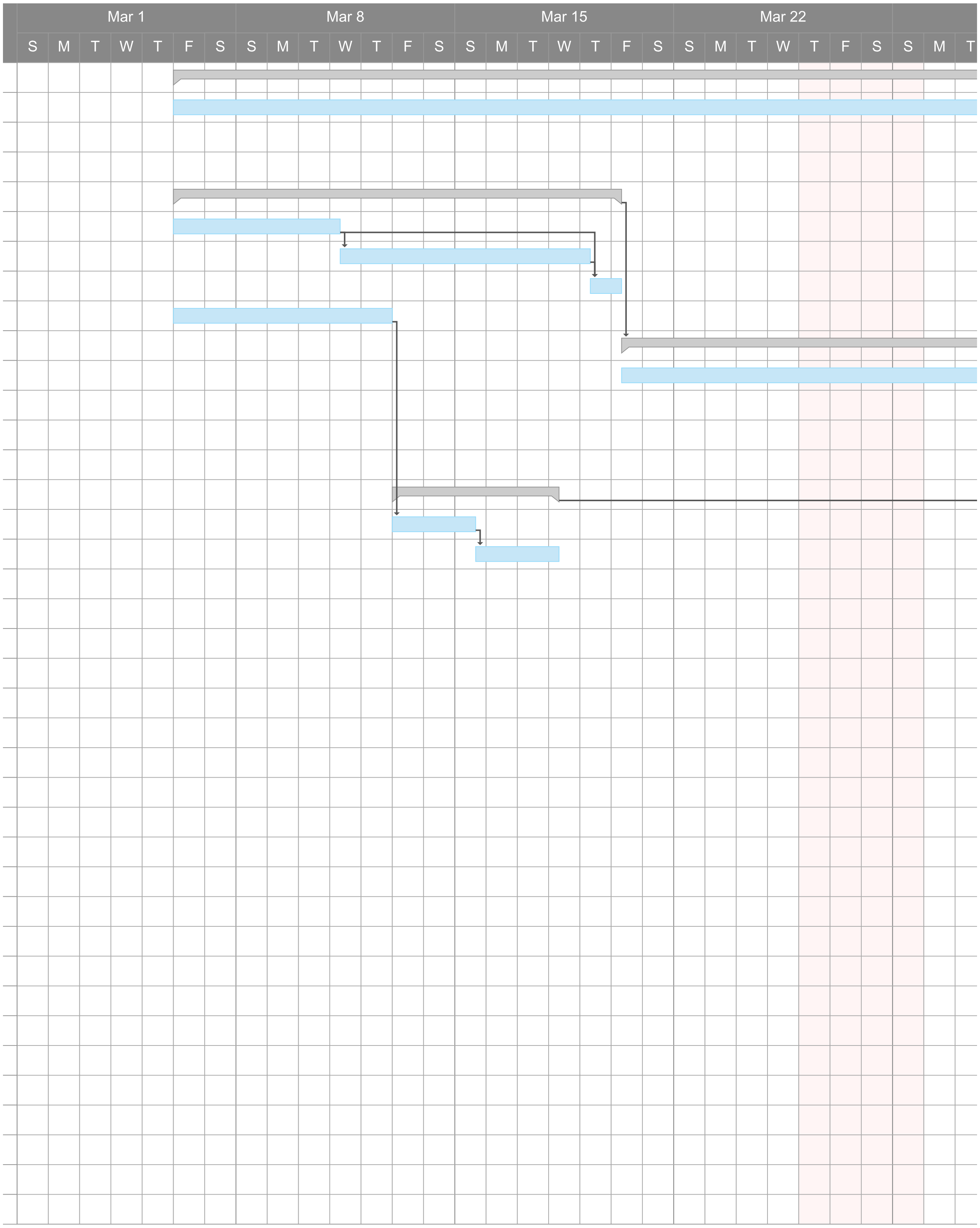
[42] (). The most popular database for modern apps, MongoDB. Library Catalog: www.mongodb.com, [Online]. Available: https://www.mongodb.com (visited on 05/05/2020).

