Software design

Application name:

“Intelligent incremental backup system for web servers”

Filip Lukáč,

Marek Kukučka, Ivan Martynenko

1. Introduction

1.1. Purpose of this document

The main purpose of this document is to have complete and detailed design of the information system. It contains all information needed to explain and understand the functionality of the system, as well as, how to implement the system. This document is primarily dedicated for developers. The content in this document covers all requirements from the software requirements document.

1.2. Scope of the document

This document is closely related to the software requirements catalog, so it is required to be familiar with it.
It contains completed and detailed design of implementation of all requirements from the software requirements document. It further specifies external interfaces, overall design of the user environment, including visualization. The implementation proposal is described by diagrams. In addition, there are described which technologies have been used and the final deployment into operation

1.3. Overview of the following chapters

The following chapters deal with the external interface, data model, user interface and its visualizations. The last chapter describes detailed system implementation design.

2. Specification of external interfaces

2.1. The application will communicate with the user using a command line interface (CLI) and it is expected that the application will run on a Linux operating system

2.2. The application will be able to transfer files to and from a remote storage server using transport utility for example SCP or SFTP

2.3. The application will store backups with the .tar extension. For the compression linux build in compression utility will be used

2.4. The application will get started by software demon cron according to crons configuration

3. Data model

File structure

Config files

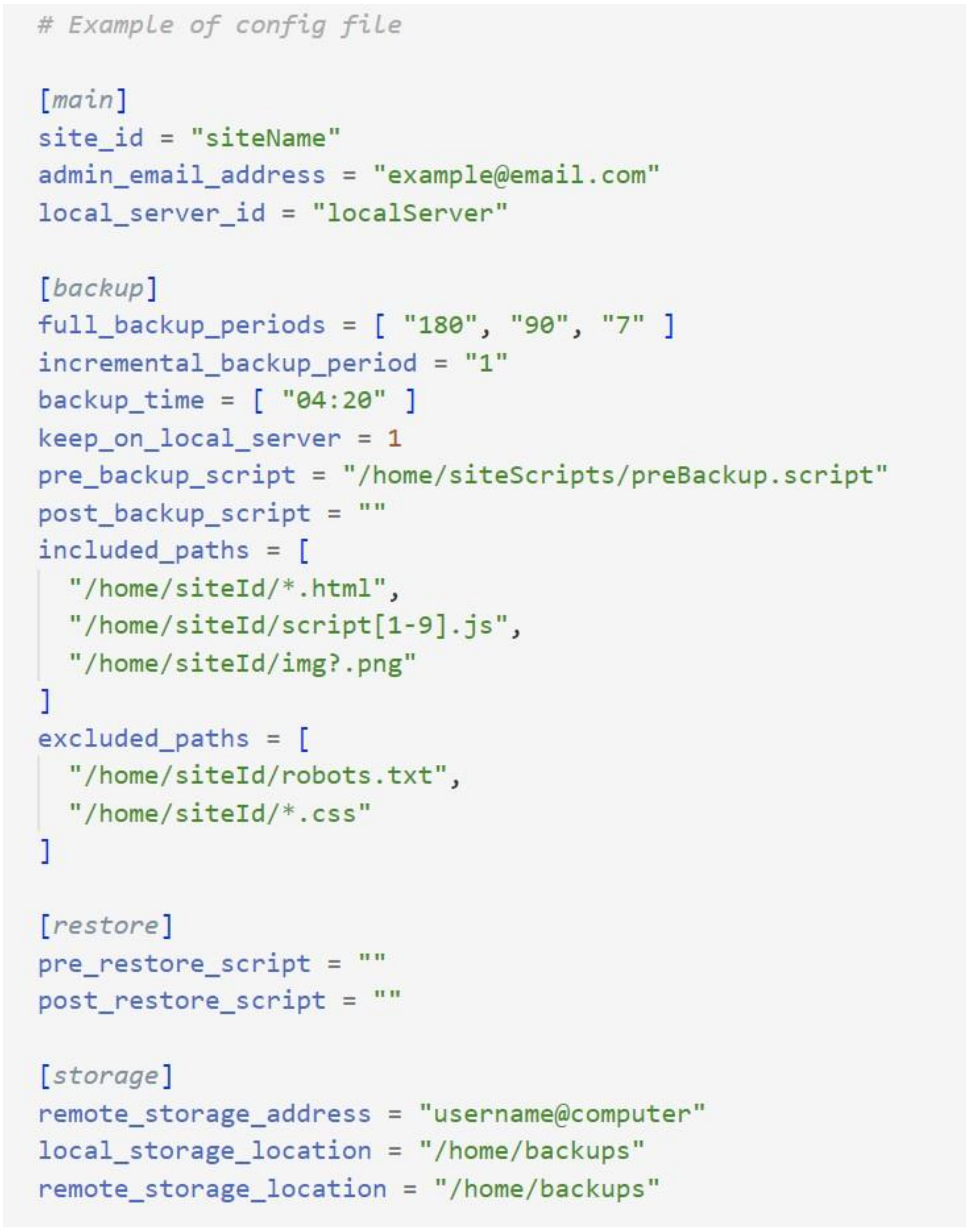
The application must be configurable using configuration files.
Config files will be modified independently of the application by the user using his or her preferred editor. (For example, using vim).
The extension of the configuration file will be .toml and it will look like this:
{configName}.toml

TOML is a file format for configuration files. It is intended to be easy to read and write due to obvious semantics which aim to be "minimal".

