Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

AUTOPSY – ENHANCED DISTRIBUTED FORENSIC ANALYSIS

PEDRO HENRIQUE GASPAR CORDEIRO FERREIRA

Leiria, 25 de Fevereiro de 2020

[March 25, 2020 at 11:20 – version 1.0]

Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

AUTOPSY – ENHANCED DISTRIBUTED FORENSIC ANALYSIS

PEDRO HENRIQUE GASPAR CORDEIRO FERREIRA
Número: 2180078

Relatório de estágio realizado sob orientação da Professora Doutora Marisa da Silva Maximiano (marisa.maximiano@ipleiria.pt).

Leiria, 25 de Fevereiro de 2020

[March 25, 2020 at 11:20 – version 1.0]

ACKNOWLEDGEMENTS

I would like to express appreciation for the opportunity provided by "VOID SOFTWARE, S.A." to work in an enterprise environment. I'd also like to thank my faculty advisor Marisa da Silva Maximiano who provided much needed help writing this report. I'm also thankful that Antonio Branco took time off his work to help me with so many minor details in my CSS layouts, João Sousa who assisted me in every step along the development, as well as everyone else in the company who provided me with everything I needed to succeed in my internship.

RESUMO

TODO.

ABSTRACT

TODO.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API Application Programming Interface.

ASCII American Standard Code for Information Interchange.

BSD Berkeley Software Distribution.

CSS Cascading Style Sheets.

DOS Disk Operating System.

EXIF Exchangeable Image File Format.

FTK Forensics Toolkit.

FTP File Transfer Protocol.

GPT GUID Partition Table.

HTML HyperText Markup Language.

JSON JavaScript Object Notation.

JWT JSON Web Token.

NFTS New Technology File System.

NIST National Institute of Standards and Technology.

OS Operating System.

OTP One Time Password.

REST REpresentational State Transfer.

RSA Rivest-Shamir-Adleman.

STIX Structured Threat Information eXpression.

TSK The Sleuth Kit.

U2F Universal Second Factor.

UUID Universally Unique Identifier.

XML Extensible Markup Language.

1 INTRODUCTION

In the scope of the master's degree in Cybersecurity and Digital Forensics, students had the choice between writing a thesis, developing a project or doing an internship. This document is the report for a curricular internship conducted at VOID SOFTWARE, S.A., where the main goal of the internship was to develop a collaborative digital forensics platform, while also learning to develop software in an enterprise environment.

This chapter encompasses the motivation for this internship, a characterization of the host entity, the scope of the proposed project, and a summary of this document's structure.

1.1 MOTIVATION

Collaboration is a major feature included in most forensics software, and while Autopsy [1] allows the software to be configured in a manner to allow collaboration, it involves a complicated setup and has certain limitations. The setup required for collaborative cases in Autopsy is as follows:

- Shared hard drive accessible to every machine using the same drive letter
- PostgreSQL [2] Database server
- Apache Solr [3] indexing server
- Apache ActiveMQ [4] messaging server.

The limitations of Autopsy's multi-user case feature are that it requires every user to be using a Windows Operating System (OS) computer and it also requires specific configuration on each machine involved in the case that is set up.

The goal of this internship is to transform the existing Autopsy platform into a more complete digital forensics platform, including a client-server model that facilitates collaboration.

As a computer science student, my major interest has always been software engineering, and even after completing the first year of a master's degree in a more advanced subject, my interests remained unchanged. Based on that interest, the opportunity to work in an enterprise environment was captivating, as I could gain experience in the field while also developing an interesting project to complement my education.

1.2 HOST ENTITY

As a first step, an interview was conducted at VOID SOFTWARE, S.A., and a 9 month long curricular internship was planned. The initial project proposals were (1) the development of a client-server model for the existing Autopsy platform, or (2) the adaptation of the existing Autopsy platform for MacOS environments; The chosen proposal was the former.

1.2.1 *Company Characterization*

VOID is a privately held software development company established in Leiria, Portugal, in 2006, focused on building high-end products embodied in web, mobile and desktop applications, supported by creative software engineering tailored to each challenge's specific needs. It currently employs 30 high-end professionals in several fields of expertise.

VOID prides itself in providing very good conditions to its workers, making them feel like they are at home while working, and also to feel motivated to come to work every day. These conditions include the work environment itself, which is an open space where everyone can interact with each other, the "play areas" where people can relax while playing a game of pool or video games, and the rooftop terrace where workers can relax on sunny days. The company also makes sure nothing is missing to provide the best work environment possible by providing food and drink to all its workers at any time.

1.2.2 *Areas of Expertise*

VOID mostly functions as a company that develops software tailored to the specifications provided by the client, although it can also provide services in different areas like cybersecurity and digital forensics.

The company is capable of comfortably providing services in the following areas:

- Blockchain
- Machine learning and data science
- Augmented reality and virtual reality
- Mobile applications
- Web applications
- Desktop applications
- Cybersecurity and digital forensics.

Throughout its 14 years of being active in the software development industry, VOID has conducted some very interesting projects, as can be seen in Table 2.

Name	Description
Yes Account	A suite of applications for automated digitization of accounting documents
Web Portal	A large scale project for the European commission
Digital Archive	A digital preservation application
Dream Football	A social network along with web and mobile applications
Fuel Write	A comprehensive platform for fleet management, data collection and route optimization
Caspers	A mobile augmented reality customer experience and engagement
PBCore Toolkit	A desktop application to support the creation, editing, and export of moving image-related inventory metadata as PBCore XML records
Avenue Securities	A trading platform

Table 2: VOID's Main Projects

1.3 PROJECT SCOPE

The proposed solution consists in adapting Autopsy into a client-server model, and such adaptation can be categorized in the field of digital forensics. Even though the main focus is on typical software development, there are plenty of advanced concepts related to forensics science and cybersecurity that must be assimilated for the project to succeed.

The platform aims to cover three main aspects:

1. Accessibility
2. Collaboration
3. Organization.

1.3.1 *Accessibility*

Given a client-server architecture, any client with access to the network where the server is located can access the contents provided by the server. The aim is to condense the processing heavy features of Autopsy in a single server and provide any number of clients access to this information, requiring less resources from each client, allowing collaboration and removing any type of setup required for each of the client machines, while also providing a more modern and user friendly design.

1.3.2 *Collaboration*

In order to provide collaboration, all the information is maintained in a single server, or a collection of servers providing different functions (like exposing endpoints, storing data and indexing searches), and every client can preform all the allowed actions whether they consist in consulting, generating, or removing information. Collaboration comes naturally with a client-server model, as the same server that provides the endpoints can also communicate with each client using WebSockets, maintaining information in a coordinated state along every connected client.

1.3.3 *Organization*

Digital forensics investigations are usually done by specialized organizations, that need to organize their human resources in an efficient and secure manner. Assigning investigators to teams, assigning teams to cases, allowing access to the platform and certain information is a critical part of the activities preformed by a company that specializes in digital forensics, so having these functionalities properly integrated into a digital forensics platform should be an important feature.

1.4 DOCUMENT STRUCTURE

This document contains five chapters: the first one provides an introduction about the internship that was carried out, the second contains important concepts to help better understand the contents of this document, the third focuses on the main goal of the internship which is the development of the proposed platform, the fourth presents experiences from being involved in different projects inside the company, and finally, the conclusion about the performed internship is presented in the fifth chapter.

2 BACKGROUND

The scope of this internship concerns digital forensics, as it focuses on adapting an existing forensics platform into a collaborative client-server model.

In this chapter, a contextualization of digital forensics is given, an analysis of both The Sleuth Kit (TSK) [5] and Autopsy is made, and it's given a brief description and analysis of the existing forensic platform alternatives.

2.1 DIGITAL FORENSICS

Forensic science [6] is the use of scientific methods or expertise to investigate crimes or examine evidence that might be presented in a court of law.

The definition of digital forensics is directly related to the definition of computer forensics, which is the collection, preservation, analysis, and presentation [7] of evidence stemming from digital sources for use in a legal matter using investigative processes, tools, and practices.

Digital forensics [8] is the application of computer technology to criminal cases where evidence includes items that are created by digital systems.

Digital forensics is the field of forensic science that is concerned with retrieving, storing and analysing electronic data that can be useful in criminal investigations. This includes information from computers, hard drives, mobile phones and other data storage devices.

Digital forensic investigators face challenges such as extracting data from damaged or destroyed devices, locating individual items of evidence among vast quantities of data and ensuring that their methods capture data reliably without altering it in any way.

Personal data should ultimately be attributable to an individual; however, making that attribution can be difficult due to the presence or absence of individualized user accounts, security to protect those user accounts, and the actual placement of a person at the same location and time when the data is created.

2.1.1 *Digital Evidence*

Digital evidence is any type of digital data with incriminating characteristics, which can result from any type of action preformed by a user, like transactions and recordings.

Nowadays it's virtually impossible not to leave a digital track behind, since most of us carry and use devices capable of connecting to the internet.

The explosion of social media sites has created a whole new area of electronic evidence. Most people today are willing to share all kinds of information through social media platforms.

In order for electronic data to become digital evidence, it must be stored and be recoverable by a forensic examiner. One of the great challenges is not whether digital evidence may exist, but where the evidence is stored, getting access to that storage, and finally, recovering and processing that digital evidence for relevance within a civil or criminal action.

The potential storage options for electronic evidence has shifted from being only contained locally to being either located locally or remotely in what is called "The Cloud" [9].

More and more everyday computing processes are moving to the Internet where companies offer software as a service [10]. Software as a service means that the customer no longer has to install software on their computer, allowing access to the software remotely, and not storing any data locally.

2.1.2 *Processes and Procedures*

Digital forensics is the application of forensic science to electronic evidence in a legal matter.

While there are many different subdisciplines and many types of devices, communication and storage methods available, the basic principles of digital forensics apply to all of them.

These principles encompass four areas:

1. Acquisition
2. Preservation

3. Analysis
4. Presentation.

Each of these areas includes specific forensic processes and procedures.

Acquisition

Acquisition is the process of collecting electronic data. Seizing a computer at a crime scene or taking custody of a smartphone in a civil suit are examples of device acquisition, but the data must be extracted from these devices using specific procedures that equate to making a copy of the storage devices, while following strict rules to ensure the integrity of all the extracted data.

Since acquisition is the first interaction between the investigators and the evidence, it is the step where it's most likely to occur modifications of the contents of the seized devices, because turning on the device or extracting the data without following the right procedures can alter its contents irreversibly.

Preservation

For evidence to be defendable in court, it must be preserved properly. Preservation in the forensics context is the process of creating a chain of custody [11] that begins before collecting the evidence and ends when the evidence is released. Any interference in the chain of custody can lead to issues regarding the validity of the evidence. Additionally, preservation includes maintaining the evidence in a safe environment, preventing intentional destruction with malicious purpose or accidental modification by unqualified people.

A chain of custody log allows proving that the integrity of the evidence has been maintained from seizure through presentation in court. It should contain entries for every time that a piece of evidence has been touched, including collection, storage transport and any time the evidence is checked out for handling by any personnel.

Analysis

Analysis is the process of locating and categorizing items from evidence that has been collected in a case. Each case is unique as the circumstances surrounding each case can vary immensely, not only in the evidence being analysed, but also in the approach used to perform the analysis. The analysis is the area where the individual skills, tools used, and the training of the forensic examiner have the greatest impact on the outcome of the examination. Considering that electronic evidence appears in so many forms and comes from diverse locations and devices, the training and experience of the examiner has a much greater impact on the results of the examination.

Analysis of digital evidence is more than just determining whether a file exists on a hard drive, it involves finding out how that file got on the hard drive, and if possible, who put the file on the hard drive.

Presentation

Presentation of the examiner's findings is the last step in the process of forensic analysis of electronic evidence. This includes not only the written findings or forensic report, but also the creation of sworn statements, depositions of experts, and court testimony. There are no exact rules or standards for reporting the results of an examination. Each entity may have its own particular guidelines for reporting. However, forensic examination reports should be written clearly, concisely, and accurately, explaining what was examined, the tools used for the examination, the procedures used by the examiner and the results of the examination. The report should also include the collection methods used, including specific steps taken to protect and preserve the original evidence and how the verification of the evidence was performed.

2.2 THE SLEUTH KIT

TSK is a library and collection of command line tools that allow the investigation of disk images. The core functionality of TSK allows volume and file system data analysis. The plug-in framework allows incorporation of additional modules to analyse file contents and build automated systems. The library can be incorporated into larger digital forensics tools and the command line tools can be directly used to find evidence.

The original part of TSK is a C library and collection of command line file and volume system forensic analysis tools. The file system tools allow examining file systems of a computer in a non-intrusive fashion. Because the tools do not rely on the operating system to process the file systems, deleted and hidden content can be shown.

The volume system tools allow examination of the layout of disks and other media. TSK supports DOS partitions, BSD partitions, Mac partitions, Sun slices, and GPT disks. With these tools, partition locations can be identified and extracted so that they can be analysed with file system analysis tools.

When performing a complete analysis of a system, command line tools can become tedious. Autopsy is a graphical interface to the tools in TSK, which allows easier conduction of an investigation. Autopsy provides case management, image integrity, keyword searching and other automated operations.

A complete analysis also requires more than just file and volume system analysis. However, a single tool can't provide support for all file types and analysis techniques. The TSK Framework allows tools to easily incorporate file analysis modules that were written by other developers.

TSK allows listing allocated and deleted ASCII and Unicode file names, can display the details and contents of all NFTS attributes, can display file system and meta-data structure details, can create time lines of file activity, which can be imported into a spread sheet to create graphs and reports. TSK allows the lookup of file hashes in hash databases, it organizes files based on their type, and pages of thumbnails can be made from graphic images to facilitate quick analysis.

2.3 AUTOPSY

Autopsy is a digital forensics platform and a graphical interface to TSK along with other digital forensics tools. It is used by law enforcement, military, and corporate examiners to investigate what happened on a computer. It can even be used by anyone to recover photos from a camera's memory card.

Autopsy was designed to be intuitive out of the box. Installation is easy and wizards guide the user through every step. All results are shown in a single tree, as can be seen in Figure 1.

Autopsy was designed to be an end-to-end platform with modules that come with it out of the box and others that are available from third-parties. An overview of Autopsy's modules can be found in Table 4.

Autopsy runs background tasks in parallel using multiple cores and provides results as soon as they are found.

Autopsy is free and open source, allowing for cost-effective digital forensics analysis and community contributions.