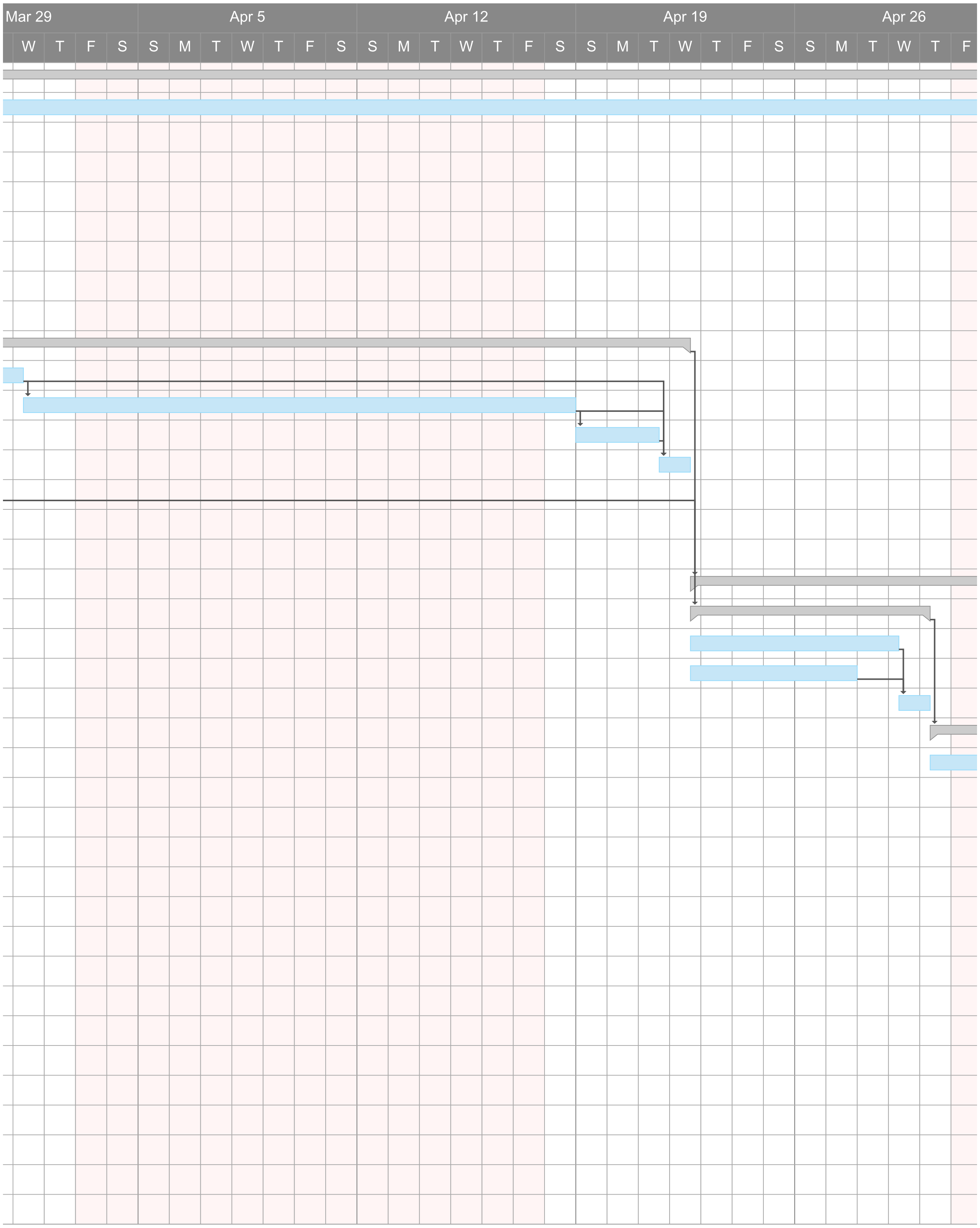
Appendix A

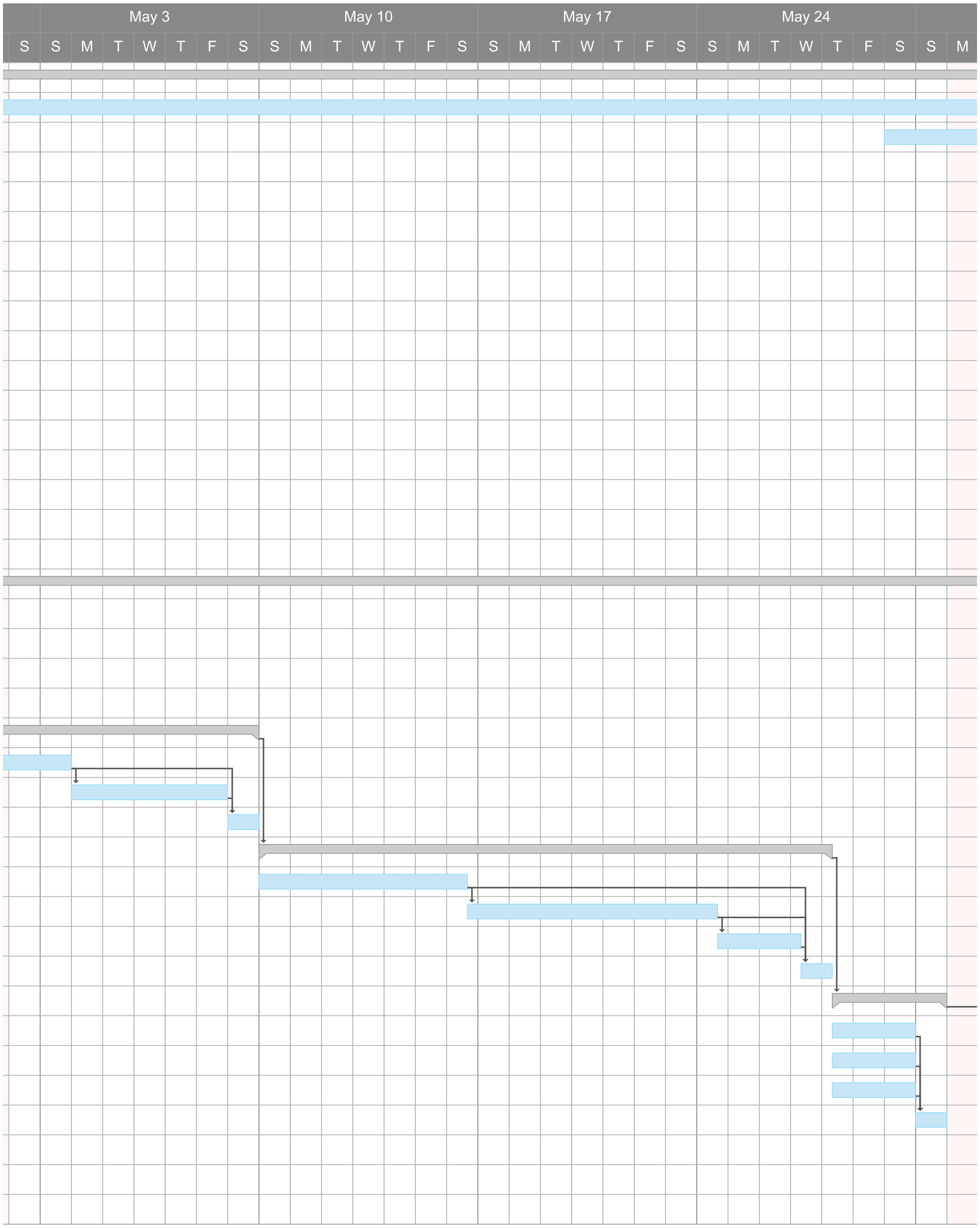
Annex: Gantt Chart

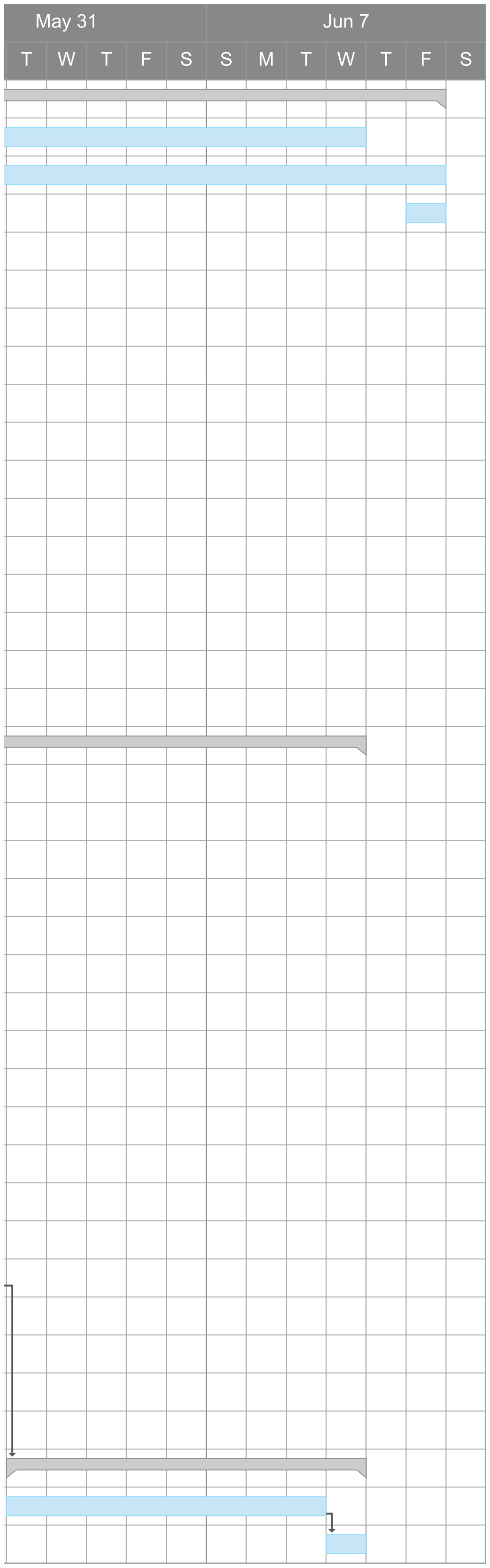
Starts at the next page

	Task Name	Duration	Start Date	Due Date	Predecessors
1	Documentation	78d	03/06/20	06/12/20	
2	Project Report (Memòria)	76d	03/06/20	06/10/20	
3	Project Presentation	13d	05/30/20	06/12/20	
4	Self Evaluation	1d	06/12/20	06/12/20	
5	Requirements Engineering	14.333d	03/06/20	03/20/20	
6	Formal description of requirements	16h	03/06/20	03/11/20	
7	Use Cases description	24h	03/11/20	03/19/20	6
8	Professor Feedback & Changes	3h	03/19/20	03/20/20	6, 7
9	Equipment provisioning	7d	03/06/20	03/12/20	
