System Requirements Specification

version 1.0

for

Maelstrom

prepared by

name	email
Iulian Rotaru	iulian@rotaru.fr

Document History

Date	Version	Description	Author
05/12/2018	1.0	Filled Introduction, Overall description and Specific Requirements	Iulian Rotaru

Table of Contents

1. Introduction

This document will provide an overview on the Maelstrom Crypto Data Scraper.

1.1. Purpose

The purpose of this document is to provide a high-level abstract description of the Maelstrom Daemon and Maelstrom CLI. This document is made for the various people that might work on the project.

1.2. Scope

This document concerns the Administrator that is going to operate the Maelstrom Daemon. It influences how the data is handled by the system and how the Administrator is able to interact with it.

1.3. Definitions, acronyms, and abbreviations

Administrator	The user that will use an interface to configure and interact with the Maelstrom Daemon.
SRS	System Requirements Specification
SAD	Software Architecture Document
CLI	Command Line Interface
API	Application Programming Interface
NPM	Node Package Manager
CSV	Comma-Separated Values
Daemon	Software running in the background of a system

1.4. References

2. Overall description

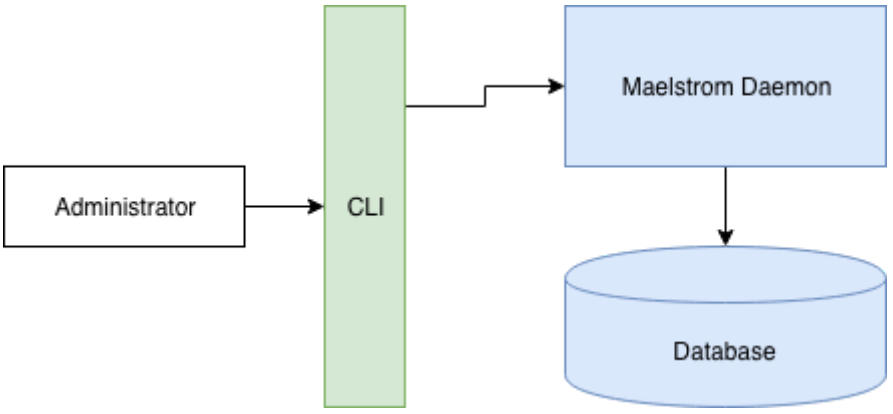

Figure 1. Client/CLI-Server Architecture

2.1. Product perspective

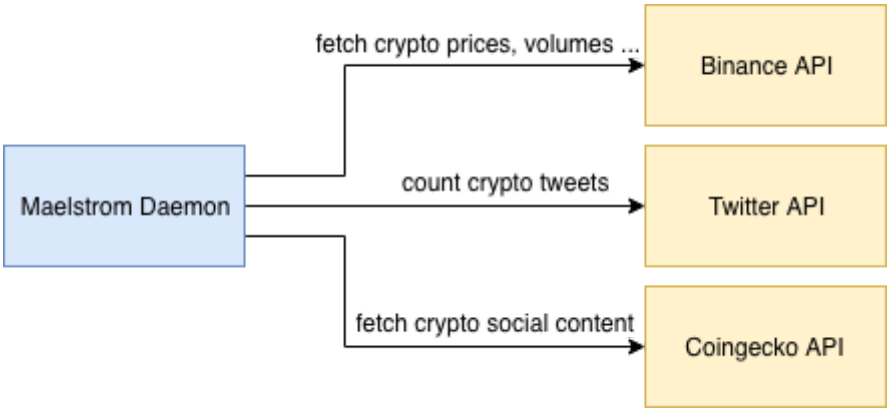

Figure 2. Alien System Dependencies

The system will fetch informations from outside sources. The Binance API will provide cryptocurrency prices, volumes and exchanges informations to fuel the datasets. The Twitter API will be queried to recover the amount of tweets mentioning the cryptocurrencies that are fetched by the Maelstrom Daemon. The Coingeck API provides various useful social data.