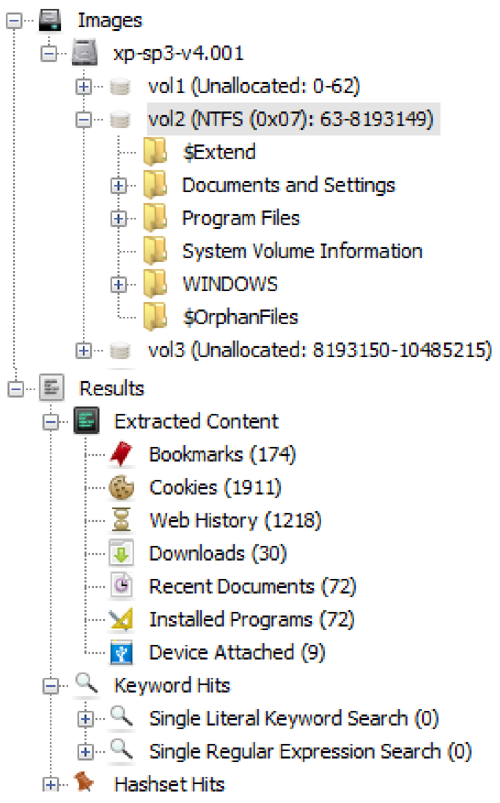


Figure 1: Autopsy's Explorer Tree

Name	Description
Timeline Analysis	Advanced graphical event viewing interface
Hash Filtering	Flag known bad files and ignore known good
Keyword Search	Indexed keyword search to find files that mention relevant terms
Web Artifacts	Extract history, bookmarks, and cookies from Firefox, Chrome, Internet Explorer and Microsoft Edge
Data Carving	Recover deleted files from unallocated space
Multimedia	Extract EXIF metadata from pictures and videos and display these files
Indicators of Compromise	Scan a computer using STIX

Table 4: Autopsy's Modules

2.4 RELATED SOFTWARE

There are plenty of alternatives to Autopsy in the form of digital forensic analysis platforms, but the ones with most recognition work on a software as a service model, without providing a free trial, so even though it wasn't possible to test them, they were analysed based on the available information.

2.4.1 *Nuix Lab*

Nuix Lab [12] allows investigators to work on large investigations. It can be used for local or small regional forensic labs handling high data volume, variety, and complex digital evidence and looking to build or upgrade a dedicated digital forensics facility.

The core technologies of the Nuix Lab, Nuix Workstation and Nuix Investigate, give digital forensic technicians and case investigators different viewpoints into the same case data. Investigators can collaborate on the same data at the same time by using browser based tools.

The Implementation of Elasticsearch [13] as a data store for the Nuix Lab boosts evidence processing, investigation, and intelligence capabilities.

Nuix Lab also claims to contain powerful artificial intelligence, machine learning, and analytics which should make it stand out from the competition.

2.4.2 *EnCase Forensic*

EnCase Forensic [14] enables searching, identifying, and prioritizing potential evidence, in computers and mobile devices, to determine whether further investigation is warranted.

It can collect from a wide variety of operating and file systems, including over 25 types of mobile devices, and parses the most popular mobile apps across iOS, Android, and Blackberry devices. Has strong decryption capabilities, allowing identification and unlocking password-protected files, and contains a custom indexing engine.

EnCase Forensic provides a wide range of capabilities that enable performing deep forensic analysis as well as fast triage analysis from the same solution, and provides a flexible reporting framework that empowers tailoring case reports to meet specific needs.

2.4.3 *Forensics Toolkit*

Forensics Toolkit (FTK) [15] is an award-winning, court-cited digital investigations solution.

It locates evidence, collects and analyses any digital device or system producing, transmitting or storing data by using a single application for multiple devices.

All digital evidence is stored in one case database. It reduces the time, cost and complexity of creating multiple datasets, and there is continuous data transfer between AccessData's forensic and e-discovery solutions, allowing for collaboration between all parties working on the case.

FTK allows users to create images, process a wide range of data types, analyse the registry, crack passwords and build reports.

FTK allows collaboration, contains indexed searches, constructs timelines and other graphical assets, categorizes all the extracted artifacts and also contains malware identification modules.

2.5 COMPARISON

A comparison of the features offered by each software is presented in Table 6.

Feature	**Autopsy**	**Nuix**	**Encase**	**FTK**
Collaboration	✓	✓	✓	✓
Web Based Interface	✗	✓	✗	✗
Indexed Searches	✓	✓	✓	✓
Decryption	✗	✗	✓	✓
Artificial Intelligence	✗	✓	✓	✗
Malware Analysis	✗	✗	✗	✓
Report Generation	✓	✓	✓	✓
Free and Open Source	✓	✗	✗	✗
Modular Plugins	✓	✗	✗	✗

Table 6: Software Comparison

Every software listed has its pros and cons; Autopsy mainly benefits from being free and open source, and allowing the development of modular plugins, while the other options seem to try to offer a very complete package out of the box.

Any of these software options seem viable for any kind of digital forensics investigation, and as a digital forensics firm, the teams should first test all these available options and chose the ones that best suit their needs.

3 KENSENTME PLATFORM DELEVOPMENT

The main goal of the internship is the development of a collaborative platform based on Autopsy, which was soon named “Kensentme“ by the company’s designer.

Each step taken towards the development of this platform is documented in this chapter, from project planning to full roll out on a production environment.

3.1 DEVELOPMENT FRAMEWORK

The project development followed an agile workflow, using a JIRA [16] board to organize bi-weekly sprints.

The project had four intervening parties involved. The developer was responsible for developing the software. The product owner provided help in decisions surrounding the developed product. The designer came up with ideas for the user interface and user experience of the platform. And the tester made sure everything was working as intended.

For each sprint the process was identical: a sprint review was conducted to assess the progress made in the previous sprint, then a new sprint would be planned by the developer, the features would be implemented during the two week duration of the sprint, and developed features would be tested thoroughly before the sprint ended.

3.2 PROJECT AIM AND MILESTONES

Autopsy by itself is capable of providing a distributed solution for multi user collaboration, but it is very resource intensive, requires many complex configuration steps and is also only fully supported on the Windows OS.

The plan for this project is to achieve the same kind of functionality provided by the original software, but without the dependency on arduous pre-configuration or hardware intensive requirements, resulting in needing only single capable server, and allowing the program to be used by multiple low capacity client devices using any web capable OS.

To achieve that goal, the project is an adaptation of the original Autopsy source code, into a client-server model, with the server developed in Java using the Quarkus [17] framework, and the client developed in JavaScript using the React [18] framework.

The project is outlined to work in a multi user environment, allowing users to be assigned to teams and teams assigned to cases, and allowing multiple users to interact with a case simultaneously.

Autopsy has a major limitation, which is it only allows one case to be open at a time; Ideally in this project a workaround should be created to allow working on multiple cases at once, but dedicating a server instance (which a single machine can contain many within) per case is a good enough approach, though as future work the ability to spawn different containers as requested to work on different cases at once is an interesting challenge.

Given that the core features will be running in a remote server, it was decided that the addition of data sources to cases will be handled by an FTP client, allowing users to transfer files to their respective data source directories, and FTP access will be controlled according to each user's credentials on the platform.

The architecture of the expected result is represented in Figure 2, where it can be seen that multiple clients can connect through a REST API, WebSockets and FTP with the Kensentme server, which requests and provides data to both database and indexation servers.

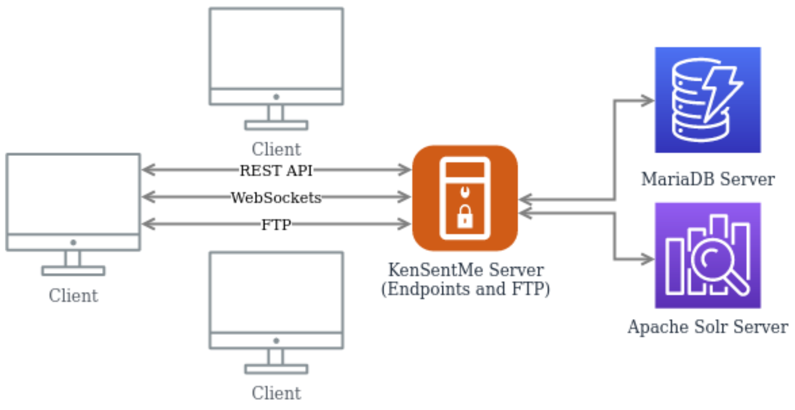


Figure 2: Software Architecture

The KenSentMe server described in the previous figure provides different services, such as authentication, case resources, entity management, and others, as can be seen in Figure 3.

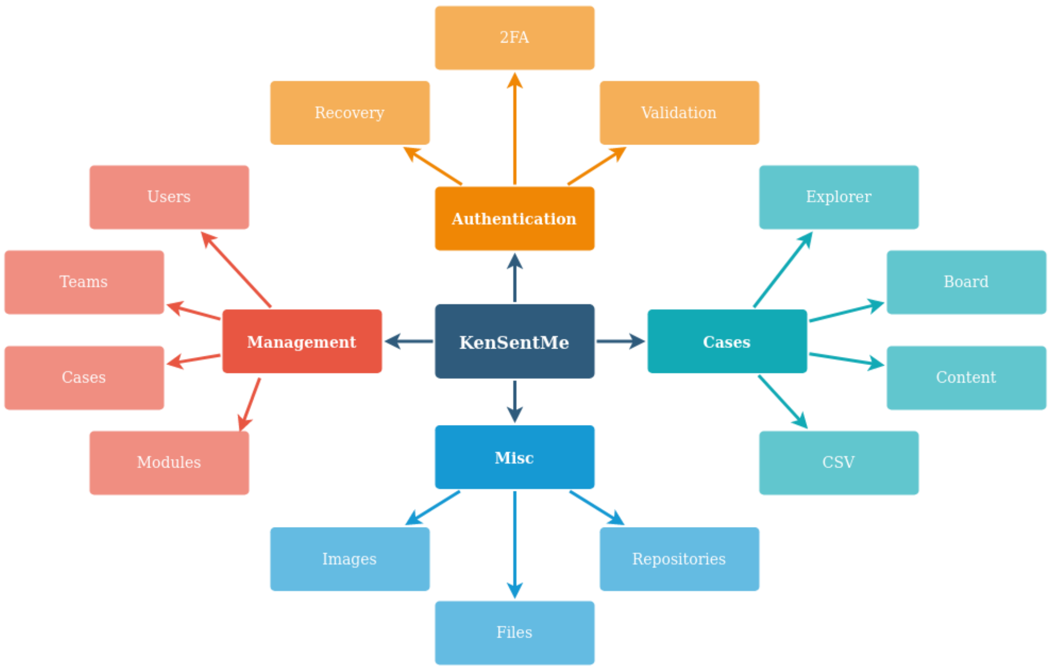


Figure 3: Platform's Services

3.3 AUTOPSY SOURCE CODE ANALYSIS

Autopsy is a digital forensics analysis software that is available as Open Source Software [19] on GitHub [20].

With the goals set for this project, the source code was analysed to understand which components need to be replicated and adapted in order to obtain the same logic flow.

The "Core" module is where the most important components are located, and after analysis it was concluded that the following directories present in Table 7 contain relevant information.

Name	Description
Actions	User interactions
Casemodule	Case class and other resources needed for the functioning of an autopsy case like data sources and artifacts
Centralrepository	Data persisted and accessed by multiple cases (Correlation Engine)
Contentviewers	Panels used for data representation
Coordinationservice	Configuration information distribution system
Core	Addition of command line options, system configurations and collaboration monitor
Corecomponents	Main user interface components
Datamodel	All the entities needed to represent ingested data
Datasourceprocessors	Data processing utilities
Directorytree	File explorer for ingested artifacts
Ingest	Utilities and events for data ingestion
Keywordsearchservice	Utility to search artifacts by keyword
Modules	All the pre-included modules (data ingestion procedures)
Progress	Progress indicators and similar classes
Python	Resources needed for the functioning of the Jython language
Rejview	Resources used to analyse Windows registry
Report	Report generation utilities and modules
Timeline	Recent addition to Autopsy, allows visualization of artifacts in temporal chart, only available for Windows OS

Table 7: Autopsy Modules Overview

The “KeywordSearch“ module is also of critical importance as it provides one of the most meaningful features, which is filtering all the artifacts in a case with a keyword search using the Apache Solr search platform, which indexes the text contents of all the artifacts and allows extremely fast searching through a large amount of data.

Another module that needs to be adapted is the "RecentActivity" module, which contains the tools needed to extract information from browsers, registry and other important resources, providing a great amount of critical evidence from data sources.

Each autopsy case contains its own SQLite [21] database, which contains the tables present in Table 8.

tsk_db_info	tsk_db_info_extended	tsk_objects
tsk_image_info	tsk_image_names	tsk_vs_info
tsk_vs_parts	tsk_fs_info	data_source_info
tsk_files	file_encoding_types	tsk_files_path
tsk_files_derived	tsk_files_derived_method	tag_names
review_statuses	blackboard_artifacts	blackboard_attributes
ingest_modules	blackboard_attribute_types	ingest_module_types
reports	blackboard_artifact_types	ingest_jobs
ingest_job_modules	ingest_job_status_types	account_types
accounts	account_relationships	tsk_event_types
tsk_examiners	blackboard_artifact_tags	content_tags
tsk_file_layout	tsk_event_descriptions	tsk_events
tsk_pool_info		

Table 8: Case Database Tables

The tables containing the information accessed most frequently are the ones related to blackboard artifacts, along with tables related to files and tags.

Autopsy's source code allows access to most of the important entities present in this database, through pre-defined queries which return entities like data sources, abstract files, or artifacts. But in order to add features like pagination and tag deletion, custom queries must be created, so that the queries may return only a specific amount of entity ids, or so that delete commands may be executed.

3.4 DEVELOPMENT STAGES

3.4.1 *Basic Autopsy Functionalities*

As a first step into replicating the functionalities of the original program, the most basic functionality from Autopsy was adapted, the ability to open an Autopsy case, as can be seen in Figure 4.

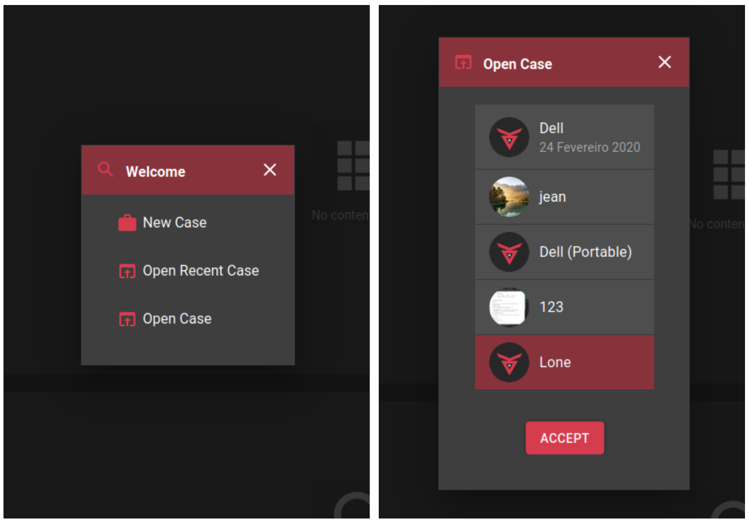


Figure 4: Welcome and Open Case Modals

For this, some elements of the original ”Casemodule” package were adapted, and after that all the other similar actions like closing, creating and deleting cases were also adapted.

Autopsy cases have a case file containing case metadata, which allows the program to connect to the right database when the case is open: this database is also present in a file inside the file system, which uses the SQLite database engine. So, for the cases to be usable in the server, these files must also be present in the server, which resulted in the creation of a directory within the server called “repository“, containing all the different cases created within the application.

Later in the development there was the need to create an additional directory alongside the “repository“ called “central-repository“, which contains the database used by the Correlation Engine, a feature that finds files present in multiple cases to ingest data that can be queried by any case.