The structure of the configuration file will consist of four attribute sections:

- main - with common attributes for configuration
- backup - with attributes for backups configuration
- restore - with attributes for configuring restores
- storage - with attributes responsible for storage

Example of a configuration file:



```
# Example of config file

[main]
site_id = "siteName"
admin_email_address = "example@email.com"
local_server_id = "localServer"

[backup]
full_backup_periods = [ "180", "90", "7" ]
incremental_backup_period = "1"
backup_time = [ "04:20" ]
keep_on_local_server = 1
pre_backup_script = "/home/siteScripts/preBackup.script"
post_backup_script = ""
included_paths = [
  "/home/siteId/*.html",
  "/home/siteId/script[1-9].js",
  "/home/siteId/img?.png"
]
excluded_paths = [
  "/home/siteId/robots.txt",
  "/home/siteId/*.css"
]

[restore]
pre_restore_script = ""
post_restore_script = ""

[storage]
remote_storage_address = "username@computer"
local_storage_location = "/home/backups"
remote_storage_location = "/home/backups"
```

Backups

The configuration and log files will be located in the working directory of the program. The working directory will be called "web-backups" and the full path to it will depend on where the software has been installed or copied to.

Site backups will be stored in a folder with site name. The path to this folder is defined in the configuration file. Also, incremental and full backups will be stored in different directories
The folder will include folders “full-backups” and “incremental-backups”.

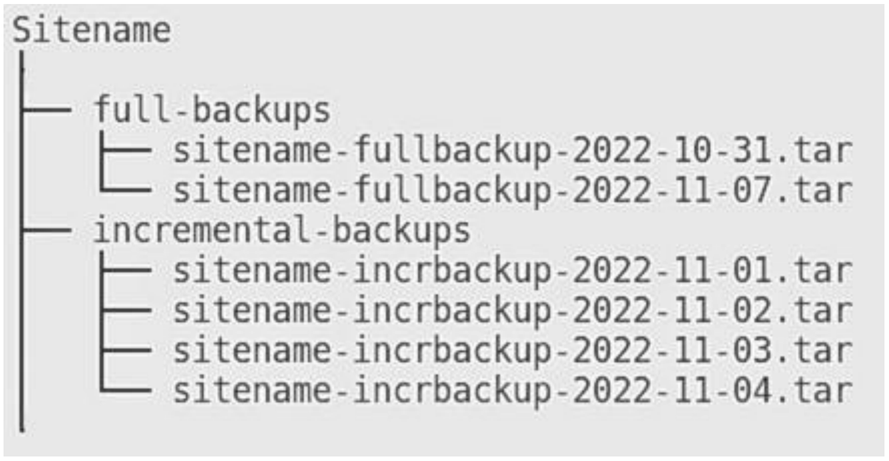
Backups will have the extension.7z

The archive data contain various file system parameters, such as name, timestamps, ownership, file-access permissions, and directory organization.

The names of archives with backups will be as follows:

- sitename-fullbackup-YYYY-MM-DD.7z
- sitename-incrbackup-YYYY-MM-DD.7z

Example of a directory structure where backups will be stored:



```
Sitename
│
├── full-backups
│   ├── sitename-fullbackup-2022-10-31.tar
│   └── sitename-fullbackup-2022-11-07.tar
├── incremental-backups
│   ├── sitename-incrbackup-2022-11-01.tar
│   ├── sitename-incrbackup-2022-11-02.tar
│   ├── sitename-incrbackup-2022-11-03.tar
│   └── sitename-incrbackup-2022-11-04.tar
│
```

Log files

The application will create two types of log files.
General log file where all backup and restore actions will be logged.

Java's native logging capabilities will be used to create log files.

Example of log file names:

- sitename-restore-generallog-YYYY-MM-DD.log
- sitename-restore-additionallog-YYYY-MM-DD.log
- sitename-fullbackup-additionallog-YYYY-MM-DD.log

- sitename-incrementalbackup- additionallog -YYYY-MM-DD.log

Example of an additional log file structure:

```
===================================================================
Starting new log
Log has been started by 'system' user
LocalServer: [machine], OS: [Ubuntu 22.04.1 LTS]
Process: [64 bit], PID: [10816], SessionId: [0]
UTC Time: [01.11.2022 15:37:45]
Culture: [en-US], UI culture: [en-US]

Site ID: 'example'
Action type: [Backup], Backup Type: [Full]
Backup Start Date: [01.11.2022 15:37:45]
Backup Finish Date: [01.11.2022 15:40:51]
Time Taken: [00:03:06]

[01.11.2022 15:37:45] Info Preparing point in full mode
[01.11.2022 15:37:45] Info Run pre_backup script. Ref: '/home/siteScripts/preBackup.script'
[01.11.2022 15:38:12] Info Set local storage location '/home/backups'
[01.11.2022 15:39:49] Info Create backup, name 'fullbackup-2022-11-01'
[01.11.2022 15:40:51] Info Full backup 'fullbackup-2022-11-01' was created. Status SUCCESS

===================================================================
```

Additional log files for each backup and restore performed with detailed information on each affected file.

