8Lab Solutions - Project "Soldino"

Developer manual

Version	1.0.0
Approval	Samuele Giuliano Piazzetta
Drafting	Francesco Donè
	Sara Feltrin
Check	Paolo Pozzan
State	Approvato
Use	Esterno
Adressed to	Red Babel
	8Lab Solutions
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Description
Developer manual made by *8Labs Solutions* for the making of the project *Soldino*.

8labsolutions@gmail.com

Changelog

Version	Date	Name	Role	Description
2.0.1	2019-03-20	Federico Bicciato	**RUOLO**	Structure of the document created.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Manual contents

This document is the developer manual of he project *Soldino*, developed by the team *8Lab Solutions* for the proponent *Red Babel*.
Within the manual you can find:

- the technologies used for the development;
- the software tools used and suggested;
- the software architecture;
- the architectural and design pattern used;
- the functionalities provided by *Soldino*.

1.2 Purpose of the manual

The contents of the manual are intended to help the developers who decide to further develop or maintain *Soldino*. Everything described here can help the developer to fully and deeply understand the design, use and features of the application, so that it can be modified and improved with ease.
Many technologies, tools and languages are used to build the app: these are only briefly explained in their parts that cover the application domain. Additional references can be found in the "Reference" section.

1.3 Purpose of the product

The platform *Soldino* is a DApp accessible on a web browser as a client interface and the plugin Metamask as a virtual wallet.
The main functionality of *Soldino* is trading goods and services online. Since the platform's backend is coded on the Ethereum blockchain, it provides more security and transparency than the traditional e-commerce websites.
The platform is built to be managed by the government. The currency used in it is called Cubit, and it's a ERC20 compliant fork of Ether, minted and managed by the government itself.

1.4 References

1.4.1 Normative

- **none**

1.4.2 Informative

- **Ganache** https://trufflframework.com/ganache
- **Git** https://it.atlassian.com/git
- **Node.js** https://nodejs.org/it/
- **Node Package Manager** https://www.npmjs.com/get-npm

- **Surge.sh** https://surge.sh/help/getting-started-with-surge
- **Truffle** https://trufflframework.com/truffle

2 Setup

2.1 Requirements

In this section we describe all the requirements needed.

2.1.1 Browser

Soldino is accessible through a web interface. The currently most recent versions of the following broswers are supported:

- **Mozilla Firefox**: version 64 or later;
- **Google Chrome** version 71 or later.

2.1.2 Tools

The following tools are needed:

- `Git`: the famous control version system: Soldino is hosted on GitHub;
- **Node.js**: you need it to run commands and as a Truffle requirement;
- **Truffle**: you need it to write and deploy contracts with ease;
- **Ganache**: you need it to put up a local Ethereum network and check transactions on it;
- **Metamask**: a browser plugin used as a virtual wallet.
- **Surge.sh**: the web platform chosen to host the website interface of Soldino.

2.1.3 Dependencies

Soldino depends on many different packages, some for use and others for development.
All these packages are located in the file `package.json` which is in the root folder of the project.
The packages required to execute the software *Soldino* are listed below.

Table 2.1.1: Packages required for software usage

Software	Version
react-text-mask	≥5.4.4
commondir	≥1.0.1
history	≥4.7.2
prop-types	≥15.7.2
react	≥16.8.3
react-dom	≥16.8.3
react-number-format	≥4.0.6
react-redux	≥6.0.1

Table 2.1.1: Packages required for software usage

Software	Version
react-router	≥4.3.1
react-router-dom	≥4.3.1
react-router-redux	≥4.0.8
react-scripts	≥2.1.8
redux	≥4.0.1
redux-thunk	≥2.3.0
web3	1.0.0-beta.37

Other packages, listed below, are required for the development.

Table 2.1.2: Packages required for development

Software	Version
eslint	5.12.0
eslint-config-airbnb	≥17.1.0
eslint-loader	≥2.1.2
eslint-plugin-import	≥2.16.0
eslint-plugin-jsx-a11y	≥6.2.1
pre-commit	≥1.2.2
truffle-contract	≥4.0.6

2.2 Installing

2.2.1 Browser

The first thing is to have your browser installed. You can get the latest chrome version here.

2.2.2 Git

Type on shell: `sudo apt install git`.

2.2.3 Node

Install Node.js. Digit on the shell the following commands:

1. `curl -sL https://deb.nodesource.com/setup_11.x | sudo -E bash -`
2. `sudo apt install -y nodejs`

3. check that `node` have been installed correctly with `node -v`.

There is no need to install npm separately, since it is automatically installed with Node.

2.2.4 Truffle

Third thing: install Truffle.
Truffle requirements are:

- an OS among Linux, Windows and MacOS (prefer Linux);
- NodeJS v8.9.4 or later (we picked version 11);
- Node Package Manager (npm).

You can then install Truffle with the command:

```
npm install -g truffle
```

2.2.5 Ganache

Fourth step: installing Ganache. There are three step to install Ganache:

1. you can download the Ganache executable at this link, clicking on the download button.;
2. give the permissions to make the Ganache file executable. This can be done on Linux with the command `chmod +x path-of-the-appimage/ganache-1.3.0-x86_64.AppImage`;
3. eventually, run it double clicking on the icon.

2.2.6 MetaMask

MetaMask comes in the form of browser extension. You can add it to your browser this way:

- Chrome: https://chrome.google.com/webstore/search/metamask?hl=it;
- Firefox: https://addons.mozilla.org/it/firefox/addon/ether-metamask/?src=search.