The development of autopsy features was done with the original software as a reference, so that every new feature implemented into the platform could be compared and verified with the original software, and it was reassuring to see that cases created through one of the versions of the software were capable of being opened through the other version, meaning that the source code provided by autopsy, with some tinkering, can definitely be used in a client-server model.

3.4.2 *Authentication Process*

The authentication process is handled through a REST API, when the authentication is completed the user receives a JSON Web Token (JWT).

JWT is an open standard that defines a compact and self-contained way for securely transmitting information between parties. This information can be verified and trusted because it is digitally signed. JWT contain three parts, as can be seen in Table 10.

Name	**Contents**
Header	Information about the signing algorithm used
Payload	Information about the user, such as username, e-mail and roles
Signature	A signature used to verify the contents weren't changed

Table 10: JWT Composition

The authentication can be performed with either username or e-mail and a password, and extra authentication factors can be added such as One Time Password (OTP) and Universal Second Factor (U2F).

If no extra factors were added to an account, the authentication attempt with the user's credentials will return the JWT when successful and the user can freely access protected content.

When extra factors are present in an account, the response from the same request will contain an array with the extra factors added, and a UUID, which is a pseudo random string, which must be sent in the next request to validate the user's identity.

This UUID is saved in the database in the form of salted hash using the Blowfish [22] algorithm, and is invalidated after a successful login attempt. The user then has the choice between any of the extra factors and must complete the required steps to validate that factor, as can be seen in Figure 5.

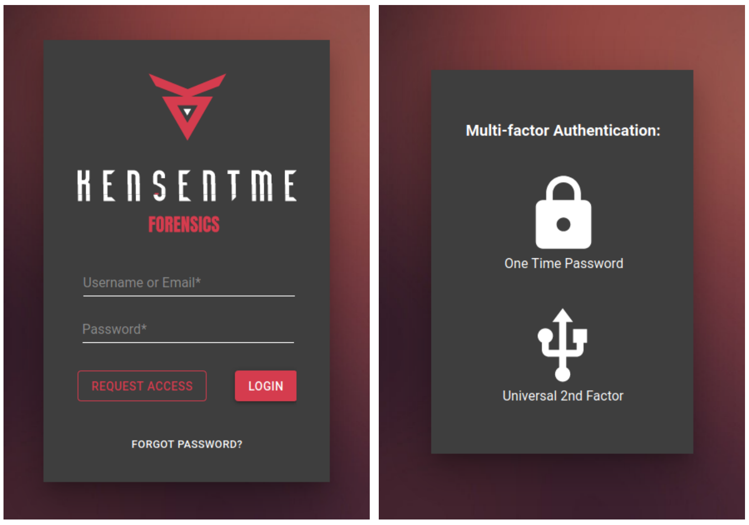


Figure 5: Login Screen and Extra Factors

One Time Password

The process of setting up OTP for an account begins with generating a secret string, which is used by both the server and the authenticator app to generate OTP. When the secret is generated, it is saved as a temporary secret after being encrypted by the RSA algorithm.

The temporary secret must be added to the user's device, either by inserting the string itself or by scanning a QR code [23], and must be validated by the user by submitting the current OTP; After validation the temporary secret is considered permanent.

All OTP validations have a 30 second threshold, which means the current OTP is valid for an extra 30 seconds after being replaced by the next OTP, allowing the user to have a better experience with these passwords.

The processes of authenticating and removing this authentication factor both depend on validating the current OTP, which is provided by the user's app.

In the event of loss of the device that the user uses to generate OTP, all the authentication factors can be reset using the ”Reset Password” functionality, which relies on e-mail validation to prove the user's identity.

Universal Second Factor

Both the set up and validation of U2F are very similar, as these procedures require the server to generate a challenge. This challenge is transmitted to the client and then transmitted to the U2F device, which then generates the response which is used to validate the challenge.

In the case of setting up the device, if the response is validated successfully, the information present in Table 12 is stored in the server and the procedure is completed.

Name	**Description**
Key Handle	Identification of the registration, allows the same key to be used for multiple accounts in the same domain
Public Key	Resulting from the generation of a key pair specific to the domain, used to verify the information provided by the device
Attestation Certificate	A certificate that can be used to verify the authenticity of the device
Counter	The device's authentication counter, used to prevent the use of cloned devices
Compromised	Flag indicating if the credential is considered to be compromised

Table 12: U2F Registration Composition

When authenticating, the challenge depends on the public key previously stored, which means only the device used to set up can generate the right response to the challenge.

The U2F procedure is represented in Figure 6.

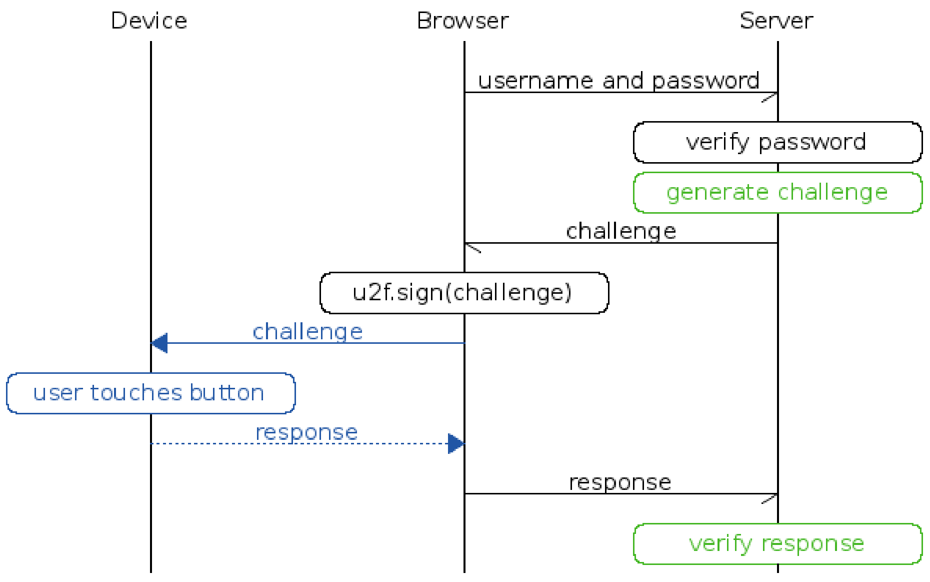


Figure 6: U2F Procedure [24]

3.4.3 *Management Entities*

Further in the development, the different persisted entities were created, which are Users, Teams, and Cases. All the endpoints for actions involving these entities were created, resulting in the ability for the client program to interact with these entities and modify their relationships and other variables.

For these functionalities there are two roles associated: (1) the Manager role allows manipulation of the existing entities, while (2) the Investigator role only has access to his own information and the teams and cases he was assigned to. For these interactions, it was decided to create a drag and drop interface, which allows users to be dragged into teams and teams dragged into cases. All these entities are listed side by side and each has its own options and filtering input, as can be observed in Figure 7.

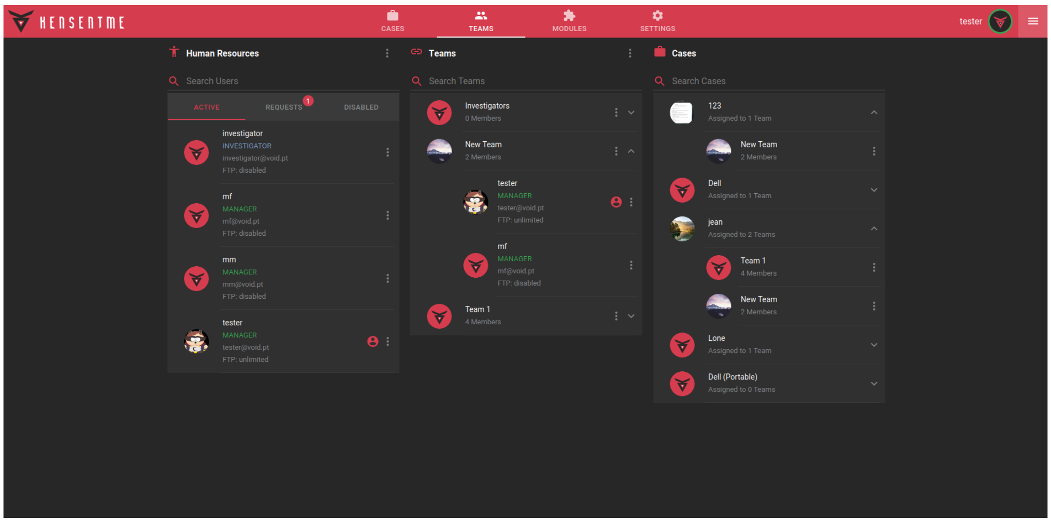


Figure 7: Entity Management Interface

Users can change their own profile picture, while Managers can also change any team or case's display picture.

Managers can add new users to the platform, can approve membership requests, can enable/disable user accounts and can create new teams.

When a user is added by a Manager or his membership request is approved, he must define a password when activating his account through a received e-mail message.

It was at this step in development that was made clear by the testers that validations must be preformed on both server and client side, and that error handling must be streamlined in order to provide a seamless experience to the user, while also allowing future additions to the platform to be implemented in a similar way without much tinkering.

Because even though the platform still wasn't fully developed, it already suffered from possible exploits that could break the experience for its users, like the upload of a 5GB file as an image file, which would stop the page from loading, or possible directory traversal attacks, resulting from the creation of case directories.

The lessons learned from the testers on this stage of development carried on through the entire development process and ensured that every new feature added was accounting for possible weaknesses which may result from the new implementations, resulting in a more secure development process.

The platform's resulting data model can be observed in Figure 8.

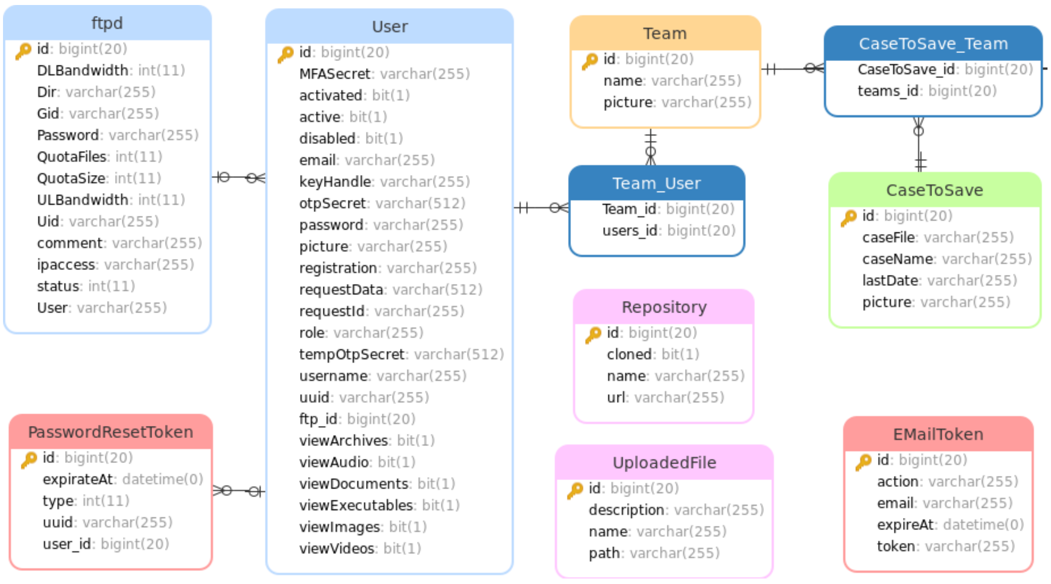


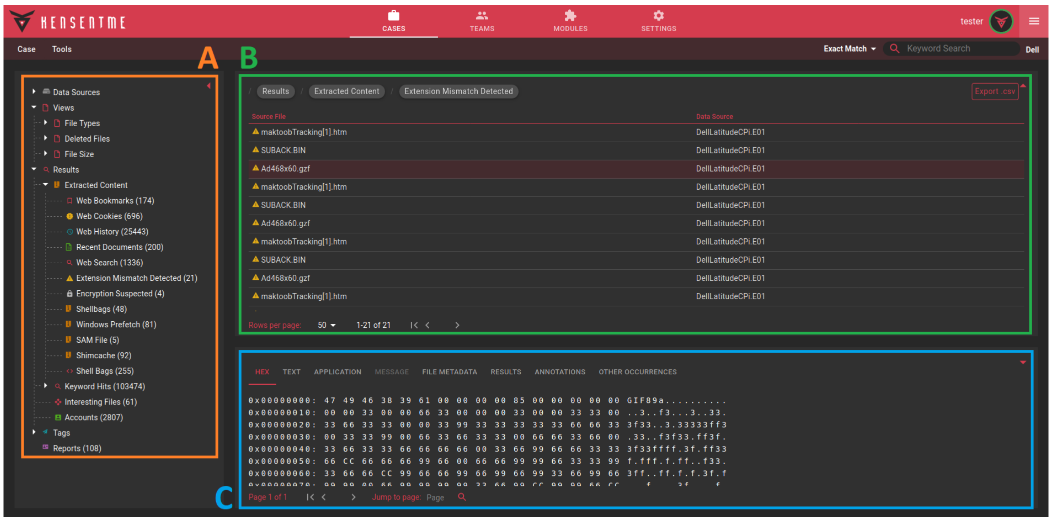
Figure 8: Database Structure

3.4.4 *Ingested Results Presentation*

Ingested results are the items present inside the provided data sources. Autopsy can run multiple modules on each data source to extract results, and the extracted results can either be a file instance or an artifact instance. The difference between a file and an artifact is that an artifact represents only a smaller piece of a file's information, which means a file can contain multiple artifacts, for example a registry file can contain multiple OS user account artifacts.

Even though the data ingestion features weren't implemented yet on the platform, the data that was ingested into a case using the original program could also be used on this platform, which allowed this feature to be developed earlier.

The ingested results are presented in three different containers, one taking the shape of a file explorer, allowing exploration of the structure of all the results, one taking the shape of a table or thumbnail viewer, that can be referenced as a board, presenting all the contents of the results selected from the explorer, and one taking the shape of a content viewer, allowing visualization of the data contained inside the result selected from the table as can be seen in Figure 9.



Source File	Data Source
maktoobTracking[1].htm	DellLatitudeCPi.E01
SUBACK.BIN	DellLatitudeCPi.E01
Ad468x60.gzf	DellLatitudeCPi.E01
maktoobTracking[1].htm	DellLatitudeCPi.E01
SUBACK.BIN	DellLatitudeCPi.E01
Ad468x60.gzf	DellLatitudeCPi.E01
maktoobTracking[1].htm	DellLatitudeCPi.E01
SUBACK.BIN	DellLatitudeCPi.E01
Ad468x60.gzf	DellLatitudeCPi.E01
maktoobTracking[1].htm	DellLatitudeCPi.E01

```
0x00000000: 47 49 46 38 39 61 00 00 00 00 85 00 00 00 00 00 GIF89a..........
0x00000010: 00 00 33 00 00 66 33 00 00 00 33 00 00 33 33 00 ..3..f3...3..33.
0x00000020: 33 66 33 33 00 00 33 99 33 33 33 33 33 66 66 33 3f33..3.33333ff3
0x00000030: 00 33 33 99 00 66 33 66 33 33 00 66 66 33 66 00 .33..f3f33.ff3f.
0x00000040: 33 66 33 33 66 66 66 66 00 33 66 99 66 66 33 33 3f33ffff.3f.ff33
0x00000050: 66 CC 66 66 66 99 66 00 66 66 99 99 66 33 33 99 f.fff.f.ff..f33.
0x00000060: 33 66 66 CC 99 66 66 99 66 99 66 99 33 66 99 66 3ff..ff.f.f.3f.f
```

Figure 9: Ingested Results Presentation (A) Explorer; (B) Board; (C) Content Viewer

The content viewer can display different kinds of information depending on the type of item selected, which can be some of the following:

- Text browser
- Media viewer
- Database browser
- Registry browser
- Key-value browser
- Table data viewer.