Additional text files

If a user wants to restore specific files he can specify a text file in the command line, which will contain the paths to the files he needs.

The structure of these text files will only consist of relative paths to specific files, separated by line breaks.

Example of an additional text file structure:

```
/home/site/src/img[1-9].png
/home/site/scripts/script?.js
/home/site/scripts/colors.css
```

4. User interface design

The user interface will be a CLI - command line interface which will be run as "wb [action] [params, …] -flag"
- flags are optional
*[parameter] - The parameter is optional. Usually, there exists a default call for specific action.

List of specific actions:

4.1. list-backups *[site_name] -flag

Output a list of backups. In default, it'll list all backups from the local server. If the site name is set, only its backups will be listed.

Flags:

★ -f	represents a full backup
★ -i	represents an incremental backup
★ -b	lists the backups that are kept on the server

Format of each line in output:
date/time of each backup, type (incremental/full)

4.2. list-sites -flag

Output the list of all configured sites of all local servers that share storage server with this local server. Each list item will include the date and time of the last backup and the location of the site (user specified id of server provided in config).

Format of each line in output:
site_name , local_server_name, storage_server(IP or hostname), date/time of last backup
Flags:

- ★ -e lists only enabled sites
- ★ -d lists only disabled sites

4.3. backup [site_name] -flag

Manually runs the backup of a specific site

Param [site_name] **required**

Flags:

- ★ -i an incremental backup

In default - **Full** backup is **set**

4.4. enable [site_name]

Enables site by adding or removing a **link** to the **sites-enabled** folder (config_name is the same as site_name)

4.5. disable [site_name]

Disables site by adding or removing a **link** to the **sites-enabled** folder

4.6. set-period [period]

Sets the backup period (days)

i.e. set period 120

4.7. set-switch [switch]

Switch - whether to keep backups also on the local server. possible values 0/1 (Bool)

4.8. restore [site_name] [backup_id]

Restores a specific site by writing its name and specifying the **backup_id**

4.9. restore-files [file.txt] [file_path]

Restores specific file(s) (if exists) from a .txt file.

4.10. auto

Checks all configs of locally **enabled sites** and performs backups if needed automatically as requested in site config files

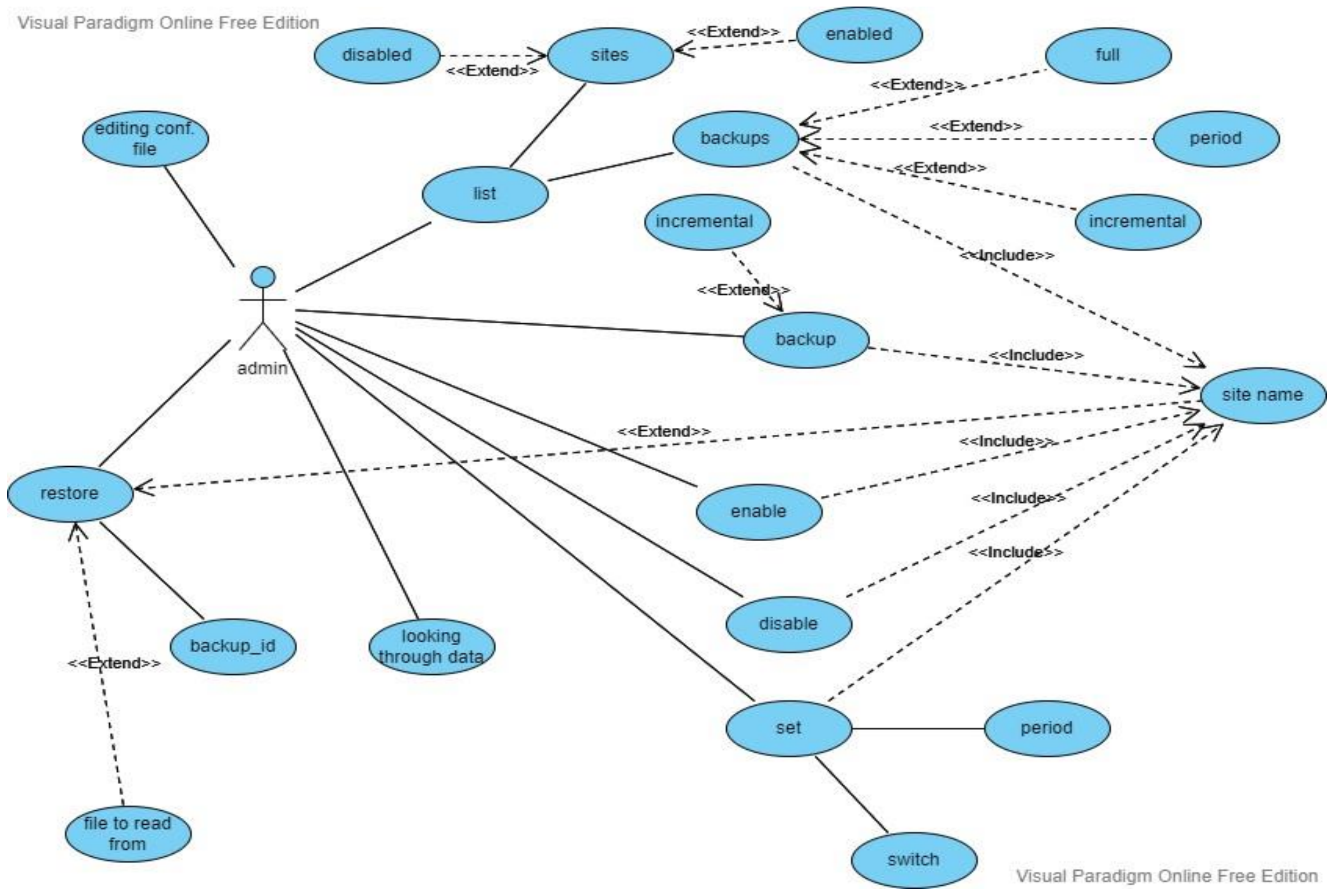
4.11. -h, --help *[action]