2.2. Product functions

This product provides a CLI to the Maelstrom Administrator to easily configure and manage the Maelstrom Daemon. The Daemon's role is to fetch informations about specific

cryptocurrencies specified by the Administrator. The CLI will allow the Administrator to start and stop the scraper, add or remove cryptocurrencies to follow, recover or remove datasets.

2.3. User characteristics

The product is intended to be used by a System Administrator that wants to manage a Maelstrom Daemon running on its System. This Administrator must have basic knowledge of UNIX Shell to properly use the CLI.

2.4. Constraints

The CLI communicate only to the local system. The Administrator must be logged in its system to perform actions upon the Maelstrom Daemon with the CLI.

2.5. Assumptions and dependencies

The Maelstrom Daemon requires an UNIX system to run. The CLI requires a shell to be summoned and used. The system should have the following dependencies installed globally.

Node.js 8.x.x	https://nodejs.org/
NPM 6.x.x	https://www.npmjs.com/get-npm

3. Specific requirements

This section contains all requirements in detail: Functional as well as non-functional requirements (quality attributes and constraints). The quality attributes are listed according to the *ISO/IEC 25010* standard that classifies software quality in a structured set of characteristics and sub-characteristics.

3.1. External interfaces

The Maelstrom Daemon exposes an HTTP Rest API. The CLI will make local requests to interact with the Daemon. Version 1 of the Maelstrom Daemon will not allow non-local requests. Using an HTTP Rest API will help when developing a web interface in future versions.

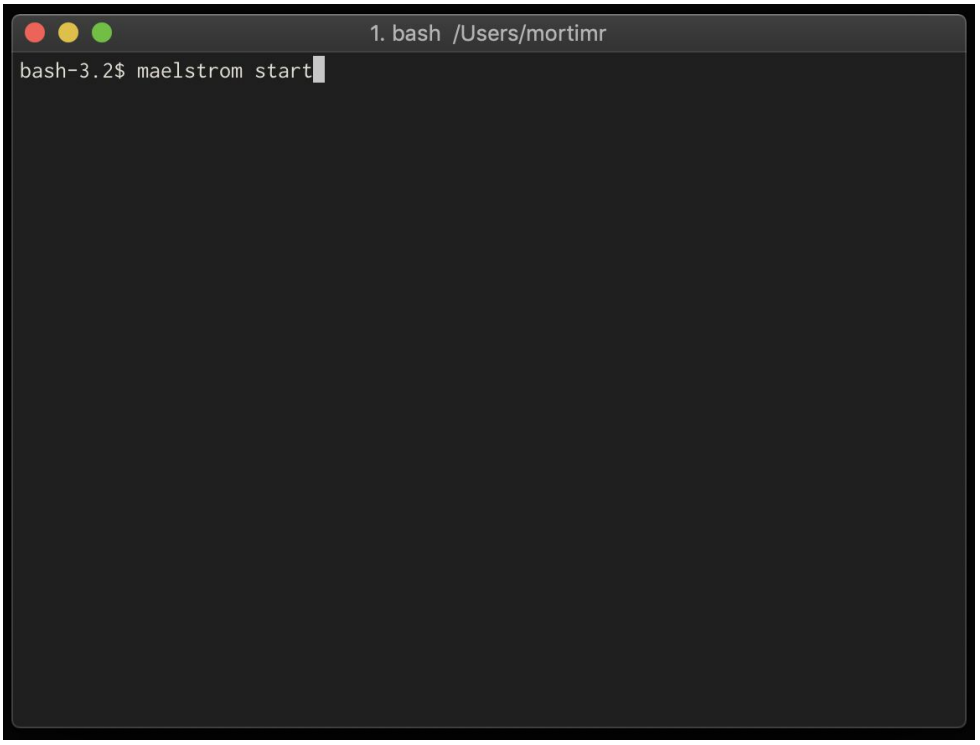

Figure 3. An example of CLI usage

3.2. Functional requirements

Functional requirements capture the intended behaviour of the Maelstrom Daemon and Maelstrom CLI. This section contains the *Actor Goal List* and the *Use Case view*.