The layout for the ingested results presentation, and for all the case related actions, was based on the original Autopsy layout, so that each container can be re-sized as needed, allowing the user to focus on the information that is most important to him.

This development stage was one of the most intensive, as it required understanding Autopsy's data structure very well. The source code used here was rewritten almost completely, because of the way the user interface and the data structure in the original program are interconnected, resulting in the need to adapt and write custom queries to obtain the desired results (which include pagination), creating 65 endpoints so that all the needed information could be supplied, as well as creating and adjusting all the user interface elements to achieve an user experience similar to what Autopsy's users are used to.

3.4.5 *Tag Management*

Tagging content is a very important step in forensics investigations, as it allows the investigators to keep an organized list of important information.

Tagging was added to the platform in the form of a context menu, that is shown by selecting information from the table/thumbnail viewer, as can be seen in Figure 10.

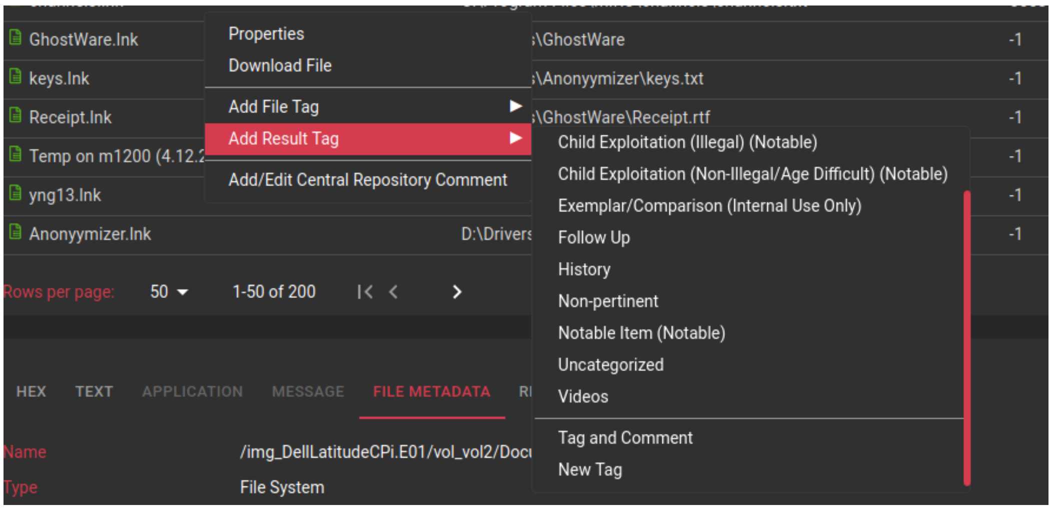


Figure 10: Tags Context Menu

Tags take the form of file or result tags, and the difference is that a result tag is specific to an artifact, while a file tag is related to the file itself.

Autopsy contains 10 different tag types by default, and more can be added, which can result in a high amount of tag types in use, which deteriorates the user experience.

Tag names are also shared between cases, so it seemed important to have more control over the management of these tags, and the ability to delete tag types was added, even though it didn't exist in the original software.

Since each case contains its own database, it's impossible to synchronize every case at once when a tag is deleted, so it was necessary to create a procedure to synchronize case tags when a case is opened.

3.4.6 *Data Sources*

Using the same credentials used to log-in to the platform, the user can also upload data source files into his folder located inside the server, using an FTP client like FileZilla [25]. Then the user can browse these directories using the web interface and select a data source to add to the case, as can be seen in Figure 11.

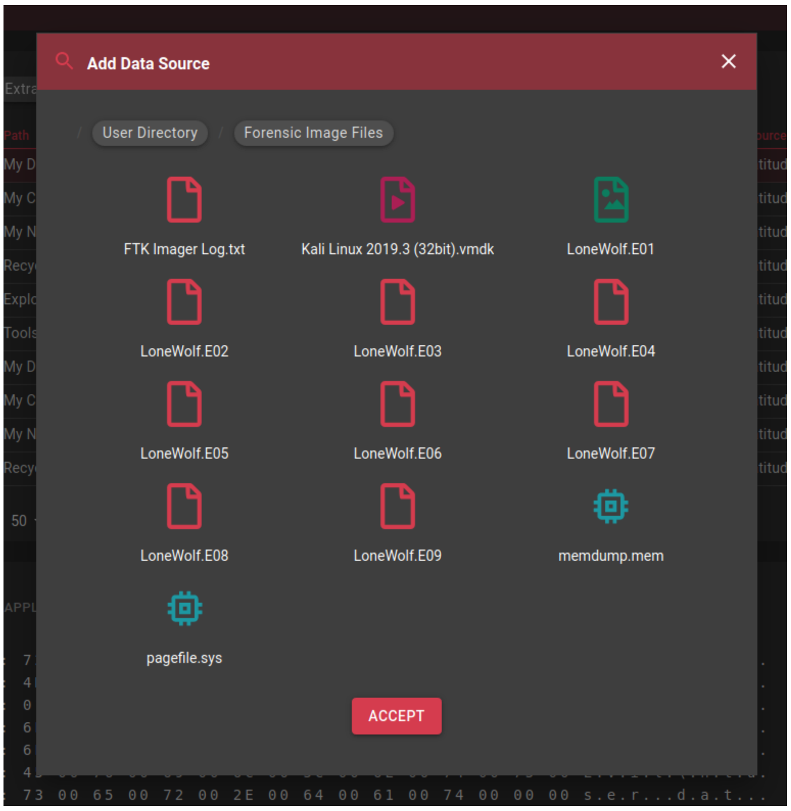


Figure 11: Data Source Selection

The procedure for adding a data source to a case was adapted from Autopsy's source code, and depends on the type of data source added, which can be one of the following:

- Disk image

- Virtual machine
- Logical file collection
- Autopsy Logical Imager [26] results.

Local disk data sources were also an option provided by Autopsy but since the local disks the software has access to belong to a server, this feature is undesired.

When a data source is added to a case, it is processed and instances of abstract files and other entities are created, which allow the user to navigate the data source's contents as soon as it is processed.

Just navigating the data source contents isn't very useful, as the contents are in a raw state, showing mostly the file structure of the source, and not even categorizing files by mime types, so running data ingestion modules is a critical step into allowing a deeper analysis of a data source, which is covered in the next section.

3.4.7 *Data Ingestion Modules*

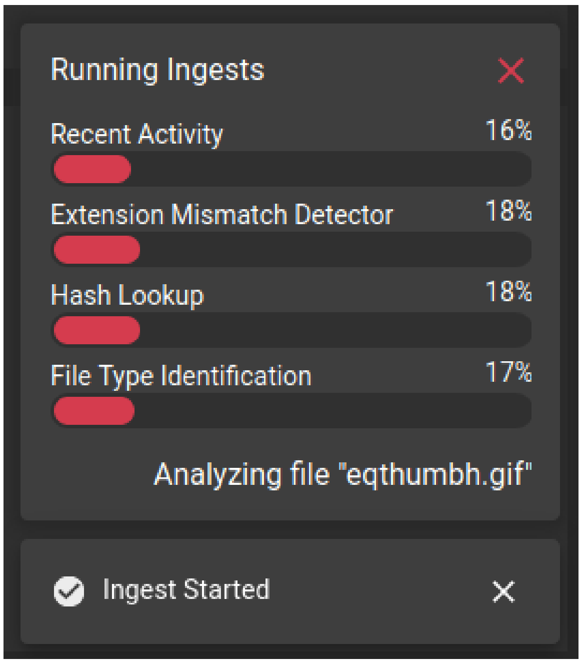
Data ingestion modules can be either file ingest modules, which analyse each file contained in a data source, one by one, or data source ingest modules, which run on specific components of a data source.

To run a collection of modules, the user selects which modules to run, and may also configure some parameters related to the modules. When the request is received by the server, the modules are started in a background thread.

When the modules are running, the server communicates to each client through WebSockets, offering feedback on the progress of the modules, as can be seen in Figure 12.

Originally it was intended to have the clients update their explorer as new information is ingested, but it would slow down the ingest progress considerably, so what was decided was that every client would refresh its explorer every 30 seconds while ingests are running.

There were other measures taken into consideration to improve the performance of the ingest procedure, while also maintaining the user informed of its progress: the server communicates a maximum of one file name per second, and it also only communicates a module's progress when it advances at least one percentage point.


Running Ingests

Recent Activity 16%

Extension Mismatch Detector 18%

Hash Lookup 18%

File Type Identification 17%

Analyzing file "eqthumbh.gif"

Ingest Started

Figure 12: Ingest Modules Communication

3.4.8 *Report Modules*

Report modules consist in generating a file or collection of files, to further process, filter, or present the results found in a case. They can take many shapes such as a text file, spreadsheets, HTML pages or even a portable case that can be imported into Autopsy.

Report modules have the ability to process existing data and generate results based on that, such as validating e-mail addresses or credit card numbers, which are found based on regular expressions by the keyword search module and may contain false positives.

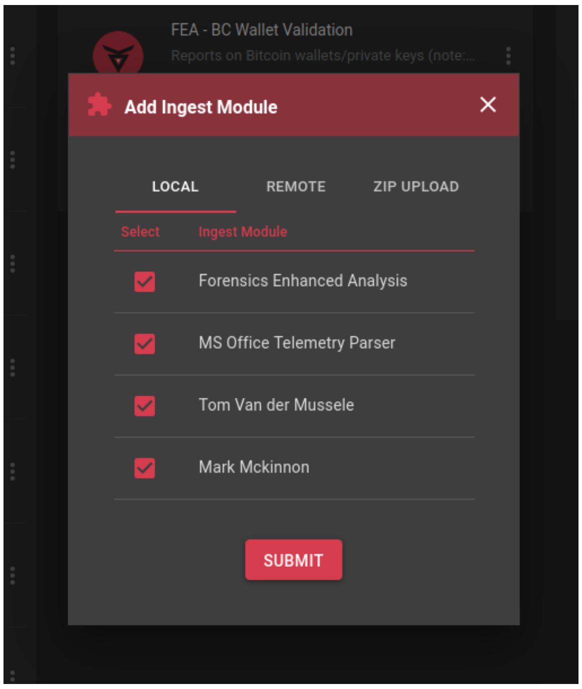
The adaptation of the source code for report modules went pretty smoothly, with the exception that some modules require the user to provide a file when running the module, which resulted in the creation of a file repository within the server, for files to be used in this manner.

3.4.9 *Add-on Modules*

Autopsy supports third party modules in the form or NetBeans [27] module files (NBM), and in the form of python scripts, using the Jython language, which allows python code to run alongside classes defined in Java.

Since the developed platform doesn't support a java user interface, the procedure to install Netbeans modules could not be implemented, as this feature is specific to Netbeans applications. But most modules are made in the Jython language, and the addition of this modules was implemented as intended.

A user with a manager role can add new modules to the platform by either selecting from defined repositories or by uploading a zip archive containing the modules, as can be seen in Figure 13.

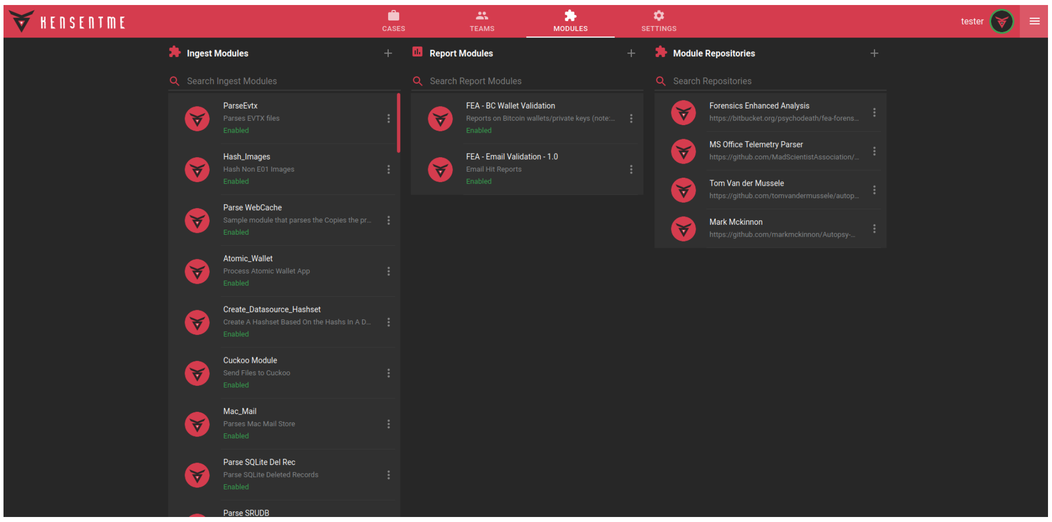

FEA - BC Wallet Validation
Reports on Bitcoin wallets/private keys (note:...
Add Ingest Module
LOCAL
REMOTE
ZIP UPLOAD
SUBMIT

Select	Ingest Module
✓	Forensics Enhanced Analysis
✓	MS Office Telemetry Parser
✓	Tom Van der Mussele
✓	Mark Mckinnon

Figure 13: Add-on Modules Addition

These repositories previously mentioned can be defined in the platform itself, or accessed from a remote source, which is a standalone platform developed just for this purpose and could allow the company to provide repository sources to any deployed version of the platform.

The platform also allows management of the added modules, so that managers can enable, disabled and remove certain modules, as can be seen in Figure 14.



Ingest Modules

Search Ingest Modules

ParseEvtx
Parses EVTX files
Enabled

Hash_Images
Hash Non E01 Images
Enabled

Parse WebCache
Sample module that parses the Copies the pr...
Enabled

Atomic_Wallet
Process Atomic Wallet App
Enabled

Create_Datasource_Hashset
Create A Hashset Based On the Hashs In A D...
Enabled

Cuckoo Module
Send Files to Cuckoo
Enabled

Mac_Mail
Parses Mac Mail Store
Enabled

Parse SQLite Del Rec
Parse SQLite Deleted Records
Enabled

Parse SRUDB

Report Modules

Search Report Modules

FEA - BC Wallet Validation
Reports on Bitcoin wallets/private keys (note:...
Enabled

FEA - Email Validation - 1.0
Email Hit Reports
Enabled

Module Repositories

Search Repositories

Forensics Enhanced Analysis
https://bitbucket.org/psychodeath/fea-forens...

MS Office Telemetry Parser
https://github.com/MadScientistAssociation/...

Tom Van der Mussele
https://github.com/tomvandermussele/autop...

Mark Mckinnon
https://github.com/markmckinnon/Autopsy-...