In default, the help command shows all possible commands that can be used via CLI. If the parameter **action** is **set**, the usage and description of the **action** will be shown.

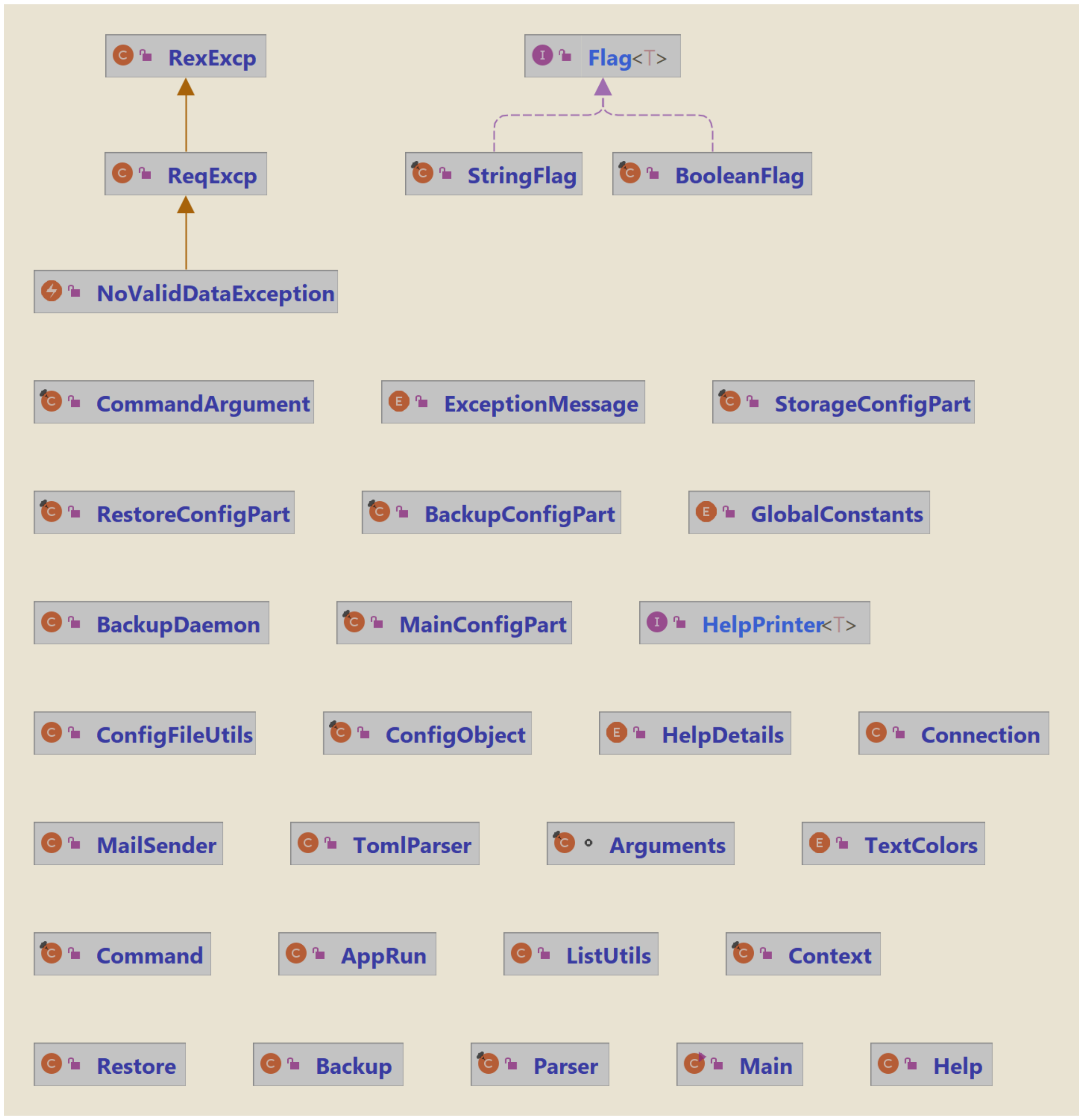
A **logger** will log the **whole process** into a .log file