3.2.1. Actor goal list

Actor	Goal
Administrator	● Start or Stop the scraping process ● Get current daemon status ● Add or Remove coins to scrap ● Recover stored informations as CSV

3.2.2. Use case view

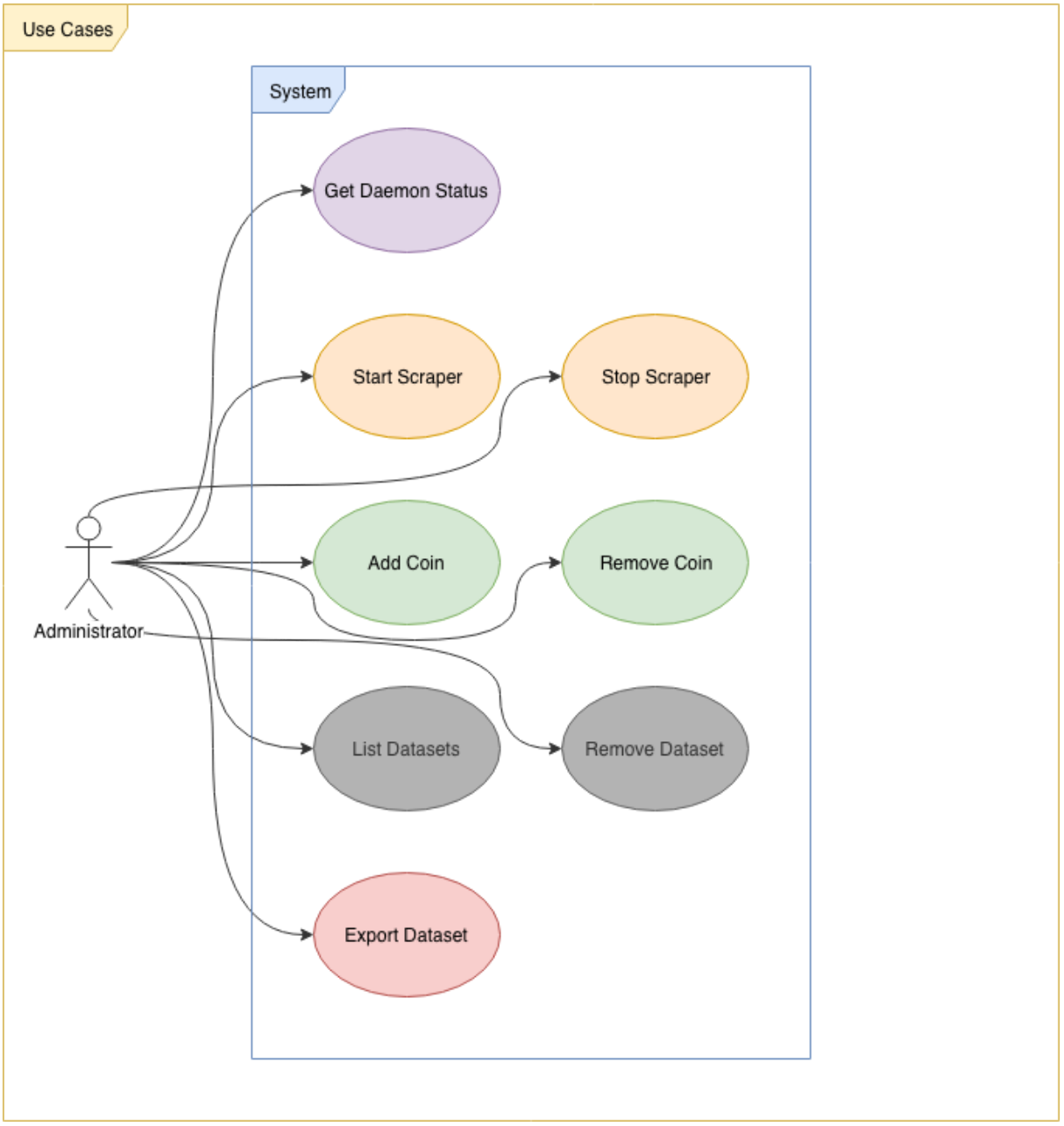

Figure 3. Use case model

List of Use Cases

Name	Get Daemon Status
Code	UC1
Importance	Non-critical
Casual Definition	With the help of the CLI, the Administrator can query the Daemon status. The Daemon should return its current status (running or not) and a list of cryptocurrencies that are currently fetched.