Figure 14: Add-on Modules Management

Third party modules require that a great deal of the original Autopsy source code is accessible in the original namespace, which resulted in the separation of the developed and adapted code into two different namespaces, the original software's namespace which is org.sleuthkit.autopsy, and the company's namespace which is pt.voidsoftware.kensentme.

This requirement of the original code can also mean that some modules which weren't tested may fail to run due to missing code, but a procedure to handle failure to run modules is implemented, and it informs the user about which modules are failing to run, allowing the user to disable the unsupported modules.

3.4.10 *Project Deployment and Documentation*

A "Read Me" document was created, detailing each step required to setup the platform, which involves installing all the dependencies required, as well as compiling the developed code.

The setup was tested in multiple versions of Linux and with different versions of needed dependencies, resulting in a controlled step-by-step guide that can be followed in about 30 minutes that leads to a successful deployment of the complete platform.

The documentation of every endpoint of the platform can be consulted in Appendix A.

3.5 CRITICAL ANALYSIS AND PROPOSED IMPROVEMENTS

The development of the platform halted due to it being considered in a finished state, which means that all the objectives for the project were completed successfully.

Given that Autopsy is still in active development, it would be very important to keep following its development and complementing this platform with new features and bug fixes which might come up in the future.

Besides accompanying Autopsy's improvements, it could also be interesting to further develop the platform to better serve the needs of its users, which would require installations of the platform and a feedback channel to better understand the user's needs.

4 CONCLUSIONS

TODO

BIBLIOGRAPHY

[1] T. S. Kit, *Autopsy*, Website, https://www.autopsy.com.

[2] PostgreSQL, *PostgreSQL*, Website, https://www.postgresql.org.

[3] Apache, *Apache Solr*, Website, http://lucene.apache.org/solr.

[4] Apache, *Apache ActiveMQ*, Website, https://activemq.apache.org.

[5] T. S. Kit, *The Sleuth Kit*, Website, http://www.sleuthkit.org.

[6] NIST, *Forensic Science*, Website, https://www.nist.gov/topics/forensic-science.

[7] L. Daniel and L. Daniel, *Digital Forensics for Legal Professionals.* Syngress, 2011.

[8] NIST, *Digital Evidence*, Website, https://www.nist.gov/topics/digital-evidence.

[9] Microsoft, *Cloud Computing*, Website, https://azure.microsoft.com/en-in/overview/what-is-cloud-computing.

[10] SearchCloudComputing, *Software as a Service*, Website, https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service.

[11] NOLO, *Chain of Custody*, Website, https://www.nolo.com/legal-encyclopedia/what-chain-custody.html.

[12] Nuix, *Nuix Lab*, Website, https://www.nuix.com/solutions/investigations.

[13] elastic, *Elasticsearch*, Website, https://www.elastic.co/products/elasticsearch.

[14] G. Software, *EnCase Forensic*, Website, https://www.guidancesoftware.com/encase-forensic.

[15] A. Data, *Forensics Toolkit*, Website, https://accessdata.com/products-services/forensic-toolkit-ftk.

[16] Atlassian, *JIRA*, Website, https://www.atlassian.com/software/jira.

[17] Quarkus, *Quarkus*, Website, https://quarkus.io.

[18] React, *React*, Website, https://reactjs.org.

[19] O. Source, *Open Source*, Website, https://opensource.com/resources/what-open-source.

[20] Microsoft, *GitHub*, Website, https://github.com.

[21] S. Consortium, *SQLite*, Website, https://sqlite.org/index.html.

[22] Schneier, *The Blowfish Encryption Algorithm*, Website, https://www.schneier.com/academic/blowfish.

[23] whatisaqrcode.co.uk, *What is a QR Code?*, Website, https://www.whatisaqrcode.co.uk.

[24] Yubico, *Using a U2F Library*, Website, https://developers.yubico.com/U2F/Libraries/Using_a_library.html.

[25] FileZilla, *FileZilla*, Website, https://filezilla-project.org.

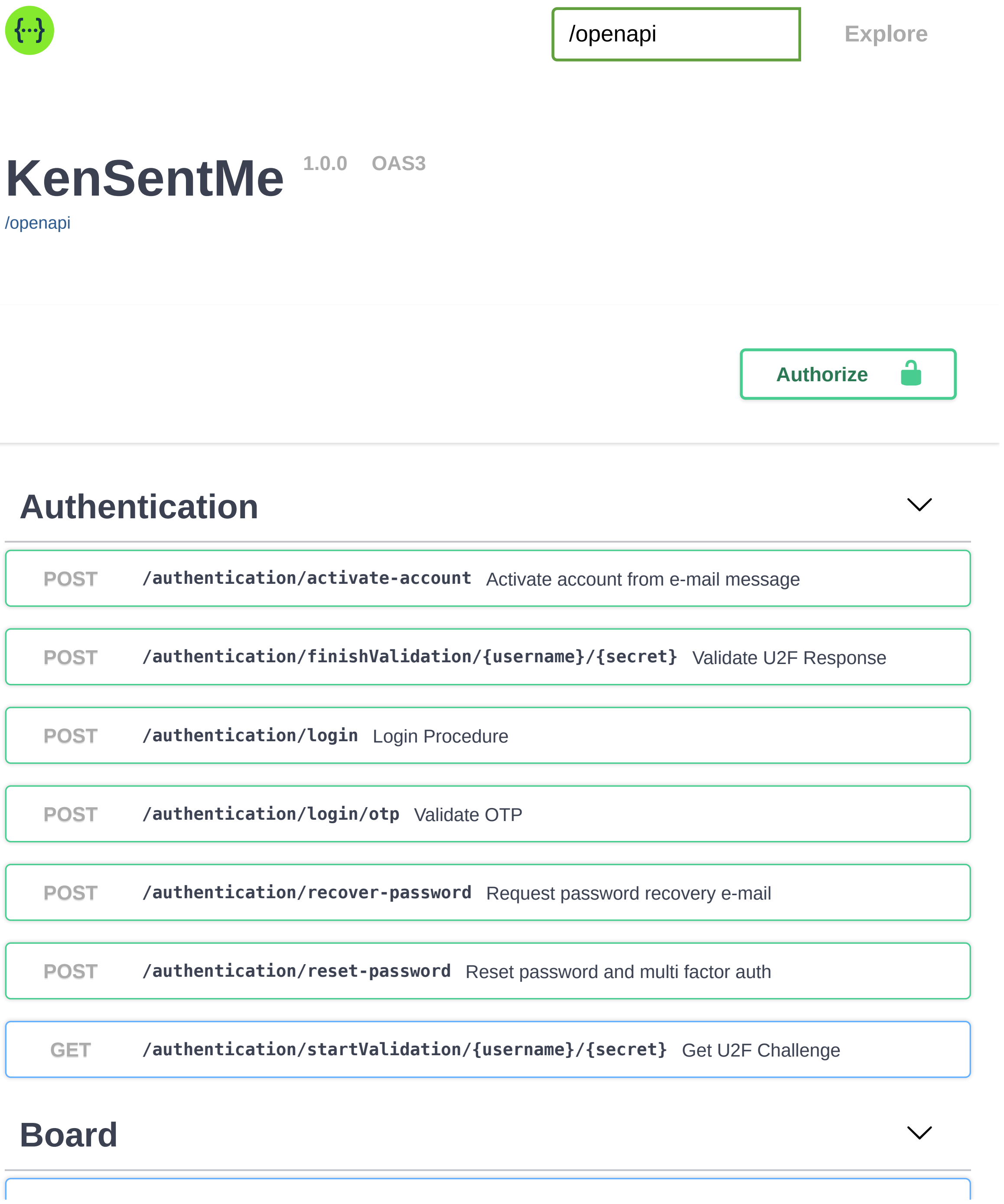
[26] A. Priestman, *Autopsy Logical Imager*, Presentation, https://www.osdfcon.org/presentations/2019/Ann-Priestman_Introducing-the-Autopsy-Logical-Imager.pdf.

[27] Apache, *NetBeans*, Website, https://netbeans.org.

APPENDICES

APPENDIX A

The endpoints of the KenSentMe platform are presented in the following pages, using the Swagger tool.



/openapi

Explore

KenSentMe 1.0.0 OAS3

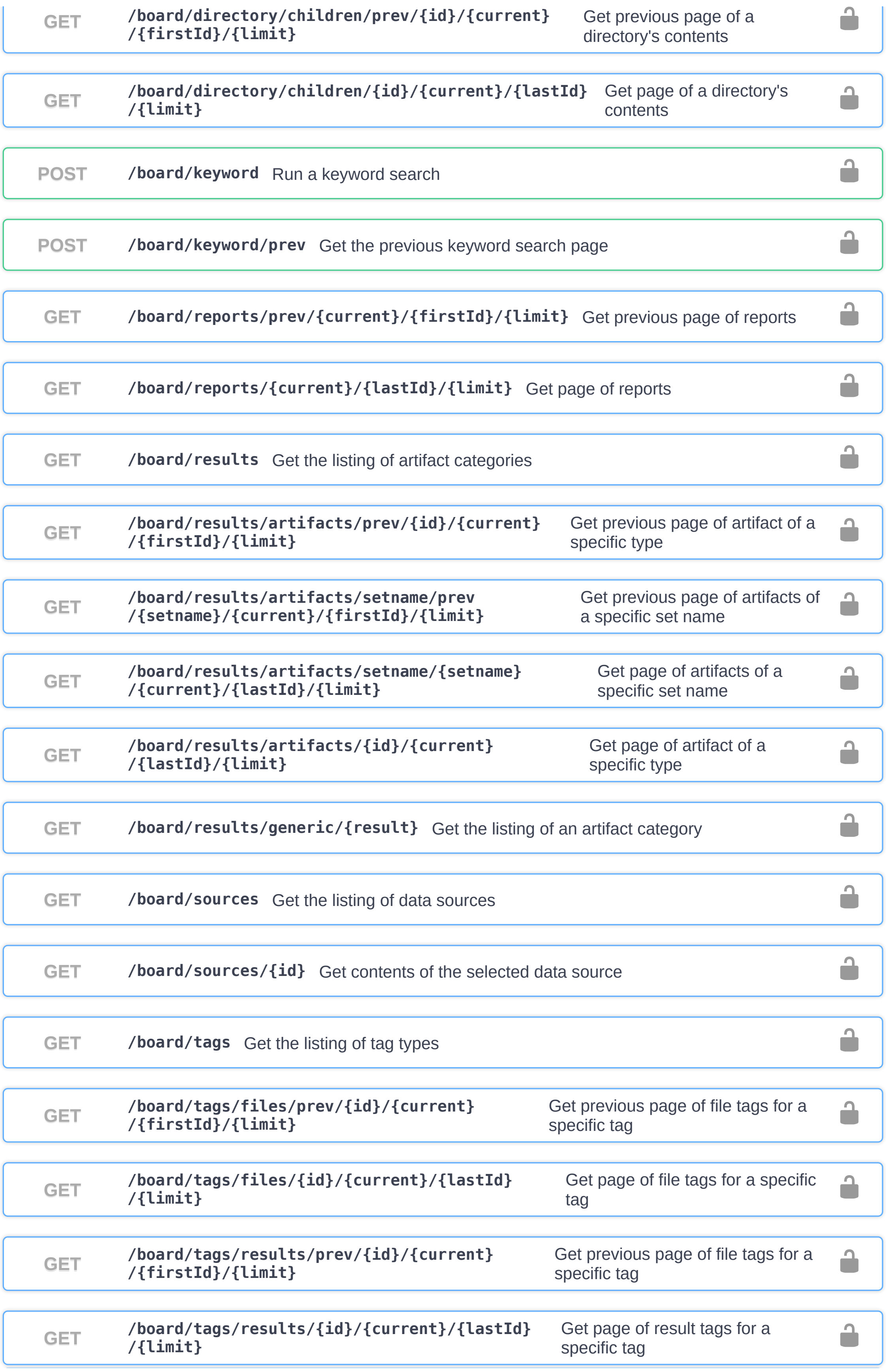
/openapi

Authorize

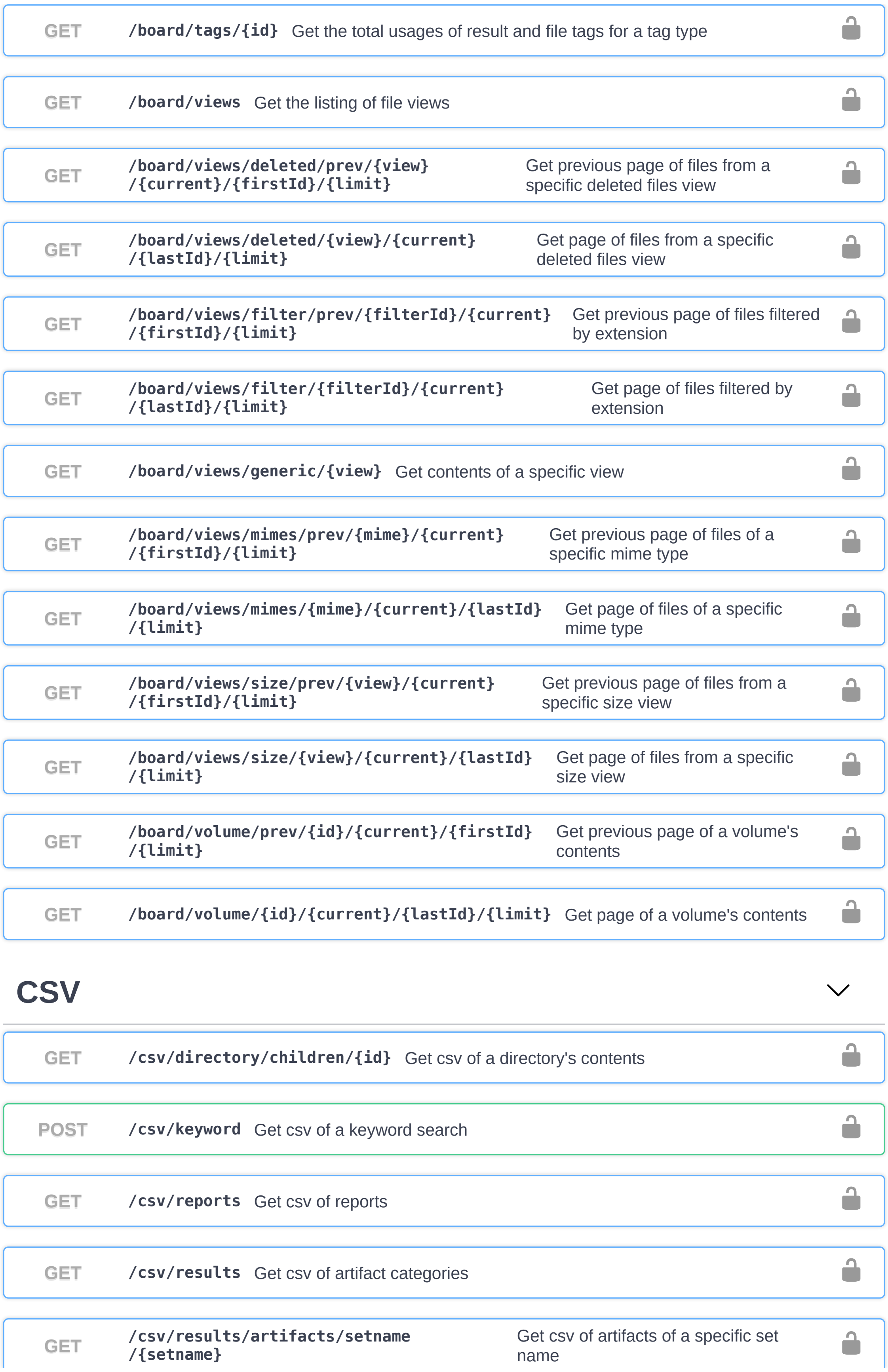
Authentication

POST	`/authentication/activate-account`	Activate account from e-mail message
POST	`/authentication/finishValidation/{username}/{secret}`	Validate U2F Response
POST	`/authentication/login`	Login Procedure
POST	`/authentication/login/otp`	Validate OTP
POST	`/authentication/recover-password`	Request password recovery e-mail
POST	`/authentication/reset-password`	Reset password and multi factor auth
GET	`/authentication/startValidation/{username}/{secret}`	Get U2F Challenge