5. Implementation design

5.1. Application use-case diagram

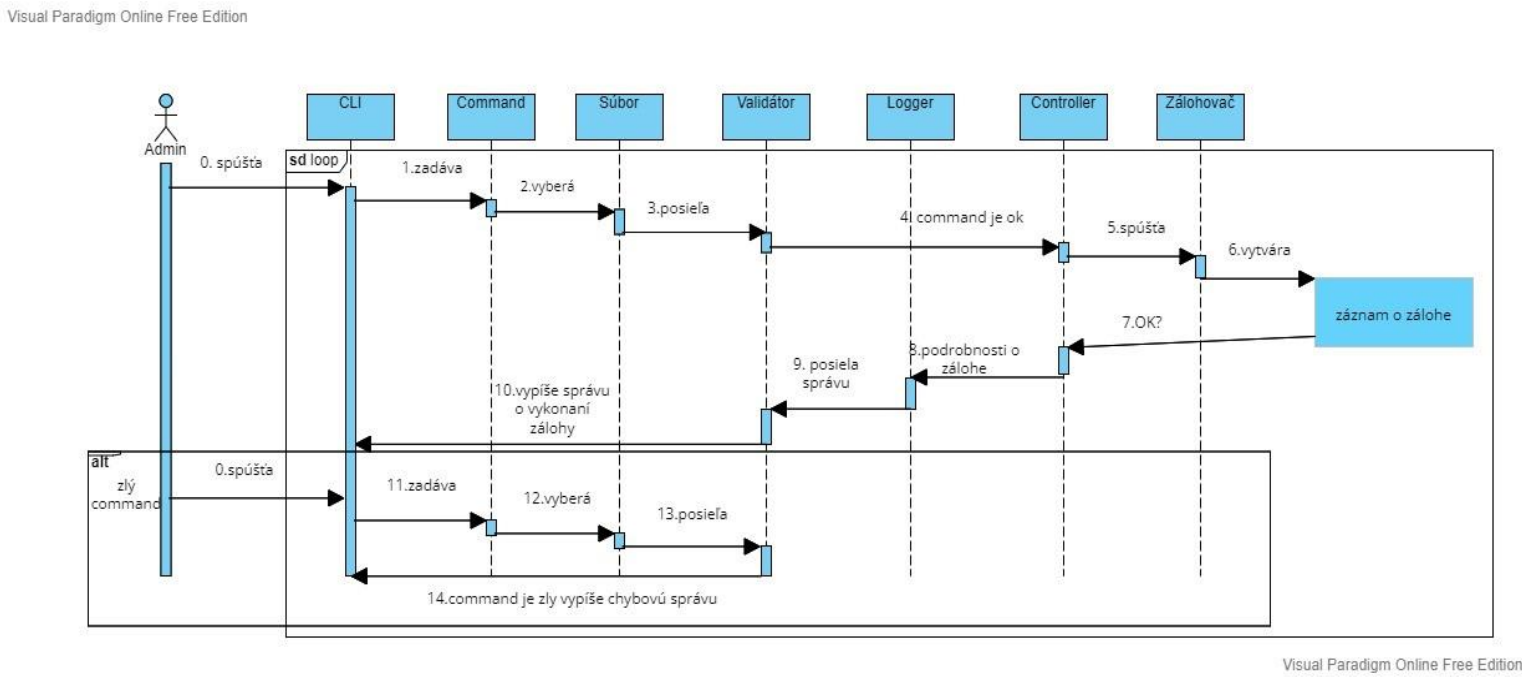


5.2. UML Class diagram

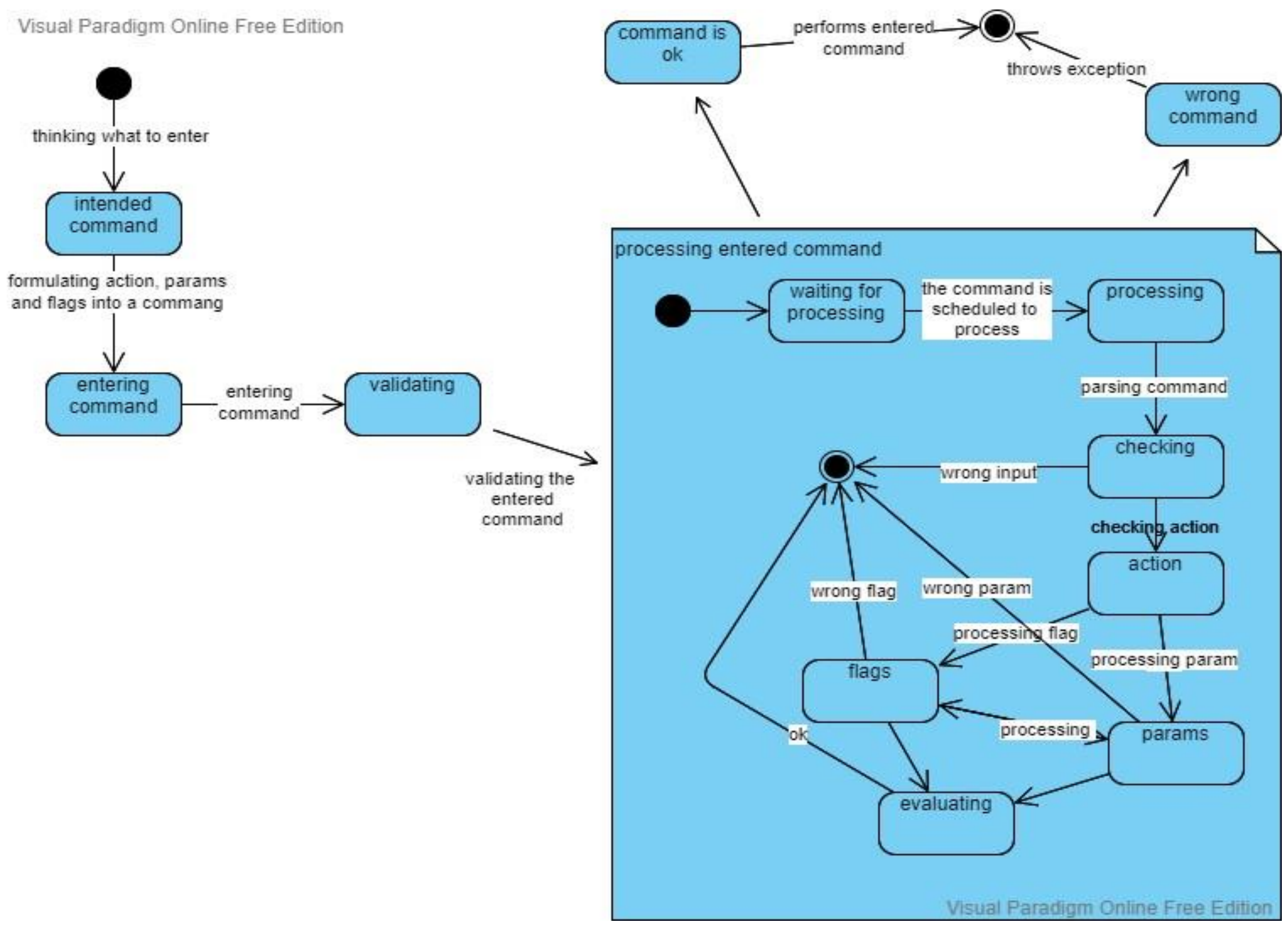


5.3. Sequence diagram

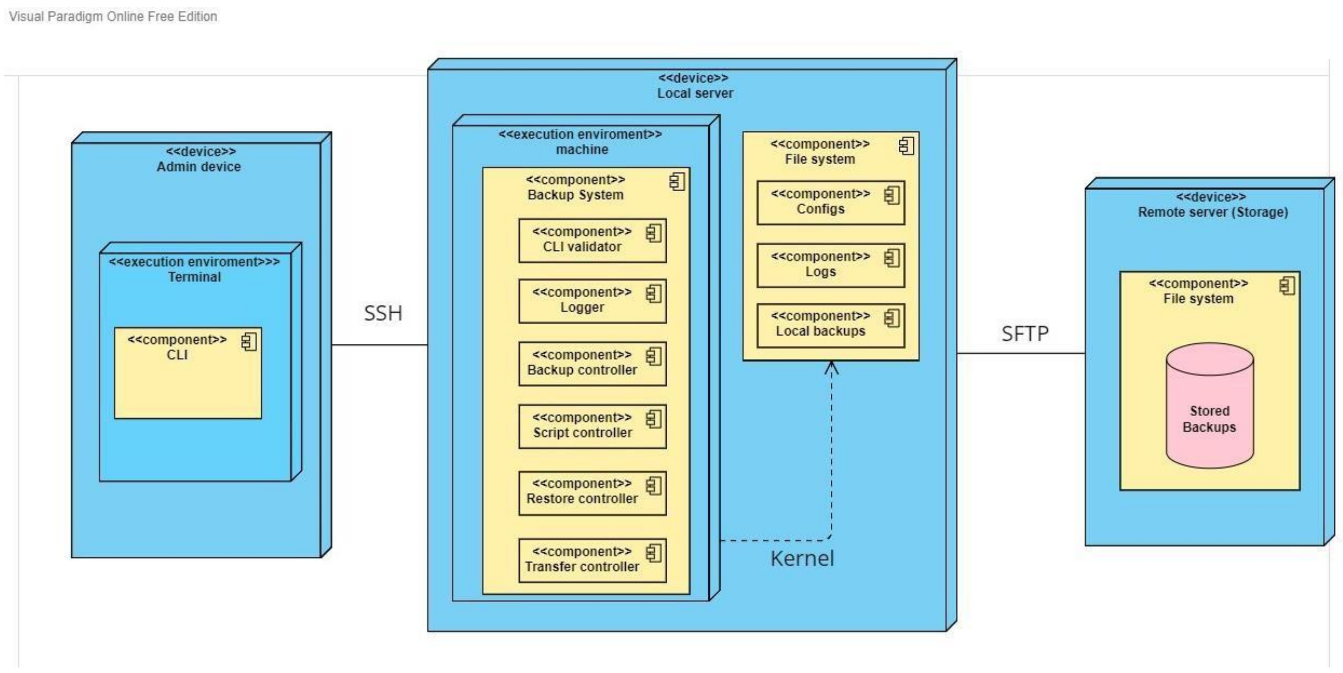
Manual Backup



5.4. State Diagram


Visual Paradigm Online Free Edition
thinking what to enter
intended command
formulating action, params and flags into a commang
entering command
entering command
validating
validating the entered command
command is ok
performs entered command
throws exception
wrong command
processing entered command
waiting for processing
the command is scheduled to process
processing
parsing command
checking
wrong input
checking action
action
wrong flag
wrong param
processing flag
processing param
flags
processing
params
ok
evaluating
Visual Paradigm Online Free Edition

5.5 Deplyoment diagram


Visual Paradigm Online Free Edition
<<device>> Admin device
<<execution enviroment>>> Terminal
<<component>> CLI
SSH
<<device>> Local server
<<execution enviroment>> machine
<<component>> Backup System
<<component>> CLI validator
<<component>> Logger
<<component>> Backup controller
<<component>> Script controller
<<component>> Restore controller
<<component>> Transfer controller
<<component>> File system
<<component>> Configs
<<component>> Logs
<<component>> Local backups
Kernel
SFTP
<<device>> Remote server (Storage)
<<component>> File system
Stored Backups

6.