Name	Start Scraper
Code	UC2
Importance	Critical
Primary Actor	Administrator
Preconditions	● Administrator is logged in a system where the Maelstrom daemon is running ● Administrator is logged in a system where the Maelstrom CLI is installed ● Administrator uses CLI to start the scraper
Postconditions	● Scraper starts ● Daemon status is changed

Name	Stop Scraper
Code	UC3
Importance	Critical
Primary Actor	Administrator
Preconditions	● Administrator is logged in a system where the Maelstrom daemon is running ● Administrator is logged in a system where the Maelstrom CLI is installed ● Administrator uses CLI to stop the scraper
Postconditions	● Scraper stops ● Daemon status is changed

Name	Add Coin
Code	UC4
Importance	Critical
Primary Actor	Administrator
Preconditions	• Administrator is logged in a system where the Maelstrom daemon is running • Administrator is logged in a system where the Maelstrom CLI is installed • Administrator uses CLI to add a coin to scrap
Postconditions	• List of coins to scrap is updated • Scraper starts scraping new coin

Name	Remove Coin
Code	UC5
Importance	Critical
Primary Actor	Administrator
Preconditions	• Administrator is logged in a system where the Maelstrom daemon is running • Administrator is logged in a system where the Maelstrom CLI is installed • Administrator uses CLI to remove a coin
Postconditions	• List of coins to scrap is updated • Scraper stops scraping removed coin

Name	List Datasets
Code	UC6
Importance	Non-critical
Casual Definition	With the help of the CLI, the Administrator can query the Daemon for a list of available datasets. The Daemon should return a list of exportable datasets, along with their size in the database and the size they would have in exported format..

Name	Remove Dataset
Code	UC7
Importance	Critical
Primary Actor	Administrator
Preconditions	● Administrator is logged in a system where the Maelstrom daemon is running ● Administrator is logged in a system where the Maelstrom CLI is installed ● Administrator uses CLI to remove a dataset
Postconditions	● Daemon removes appropriate table from database

Name	Export Dataset
Code	UC8
Importance	Critical
Primary Actor	Administrator
Preconditions	● Administrator is logged in a system where the Maelstrom daemon is running ● Administrator is logged in a system where the Maelstrom CLI is installed ● Administrator uses CLI to export a dataset
Postconditions	● Daemon queries the database for appropriate table ● Daemon converts result into CSV format ● CSV is return to the Administrator

3.3. Non-functional requirements

3.3.1. Performance efficiency

There are no performance requirements.

3.3.2. Compatibility

The Maelstrom Daemon will be compatible with any OS capable of running Node.js scripts.

The CLI will be able to run as any UNIX executable.

3.3.3. Usability

Only the Administrator will be able to use the CLI to interact with the locally running Maelstrom Daemon.

3.3.4. Reliability

The system must be able to run for long periods of time (months) without crashing. The logics must be fault tolerant and should properly act upon any fail or invalid response from an external system.

3.3.5. Security

The Maelstrom Daemon must only be accessible locally on the system.

3.3.6. Maintainability

The Software Architecture Document will explain where is located the scraper modularity and how the Daemon works with plugins. Plugins are made to be easily changed or replaced.

3.3.7. Portability

The Maelstrom Daemon can run on any OS capable of running Node.js scripts. The goal of the Maelstrom Daemon is not to be portable. The Maelstrom CLI will run on any OS capable of executing UNIX executables.

3.3.8. Design constraints

The whole scraper will be implemented in Typescript. HTTP error codes and request types should be respected on every route of the REST API.

4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

Illustrate (system) UML sequence diagrams (one for each critical scenario), identify system operations and describe operation contracts, one per critical system operation.

You may also use UML state diagrams to describe critical use cases, one state diagram per use case.

Finally, create a UML conceptual class diagram ("domain model") for the system. If the model gets too large, you can use UML package diagrams to provide logical grouppings for the model.