Board



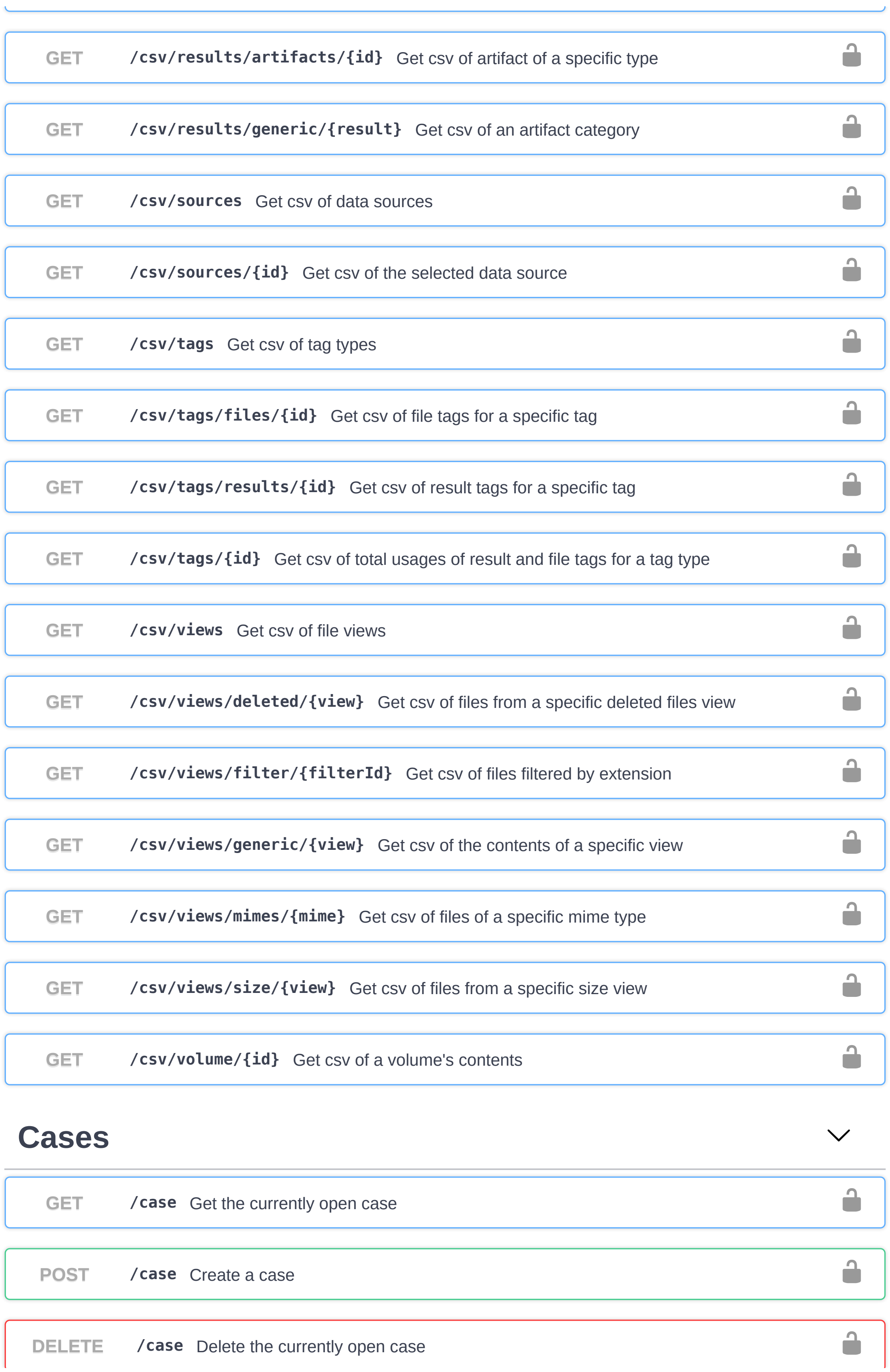
GET	`/board/directory/children/prev/{id}/{current}/{firstId}/{limit}`	Get previous page of a directory's contents
GET	`/board/directory/children/{id}/{current}/{lastId}/{limit}`	Get page of a directory's contents
POST	`/board/keyword`	Run a keyword search
POST	`/board/keyword/prev`	Get the previous keyword search page
GET	`/board/reports/prev/{current}/{firstId}/{limit}`	Get previous page of reports
GET	`/board/reports/{current}/{lastId}/{limit}`	Get page of reports
GET	`/board/results`	Get the listing of artifact categories
GET	`/board/results/artifacts/prev/{id}/{current}/{firstId}/{limit}`	Get previous page of artifact of a specific type
GET	`/board/results/artifacts/setname/prev/{setname}/{current}/{firstId}/{limit}`	Get previous page of artifacts of a specific set name
GET	`/board/results/artifacts/setname/{setname}/{current}/{lastId}/{limit}`	Get page of artifacts of a specific set name
GET	`/board/results/artifacts/{id}/{current}/{lastId}/{limit}`	Get page of artifact of a specific type
GET	`/board/results/generic/{result}`	Get the listing of an artifact category
GET	`/board/sources`	Get the listing of data sources
GET	`/board/sources/{id}`	Get contents of the selected data source
GET	`/board/tags`	Get the listing of tag types
GET	`/board/tags/files/prev/{id}/{current}/{firstId}/{limit}`	Get previous page of file tags for a specific tag
GET	`/board/tags/files/{id}/{current}/{lastId}/{limit}`	Get page of file tags for a specific tag
GET	`/board/tags/results/prev/{id}/{current}/{firstId}/{limit}`	Get previous page of file tags for a specific tag
GET	`/board/tags/results/{id}/{current}/{lastId}/{limit}`	Get page of result tags for a specific tag



GET	`/board/tags/{id}`	Get the total usages of result and file tags for a tag type
GET	`/board/views`	Get the listing of file views
GET	`/board/views/deleted/prev/{view}/{current}/{firstId}/{limit}`	Get previous page of files from a specific deleted files view
GET	`/board/views/deleted/{view}/{current}/{lastId}/{limit}`	Get page of files from a specific deleted files view
GET	`/board/views/filter/prev/{filterId}/{current}/{firstId}/{limit}`	Get previous page of files filtered by extension
GET	`/board/views/filter/{filterId}/{current}/{lastId}/{limit}`	Get page of files filtered by extension
GET	`/board/views/generic/{view}`	Get contents of a specific view
GET	`/board/views/mimes/prev/{mime}/{current}/{firstId}/{limit}`	Get previous page of files of a specific mime type
GET	`/board/views/mimes/{mime}/{current}/{lastId}/{limit}`	Get page of files of a specific mime type
GET	`/board/views/size/prev/{view}/{current}/{firstId}/{limit}`	Get previous page of files from a specific size view
GET	`/board/views/size/{view}/{current}/{lastId}/{limit}`	Get page of files from a specific size view
GET	`/board/volume/prev/{id}/{current}/{firstId}/{limit}`	Get previous page of a volume's contents
GET	`/board/volume/{id}/{current}/{lastId}/{limit}`	Get page of a volume's contents

CSV

GET	`/csv/directory/children/{id}`	Get csv of a directory's contents
POST	`/csv/keyword`	Get csv of a keyword search
GET	`/csv/reports`	Get csv of reports
GET	`/csv/results`	Get csv of artifact categories
GET	`/csv/results/artifacts/setname/{setname}`	Get csv of artifacts of a specific set name



GET `/csv/results/artifacts/{id}` Get csv of artifact of a specific type

GET `/csv/results/generic/{result}` Get csv of an artifact category

GET `/csv/sources` Get csv of data sources

GET `/csv/sources/{id}` Get csv of the selected data source

GET `/csv/tags` Get csv of tag types

GET `/csv/tags/files/{id}` Get csv of file tags for a specific tag

GET `/csv/tags/results/{id}` Get csv of result tags for a specific tag

GET `/csv/tags/{id}` Get csv of total usages of result and file tags for a tag type

GET `/csv/views` Get csv of file views

GET `/csv/views/deleted/{view}` Get csv of files from a specific deleted files view

GET `/csv/views/filter/{filterId}` Get csv of files filtered by extension

GET `/csv/views/generic/{view}` Get csv of the contents of a specific view

GET `/csv/views/mimes/{mime}` Get csv of files of a specific mime type

GET `/csv/views/size/{view}` Get csv of files from a specific size view

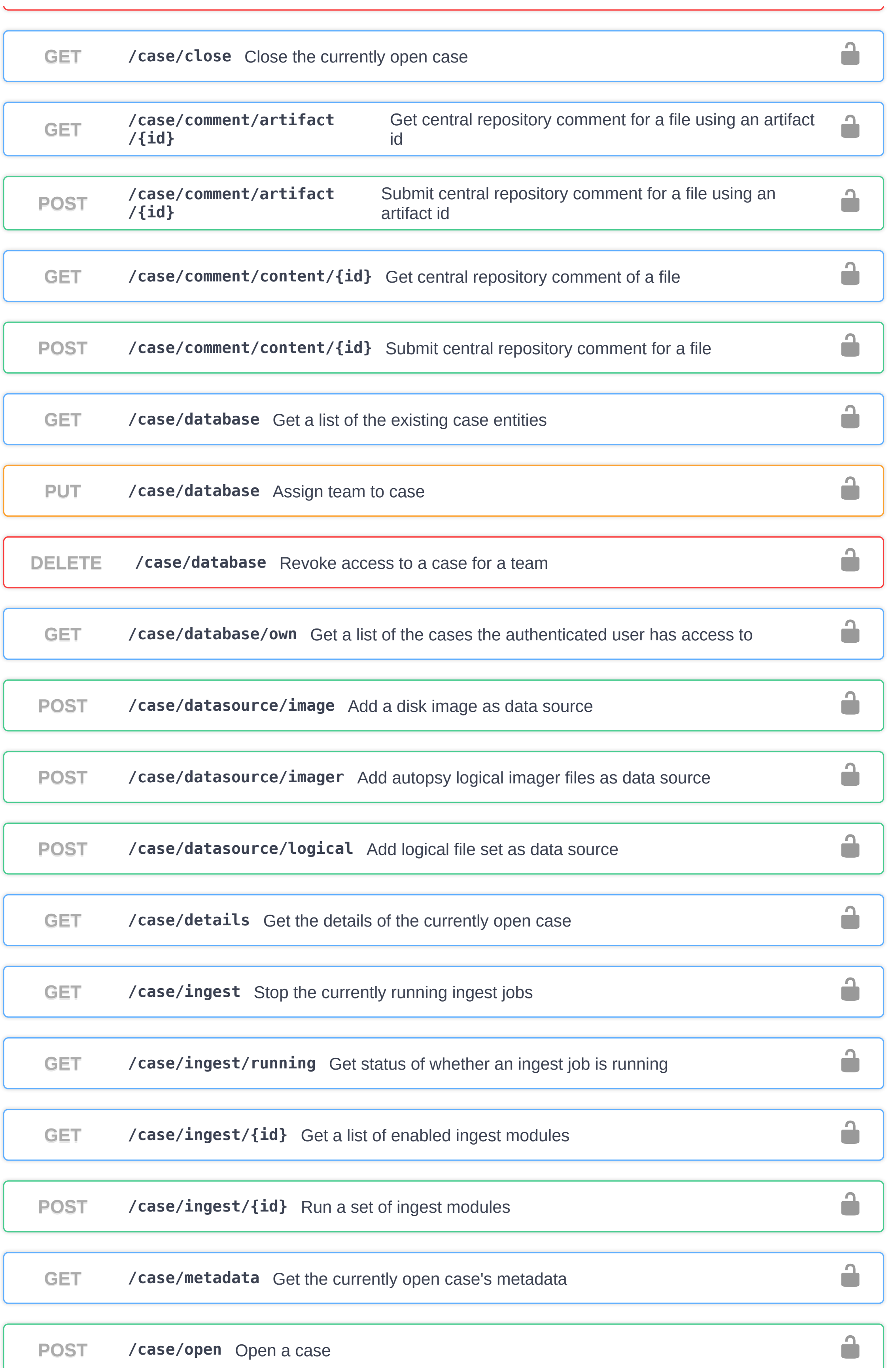
GET `/csv/volume/{id}` Get csv of a volume's contents