6.1. List management

Handler for the list command. Handles the listing of any action (see section 4) and specific data provided by the descriptor.
There are two possible actions:

- backups
- sites

In case of bad input, an exception based on error type is thrown.
Possible error types:

- Incorrect flag
- Incorrect parameter

- Duplicities, simultaneously providing more parameters or flags which could not be combined

Output format:

Backups action:

Default:

Site name:

- **Full backups**:
  - *DD-MM-YYYY*
  - *DD-MM-YYYY*
- **Incremental backups:**
  - *DD-MM-YYYY*
  - *DD-MM-YYYY*

  ….

Site name2:

- **Full backups**:
  - *DD-MM-YYYY*
  - *DD-MM-YYYY*
- **Incremental backups:**
  - *DD-MM-YYYY*
  - *DD-MM-YYYY*

  ….

  …

With site Name specified:

- **Full backups**:
  - *DD-MM-YYYY*
  - *DD-MM-YYYY*
- **Incremental backups:**
  - *DD-MM-YYYY*
  - *DD-MM-YYYY*

6.2 backup management

Handler for the backup action.
Automatic and **manual** backups provided by the CLI request are possible.

- Automatic backups are provided based on the **period** specified in the configuration file on the application start. Both incremental and full.
- Manual backups are possible by running the backup action via CLI at any time. Parameter **site_name** is **required**. By adding **-i flag**, the **incremental** backup is being run.
  Incorrect input handling:
  In case of *invalid* **parameter** or **flag**, an exception is being thrown and the user is asked whether to repeat the action by providing the valid arguments or to exit and choose another action. In case the backup fails before it's completion, the stack trace is shown.
  If the backup completes without any issue, success message is shown.

6.3 Restore management

Handler for the restore action.
Only **manual** restore is available and is accessed through the CLI request.
Possible paramters combination:

- **Site_name** with **Backup_id**
- **File.txt** with the flag **-p** and [**file_path**] parameter

Correct inputs:
In case of site_name and backup_id choice, the specific site is being restored and the backup with requested ID is extracted.
Second choice needs to have the file_path to be extracted from specified.

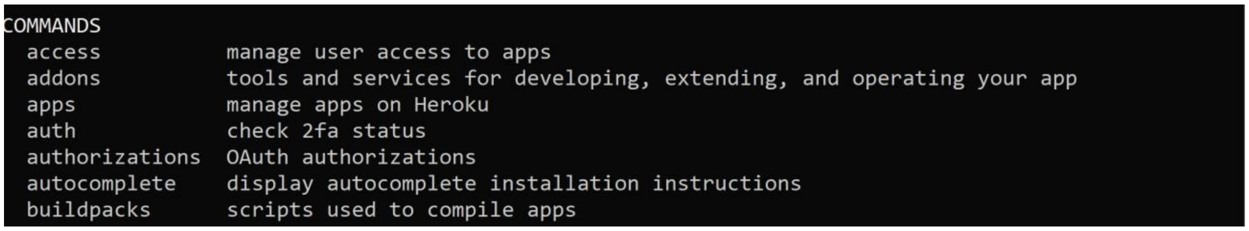
Incorrect input:
In case of *invalid* **parameter** or **flag**, an exception is being thrown and the user is asked whether to repeat the action by providing the valid arguments or to exit and choose another action. In case the restore fails, the stack trace is shown.

6.4 AppRun handler

When starting the program, a message will be displayed in the console with all possible commands and their detailed descriptions.

Commands and command descriptions are synchronised with all added commands using the common-cli library.

Example command output structure and descriptions:



```
COMMANDS
  access          manage user access to apps
  addons          tools and services for developing, extending, and operating your app
  apps            manage apps on Heroku
  auth            check 2fa status
  authorizations  OAuth authorizations
  autocomplete    display autocomplete installation instructions
  buildpacks      scripts used to compile apps
```

6.5 Exception handler

When exceptions are raised while the program is running, they will all be shown in the log file and can also be displayed in the console.