2.2.7 Surge

Install Surge.sh with the shell command

```
npm install -g surge.
```

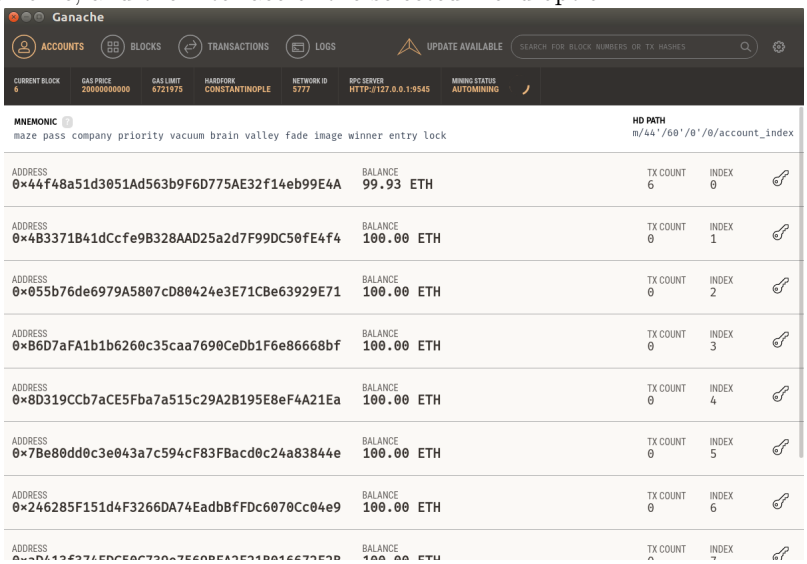
2.3 Configuration

This section shows how to configure your work environment, so that it's the same as ours, in order to minimize the number and entity of troubles you will occur in.
Make sure to configure the tools in order. In particular, Truffle requires Ganache open and configured to execute successfully.

2.3.1 Cloning the repo

Clone the Soldino repository on GitHub: open the shell, move to the directory where you want Soldino to be, then use `git clone https://github.com/8LabSolutions/Soldino-PoC`. **LINK DA MODIFICARE CON IL LINK ALLA REPOSITORY FINALE**

Figure 2.3.1: Ganache UI: from top to bottom you can see the menu bar, the current configuration, the mnemonic, and the interface of the selected menu option



2.3.2 Ganache

Go to the folder where you've put Ganache, and open it with double click.

Click then on the cog at the top left corner to access Ganache settings. Make sure to match the follow settings (most of which are defined in `truffle-config.js`) on each respective window:

- Server:
  - hostname: `127.0.0.1`;
  - port number: `9545`;
  - network it: any (you can keep the default one).
- Account & keys:
  - nothing to configure here, but have a look at the **mnemonic**: it will help us later.

2.3.3 Truffle

The configuration of Truffle is defined in the file `truffle-config.js` in the root directory of Soldino. On your shell type the following shell commands:

- `truffle console`
  opens the truffle environment in the shell under the configuration defined in truffle-config.js. From now on, every command is executed in the truffle environment, from which you can exit double typing `ctrl + C`.
- `compile`
  to compile the contracts: these are compiled in in a .json format (which enables interaction with the frontend) and put into the folder defined in `truffle-config.js` at `contracts_build_directory` (in our case the location is `./src/contracts_build`)

- `migrate --network development`
  puts the contracts runnning on the blockchain.

2.3.4 MetaMask

First, it's necessary to create an account.

- open MetaMask on your browser
- select get started
- select import wallet
- use the seed frase (AKA the mnemonic) copy pasting the one
- from ganache and put your password

Now your account is up and synchronized with Ganache.

You also have to connect the wallet to Ganache network. Ganache settings are exposed in the Ganache UI just above the mnemonic. Let's synchronize MetaMask to the same network. On the top right corner there is a drop-down menu to select the network. Select `Custom RPC` to set your own local network.
You are now in the MetaMask advanced settings screen. In the field `Net Network` click on `Show Advanced Options` to open the form, then insert these data:

1. **New RPC URL**: http://127.0.0.1:9545 (the port number matches the one in truffle-config.js);
2. **Nickname**: the name you wanna give to your network;
3. **save** when you're done.

MetaMask is now connected to Ganache.

To enable transactions on your local test network, you can find free ether for testing networks on this site: https://faucet.metamask.io/.

2.4 Running

Now that you have all the required software installed and configured, it's time to get it up running.
All you have to do is moving to the root directory of Soldino and prompt

```
npm run start
```

This will open the website on your browser at the address `http://127.0.0.1:9545` and allow you to explore it.

2.5 Deploying

This part shows how to deploy you contracts with Truffle and have them up running on Soldino.

3 React

3.1 Overview

3.2 Components

3.2.1 Presentational

3.2.2 Containers

3.3 Routing

4 Redux

4.1 Overview

4.2 Unidirectional flux

4.3 Components

4.4 How to extend

5 Web3

5.1 Overview

5.2 Collaborations

6 Solidity

6.1 Overview

6.2 Contracts

6.3 Collaborations

6.4 How to extend

7 Testing

This chapter shows how to

- test the javascript and solidity code automatically;
- check if the code syntax is complied to the rules given.

A Glossary

A.1 A

A.2 B

A.3 C