Cases

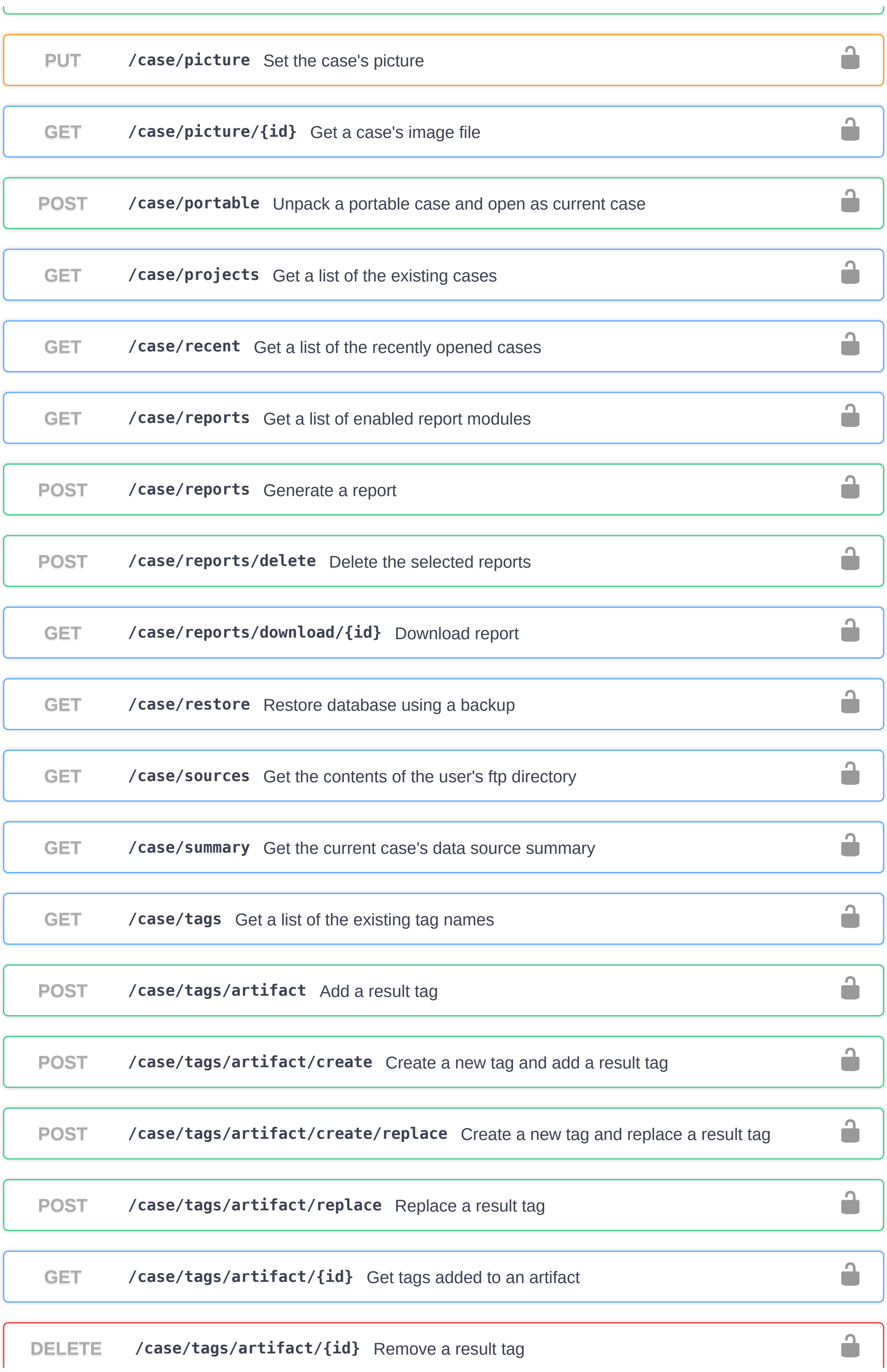
GET `/case` Get the currently open case

POST `/case` Create a case

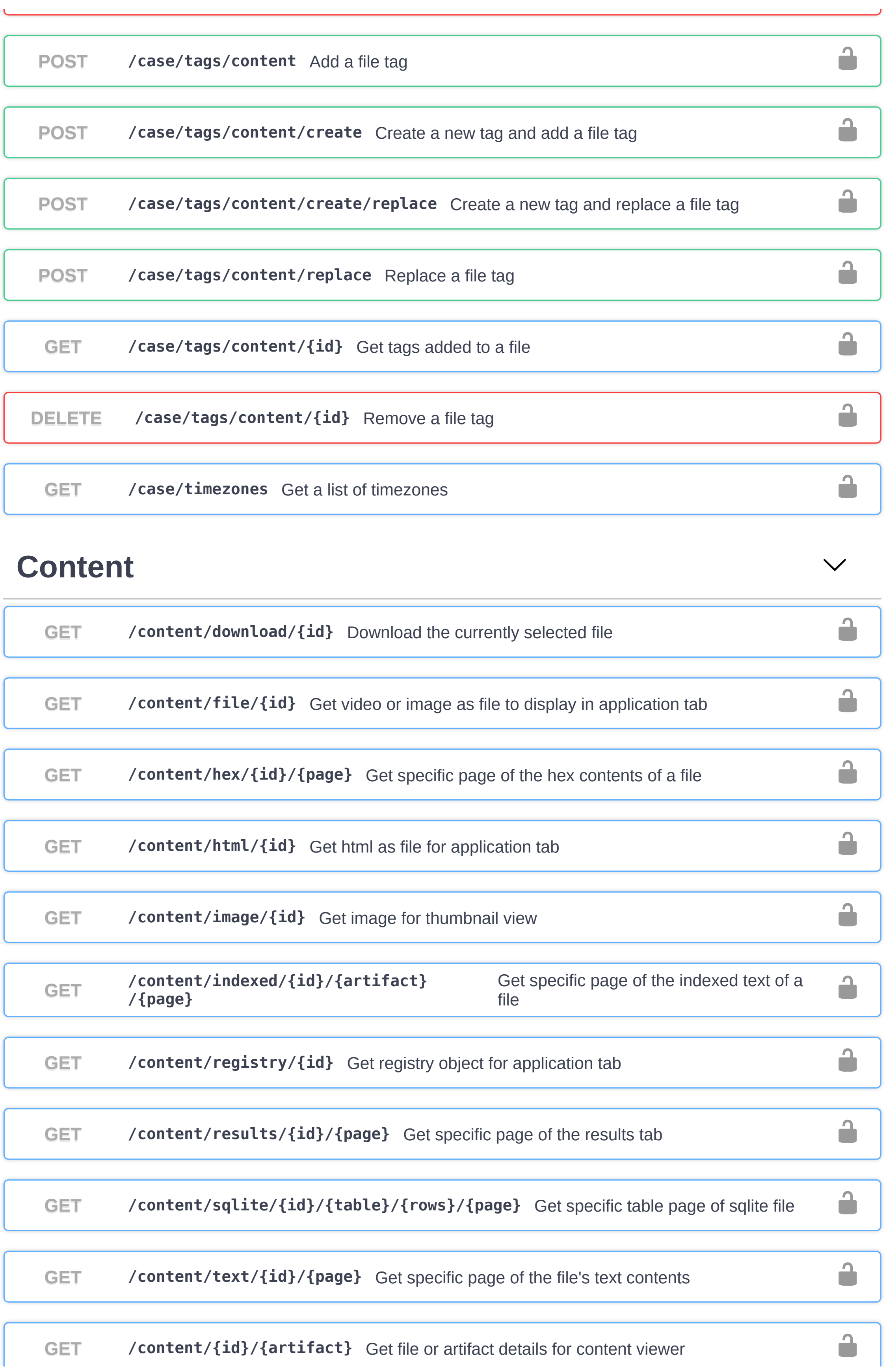
DELETE `/case` Delete the currently open case



GET	`/case/close`	Close the currently open case
GET	`/case/comment/artifact/{id}`	Get central repository comment for a file using an artifact id
POST	`/case/comment/artifact/{id}`	Submit central repository comment for a file using an artifact id
GET	`/case/comment/content/{id}`	Get central repository comment of a file
POST	`/case/comment/content/{id}`	Submit central repository comment for a file
GET	`/case/database`	Get a list of the existing case entities
PUT	`/case/database`	Assign team to case
DELETE	`/case/database`	Revoke access to a case for a team
GET	`/case/database/own`	Get a list of the cases the authenticated user has access to
POST	`/case/datasource/image`	Add a disk image as data source
POST	`/case/datasource/imager`	Add autopsy logical imager files as data source
POST	`/case/datasource/logical`	Add logical file set as data source
GET	`/case/details`	Get the details of the currently open case
GET	`/case/ingest`	Stop the currently running ingest jobs
GET	`/case/ingest/running`	Get status of whether an ingest job is running
GET	`/case/ingest/{id}`	Get a list of enabled ingest modules
POST	`/case/ingest/{id}`	Run a set of ingest modules
GET	`/case/metadata`	Get the currently open case's metadata
POST	`/case/open`	Open a case



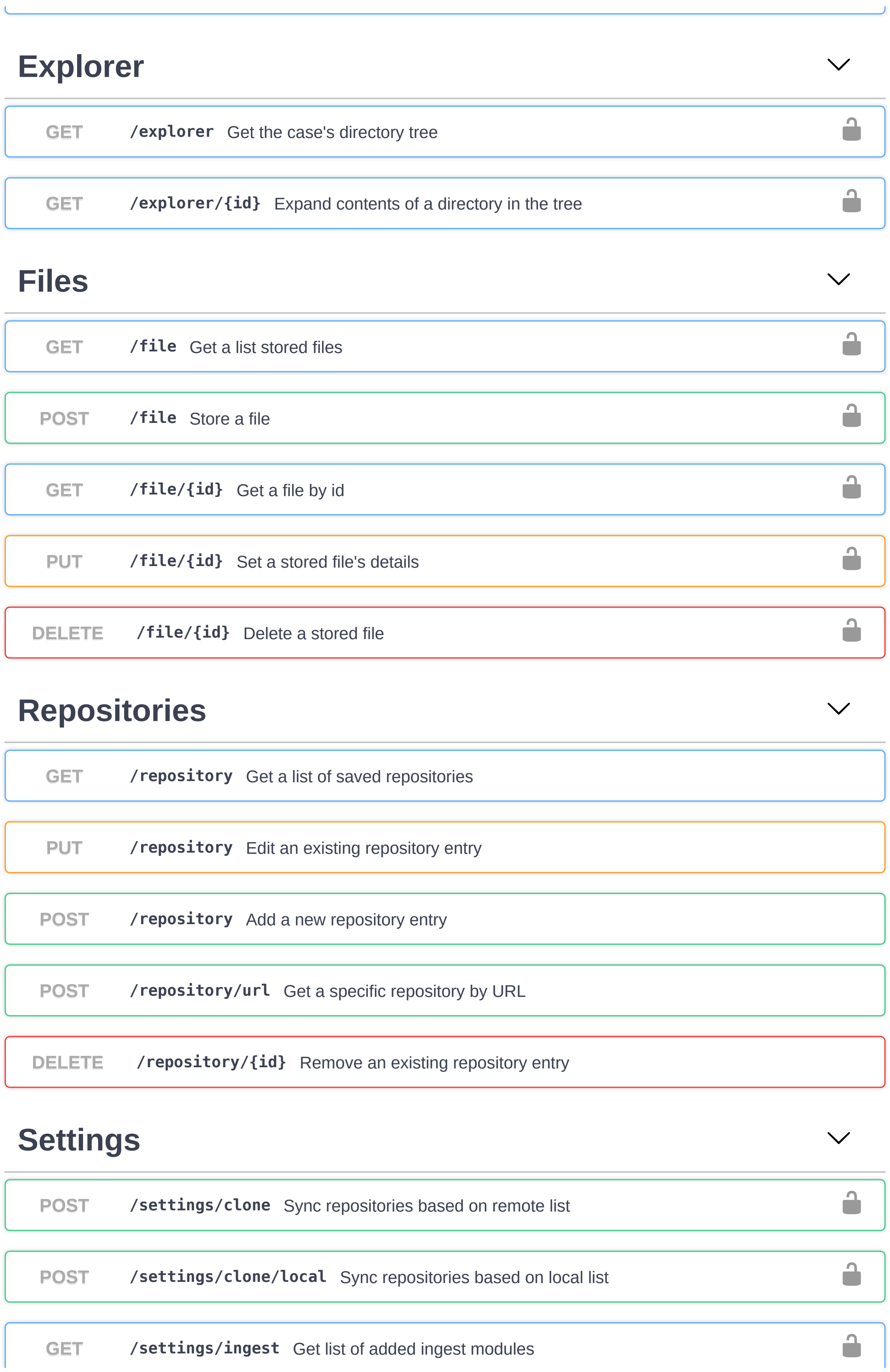
PUT	`/case/picture`	Set the case's picture
GET	`/case/picture/{id}`	Get a case's image file
POST	`/case/portable`	Unpack a portable case and open as current case
GET	`/case/projects`	Get a list of the existing cases
GET	`/case/recent`	Get a list of the recently opened cases
GET	`/case/reports`	Get a list of enabled report modules
POST	`/case/reports`	Generate a report
POST	`/case/reports/delete`	Delete the selected reports
GET	`/case/reports/download/{id}`	Download report
GET	`/case/restore`	Restore database using a backup
GET	`/case/sources`	Get the contents of the user's ftp directory
GET	`/case/summary`	Get the current case's data source summary
GET	`/case/tags`	Get a list of the existing tag names
POST	`/case/tags/artifact`	Add a result tag
POST	`/case/tags/artifact/create`	Create a new tag and add a result tag
POST	`/case/tags/artifact/create/replace`	Create a new tag and replace a result tag
POST	`/case/tags/artifact/replace`	Replace a result tag
GET	`/case/tags/artifact/{id}`	Get tags added to an artifact
DELETE	`/case/tags/artifact/{id}`	Remove a result tag



POST	`/case/tags/content`	Add a file tag
POST	`/case/tags/content/create`	Create a new tag and add a file tag
POST	`/case/tags/content/create/replace`	Create a new tag and replace a file tag
POST	`/case/tags/content/replace`	Replace a file tag
GET	`/case/tags/content/{id}`	Get tags added to a file
DELETE	`/case/tags/content/{id}`	Remove a file tag
GET	`/case/timezones`	Get a list of timezones

Content

GET	`/content/download/{id}`	Download the currently selected file
GET	`/content/file/{id}`	Get video or image as file to display in application tab
GET	`/content/hex/{id}/{page}`	Get specific page of the hex contents of a file
GET	`/content/html/{id}`	Get html as file for application tab
GET	`/content/image/{id}`	Get image for thumbnail view
GET	`/content/indexed/{id}/{artifact}/{page}`	Get specific page of the indexed text of a file
GET	`/content/registry/{id}`	Get registry object for application tab
GET	`/content/results/{id}/{page}`	Get specific page of the results tab
GET	`/content/sqlite/{id}/{table}/{rows}/{page}`	Get specific table page of sqlite file
GET	`/content/text/{id}/{page}`	Get specific page of the file's text contents
GET	`/content/{id}/{artifact}`	Get file or artifact details for content viewer