10	Software Specification	14.333d	03/20/20	04/22/20	5
11	User Flow Diagrams	24h	03/20/20	04/01/20	
12	Class Model Specification	8h	04/01/20	04/18/20	11
13	Data Model Specification	8h	04/19/20	04/21/20	12
14	Professor Feedback & Changes	3h	04/21/20	04/22/20	11, 12, 13
15	Hardware Setup	5.333d	03/13/20	03/18/20	
16	Raspberry Pi Setup	8h	03/13/20	03/15/20	9
17	RAID Array Setup	8h	03/15/20	03/18/20	16
18	Software Development	47.333d	04/22/20	06/10/20	10
19	Server Daemon	7.667d	04/22/20	04/30/20	15
20	NAS Runner	20h	04/22/20	04/29/20	
21	User Authentication	16h	04/22/20	04/27/20	
22	Professor Feedback & Changes	3h	04/29/20	04/30/20	20, 21
23	Server API	8.667d	04/30/20	05/09/20	19
24	AWS Integration	8h	04/30/20	05/03/20	
25	API Endpoints	15h	05/04/20	05/08/20	24
26	Professor Feedback & Changes	3h	05/09/20	05/09/20	24, 25
27	Client API	18.333d	05/10/20	05/28/20	23
28	Partition Image Creation	20h	05/10/20	05/16/20	
29	Image Encryption	24h	05/16/20	05/24/20	28
30	Cryptographic Keys Handling	8h	05/24/20	05/27/20	29
31	Professor Feedback & Changes	3h	05/27/20	05/28/20	28, 29, 30
32	CLI Client	3.667d	05/28/20	05/31/20	27
33	Linux	8h	05/28/20	05/30/20	
34	macOS	8h	05/28/20	05/30/20	
35	Windows	8h	05/28/20	05/30/20	
36	Professor Feedback & Changes	3h	05/31/20	05/31/20	33, 34, 35
37	GUI Client	9d	06/02/20	06/10/20	32
38	Linux	24h	06/02/20	06/09/20	
39	Professor Feedback & Changes	3h	06/10/20	06/10/20	38