Types of exceptions:

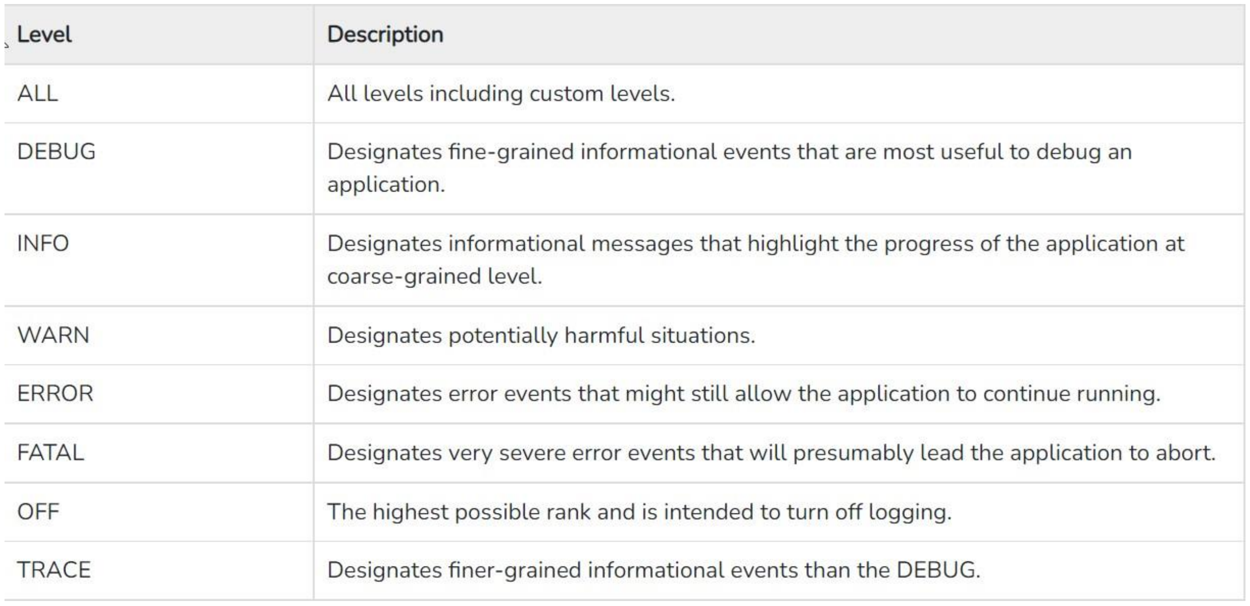
- *File system exceptions*
- *Exceptions during processing cli commands*
- *Exceptions during processing configuration files*
- *Backup creation exceptions*
- *Exceptions during restoration process*
- *Exceptions during connection to a remote server*
- *Standard exceptions during program run*

6.6 Logger

The logger will consist of three logical parts:

- *Logger* - this is where messages will be sent in the application.
- *Appender* - these are the different handlers of the log messages. Each Appender handles some specific destination for log messages. Within our application this will be the ConsoleAppender and the FileAppender.
- *Layout* - formats the log messages.

Logging Levels:



Level	Description
ALL	All levels including custom levels.
DEBUG	Designates fine-grained informational events that are most useful to debug an application.
INFO	Designates informational messages that highlight the progress of the application at coarse-grained level.
WARN	Designates potentially harmful situations.
ERROR	Designates error events that might still allow the application to continue running.
FATAL	Designates very severe error events that will presumably lead the application to abort.
OFF	The highest possible rank and is intended to turn off logging.
TRACE	Designates finer-grained informational events than the DEBUG.

6.7 Parsing options

Commands will be validated using the common-cli library.
All work with commands and options will be divided into three parts.

Definition phase
In the definition phase, we define the options that the application can accept and act on accordingly. common-cli provides the Options class, which is a container for Option objects.

```
// create Options object
Options options = new Options();

// add a option
options.addOption("a", false, "add two numbers");
```

Parsing phase
In the parsing phase, we analyse the parameters passed in using command line arguments after the parser instance is created.

```
//Create a parser
CommandLineParser parser = new DefaultParser();

//parse the options passed as command line arguments
CommandLine cmd = parser.parse( options, args);
```

Interrogation phase
In the interrogation phase, we check whether a particular option is present or not and process the command accordingly.

```
//hasOptions checks if option is present or not
if(cmd.hasOption("a")) {
   // add the two numbers
} else if(cmd.hasOption("m")) {
   // multiply the two numbers
}
```

6.8 Enable

Handler for the enable action, enables a site through a CLI request by adding its link to the sites_enabled folder
Command is written in format: wb enable [site_name]

Parameter [site_name] is required, if there’s a *different* amount of parameters than *one* an exception is thrown

If the command is written correctly the application will check if the parameter [site_name] is linked in the sites_available folder.

In the case of the parameter *not* being linked in the sites_available folder an exception will be thrown.

The application will then add a link to the site in the sites_enabled folder.

In the case of the site *already being linked* in the sites_enabled folder the action will do nothing.

6.9 Disable

Handler for the disable action, disables a site through a CLI request by removing its link in the sites_enabled folder

Command is written in format: wb disable [site_name]

Parameter [site_name] is required, if there’s a different amount of parameters than one an exception is thrown

If the command is written correctly the application will remove the link to the site in the sites_enabled folder.

In the case of the parameter *not being linked* in the sites_enabled folder the action will do nothing, and the user will be notified

6.10 Send email message

Email message will only be sent when a backup that hasn’t been started manually runs into a problem.

After a problem during a backup occurs the application will report the problem in a log file.

Then the application automatically uses the Linux mail command to send an email to the administrator with the same text that was written to the log file. The email address will be found in the config file

After the email is sent the application will continue with the backuping as if no problem occurred.

6.11 Edit config files

The user won’t be able to change the config files within the application. When the user wants to edit the files they can do so in two different ways:

- By manually opening the file in the user’s preferred editor
- By opening the file in the CLI using vim or nano file editor