Explorer

GET `/explorer` Get the case's directory tree

GET `/explorer/{id}` Expand contents of a directory in the tree

Files

GET `/file` Get a list stored files

POST `/file` Store a file

GET `/file/{id}` Get a file by id

PUT `/file/{id}` Set a stored file's details

DELETE `/file/{id}` Delete a stored file

Repositories

GET `/repository` Get a list of saved repositories

PUT `/repository` Edit an existing repository entry

POST `/repository` Add a new repository entry

POST `/repository/url` Get a specific repository by URL

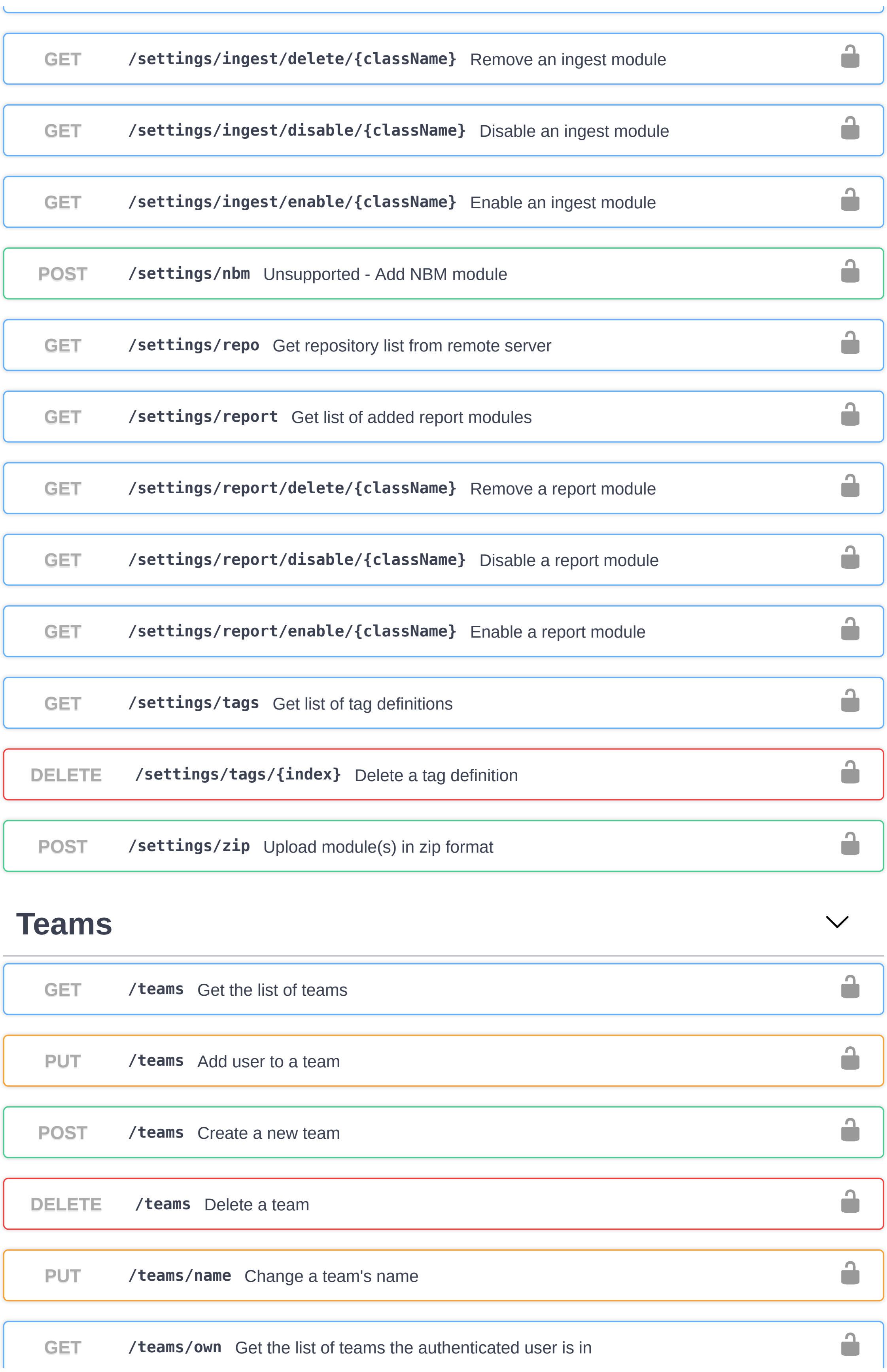
DELETE `/repository/{id}` Remove an existing repository entry

Settings

POST `/settings/clone` Sync repositories based on remote list

POST `/settings/clone/local` Sync repositories based on local list

GET `/settings/ingest` Get list of added ingest modules



GET `/settings/ingest/delete/{className}` Remove an ingest module

GET `/settings/ingest/disable/{className}` Disable an ingest module

GET `/settings/ingest/enable/{className}` Enable an ingest module

POST `/settings/nbm` Unsupported - Add NBM module

GET `/settings/repo` Get repository list from remote server

GET `/settings/report` Get list of added report modules

GET `/settings/report/delete/{className}` Remove a report module

GET `/settings/report/disable/{className}` Disable a report module

GET `/settings/report/enable/{className}` Enable a report module

GET `/settings/tags` Get list of tag definitions

DELETE `/settings/tags/{index}` Delete a tag definition

POST `/settings/zip` Upload module(s) in zip format

Teams

GET `/teams` Get the list of teams

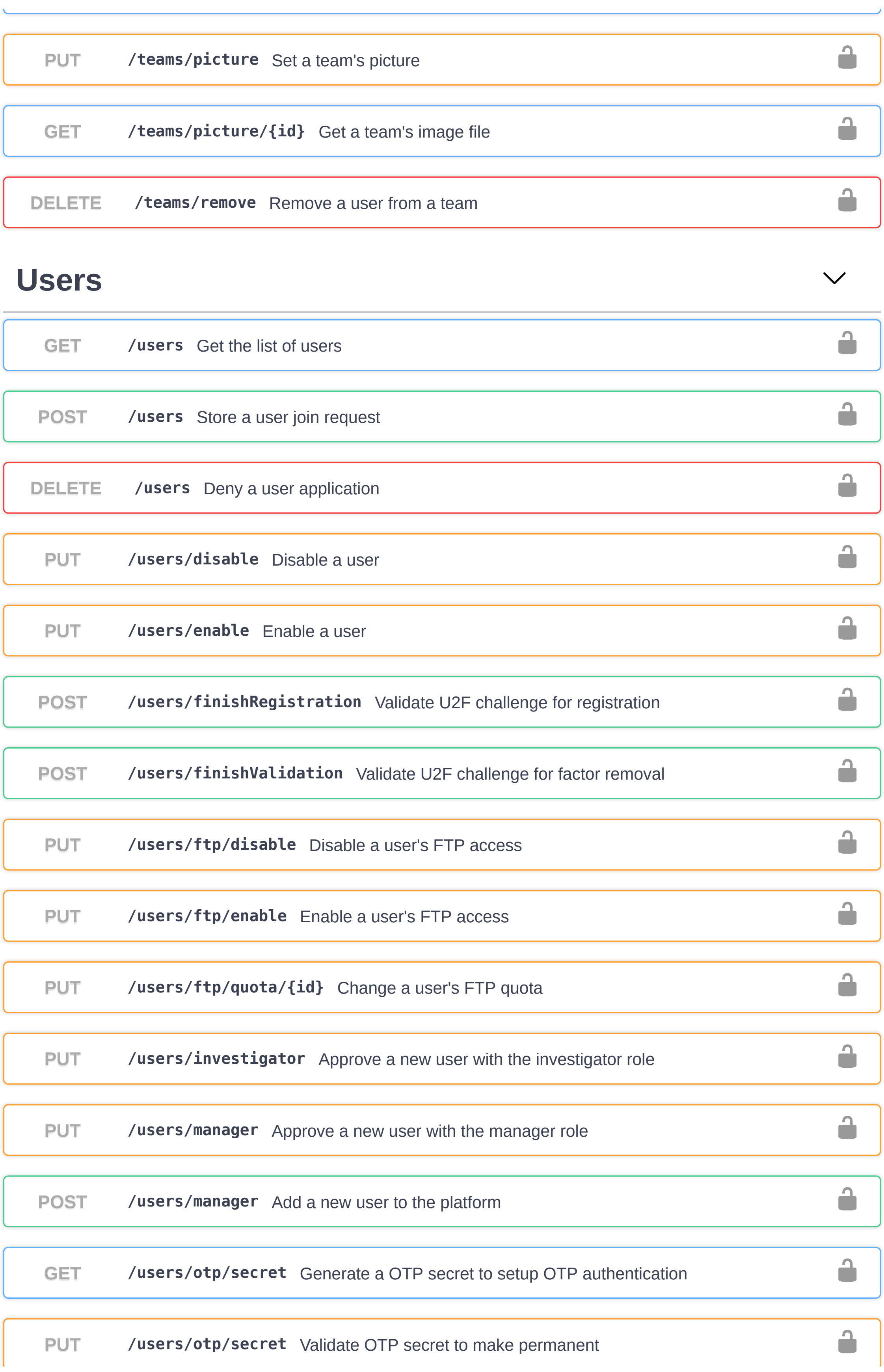
PUT `/teams` Add user to a team

POST `/teams` Create a new team

DELETE `/teams` Delete a team

PUT `/teams/name` Change a team's name

GET `/teams/own` Get the list of teams the authenticated user is in



PUT /teams/picture Set a team's picture

GET /teams/picture/{id} Get a team's image file

DELETE /teams/remove Remove a user from a team

Users

GET /users Get the list of users

POST /users Store a user join request

DELETE /users Deny a user application

PUT /users/disable Disable a user

PUT /users/enable Enable a user

POST /users/finishRegistration Validate U2F challenge for registration

POST /users/finishValidation Validate U2F challenge for factor removal

PUT /users/ftp/disable Disable a user's FTP access

PUT /users/ftp/enable Enable a user's FTP access

PUT /users/ftp/quota/{id} Change a user's FTP quota

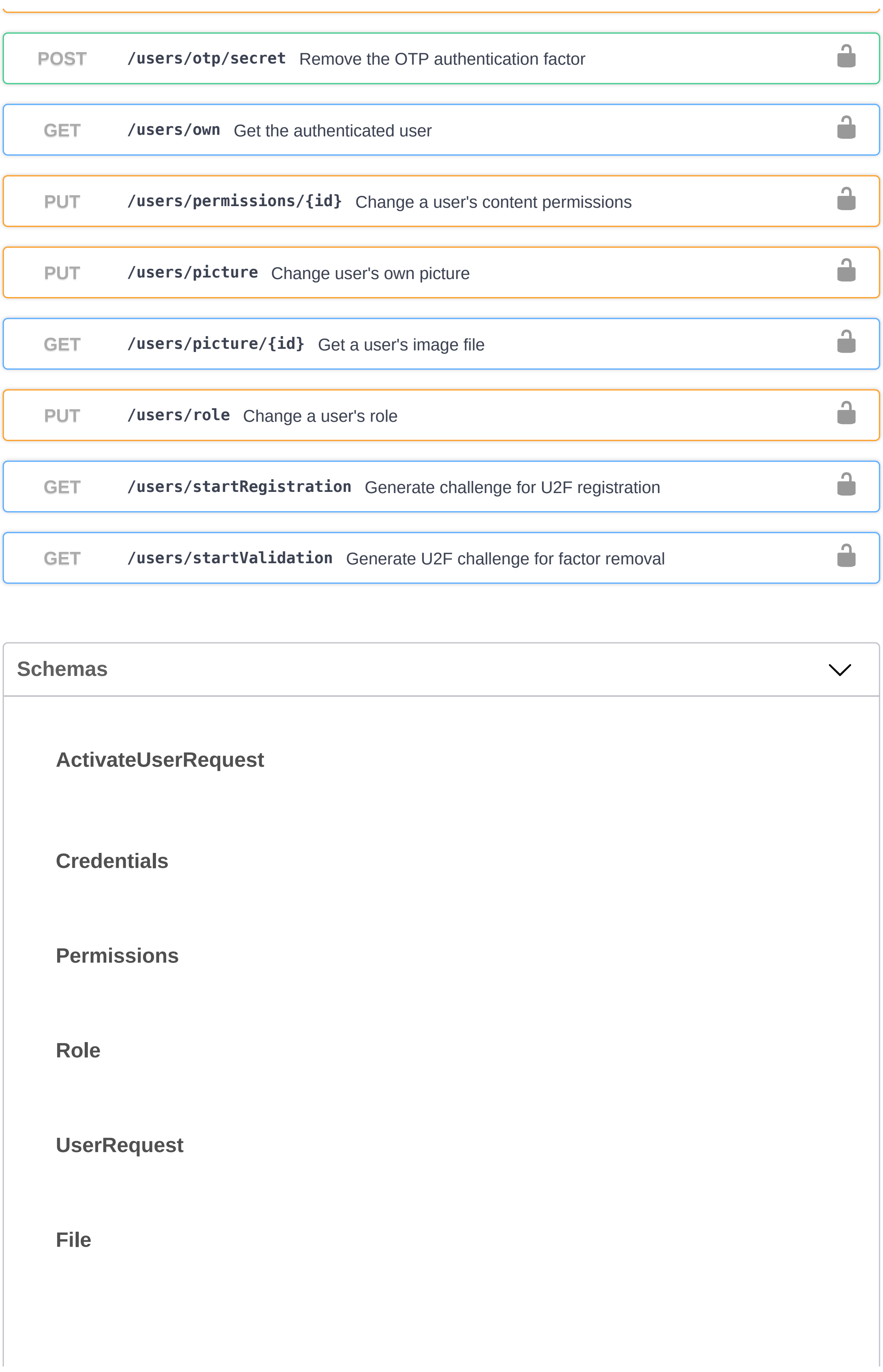
PUT /users/investigator Approve a new user with the investigator role

PUT /users/manager Approve a new user with the manager role

POST /users/manager Add a new user to the platform

GET /users/otp/secret Generate a OTP secret to setup OTP authentication

PUT /users/otp/secret Validate OTP secret to make permanent



POST `/users/otp/secret` Remove the OTP authentication factor

GET `/users/own` Get the authenticated user

PUT `/users/permissions/{id}` Change a user's content permissions

PUT `/users/picture` Change user's own picture

GET `/users/picture/{id}` Get a user's image file

PUT `/users/role` Change a user's role

GET `/users/startRegistration` Generate challenge for U2F registration

GET `/users/startValidation` Generate U2F challenge for factor removal

Schemas

ActivateUserRequest

Credentials

Permissions

Role

UserRequest

File

Repository

ListRepository

KeywordRequest

ListLong

AddTagRequest

ImageRequest

LogicalRequest

CaseAddRequest

NewTagRequest

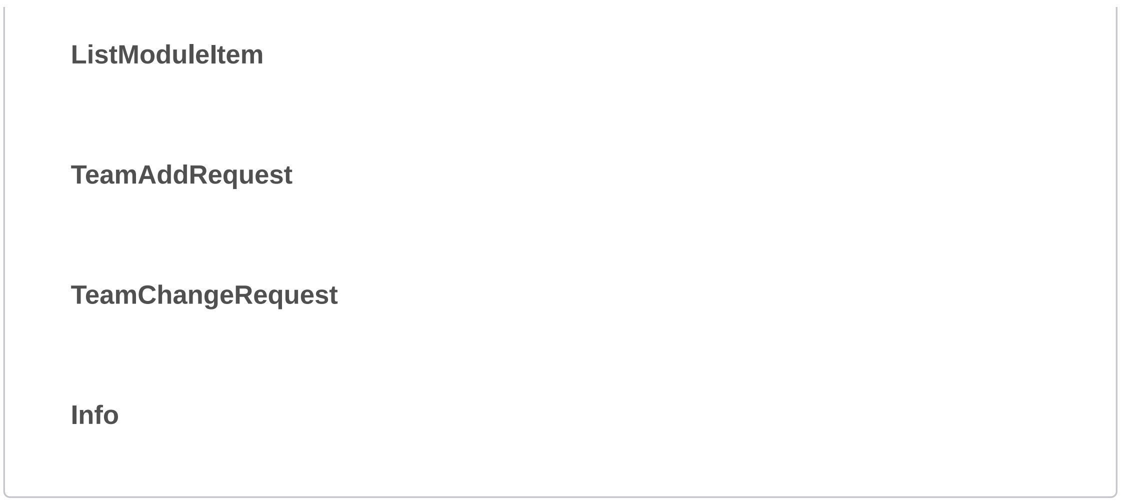
CaseDetails

ModuleAttributes

ListModuleAttributes

ReportModuleItem

ModuleItem



ListModuleItem

TeamAddRequest

TeamChangeRequest

Info

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"AUTOPSY – Enhanced distributed forensic analysis"*, é original e foi realizado por Pedro Henrique Gaspar Cordeiro Ferreira (2180078) sob orientação de Professora Doutora Marisa da Silva Maximiano (marisa.maximiano@ipleiria.pt).

Leiria, 25 de Fevereiro de 2020

Pedro Henrique Gaspar Cordeiro Ferreira