Mar 1
Mar 8
Mar 15
Mar 22
S
M
T
W
T
F
S


Mar 29
Apr 5
Apr 12
Apr 19
Apr 26
W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F


May 3
May 10
May 17
May 24
S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M


May 31
Jun 7
T
W
T
F
S
S
M
T
W
T
F
S

Appendix B

Annex: Class Hierarchy in PlantUML format

B.1 Client Application

```
@startuml client
scale 1024 width
scale 768 height

class Account {
    + id : String
    + usedQuota : Integer
    + quota : Integer
    + createdAt : Integer
    + encryptedMasterKey : String
    + uploadBackup(file : PartitionImage)
    + downloadBackup(id : String )
}

Account "1" *-- "1..*" User

abstract class User {
    + email : String
    + createdAt : Integer
    + usedQuota : Integer
    - apiKey : String
    + getMasterEncryptionKey(masterPassword : String) : String
}

User <|-- Admin
User <|-- ManagedUser
Admin "1" o- "*" ManagedUser

enum WeekDay {
    Monday
    Tuesday
    Wednesday
    Thursday
    Friday
    Saturday
    Sunday
}

class UserSettings {
    + partitionsToBackupList[0..*] : String
    + backupDays[0..*] : WeekDay
    + backupTimes[0..*] : Integer
    + fullBackupInterval : Integer
}

User "1" o- "1" UserSettings

abstract class Partition {
    + id : String
    + label : String
    + rootPath : String
    + size : Integer
    + mount(adminPass : String, mountPoint : String) : Int
    + unmount(adminPass : String) : Int
    + createImage(adminPass : String, path : String) : PartitionImage
}

class NtfsPartition {
    + winLetter : String
}

class ExtPartition {
    + device : String
}

Partition <|-- NtfsPartition
Partition <|-- ExtPartition
Partition <|-- ApfsPartition

User "1" <- "*" Partition

abstract class PartitionImage {
    + id : String
    + parentId : String
    + label : String
    + path : String
    + fileExtension : String
    + size : Integer
    + isEncrypted : Boolean
    + encryptedAesKey : String
    + aesKeyEncryptionIv : String
    + encryptedInitVector : String
    + initVectorEncryptionIv : String
    + createdAt : Integer
    + encrypt(aesKey : String) : void
    + unencrypt(aesKey : String) : void
}
```

```
class DifferentialImage {
    + parentId : String
    + buildFromParent() : PartitionImage
}

Partition "1" o- "*" PartitionImage
PartitionImage <|-- ParentImage
PartitionImage <|-- DifferentialImage
ParentImage "1" o- "*" DifferentialImage

hide empty members
@enduml
```

B.2 Web Service Application

```
@startuml server
scale 1024 width
scale 768 height

class Account {
    + id : String
    + usedQuota : Integer
    + quota : Integer
    + encryptedMasterKey : String
    + createdAt : Integer
}

Account "1" *-- "1..*" User

abstract class User {
    + email : String
    + usedQuota : Integer
    + 3rdPartyAuthProvider: String
    + 3rdPartyAuthUserKey: String
    + clientApiKeys[0..*] : String
}

User <|-- Admin
User <|-- ManagedUser
Admin "1" o- "*" ManagedUser

!define datatype(x) class x << (D,#FF7700) DataType>>
hide empty members
datatype(3rdPartyUploadDetails) {
  uploadId : String
  containerBucketId : String
  fileObjectId : String
}

class File {
    + id : String
    + path : String
    + fileExtension : String
    + size : Integer
    + encryptedAesKey : String
    + encryptedInitVector : String
    + 3rdPartyStorageProvider : String
    + 3rdPartyStorageUniqueId : String
    + createdAt : Integer
    + uploadedAt : Integer
    + uploadedTo3rdPartyAt : Integer
    + uploadTo3rdParty() : 3rdPartyUploadDetails
    + downloadFrom3rdParty() : void
}

User "1" <- "*" File

class DifferentialFile {
    parentId : String
```

```
}

File <|-- DifferentialFile
File "1" o-- "*" DifferentialFile

hide empty members
@enduml
```