Documentation

Introduction

Welcome to the documentation for our web application designed to compare the earnings percentage of various cryptocurrencies across Binance,, and Crypto.com. This project serves as a dummy project developed during an internship and showcases the technical skills applied to explore the dynamic world of cryptocurrencies.

Our web application fills a significant gap in the market by providing a consolidated platform for users to compare cryptocurrency earnings percentages across multiple platforms. As of our knowledge cutoff date, there is no existing software that offers this specific functionality, making our web application unique and valuable for users seeking to make informed investment decisions.

By leveraging Java as the primary programming language and utilizing JSP, HTML, CSS, Spring, and JavaScript, we have built a feature-rich web application that offers a seamless user experience and real-time access to earnings data. Through our application, users can conveniently analyze and compare earnings percentages from Binance, and Crypto.com, allowing them to identify potential opportunities and optimize their investment strategies.

The documentation will guide you through the installation, setup, and usage of the web application. It will provide detailed instructions, explain the available features, and offer insights into the underlying technologies. We have strived to adopt industry-standard practices to ensure reliability, scalability, and a user-friendly interface.

We appreciate your interest in our web application and believe it will serve as a valuable tool for individuals navigating the cryptocurrency market. The documentation aims to assist you in understanding the functionalities and leveraging the capabilities of our unique web application effectively.

Remember to update the information if, in the future, similar software becomes available, to accurately represent the competitive landscape.

How to use?

Software Versions

- Java (JDK 20)
- Apache Maven 3.9.3
- Apache Tomcat 10.1.10
- Spring 3.0.4
- Selenium 3.141.59
- org.json 20230227
- ChromeDriver 113.0.5672.63

This web applications needs 4 different inputs:

1. Amount- The first input is the money that the user wants do invest.
2. Crypto - Type of crypto (e.g. ADA or Cardano).
3. Duration - Type of duration for earning. Both sites have Flexible, 1 month and 3 months.
4. Lockup - This input is only for the crypto.com website and it has 3 variants, less then 4000, 4000 to 40000 and 40000 or more.

Once inserted the 4 inputs, the user have to click the "compare" button and the web application will procede to execute the code and after some time, the results will be shown in the table.

How does the web application works?

The web application was developed using IntelliJ, leveraging the webapp archetype. This approach automatically generates the necessary project structure, including the essential files and folders required for a web application. One of the key components in the project is the *_pom.xml_ *file, which plays a crucial role in managing dependencies and controlling the build process through Apache Maven.

Let's deep dive into the classes.

CryptoValueFetcher

The main purpose of this class is to get the market value of a specific crypto that is received as an argument in the abbreviation form.

This class will use the CoinGeckoAPI to get the list of all coins and the market value of the crypto passed as an argument.

The class is separated in 2 essencial methods:

The first one is getCryptoIdByAbbreviation() and this method was created because this API does not have an endpoint that can take the market value using only its abbreviation, so this method was created using the endpoint for the list of cryptos, and transform the abbreviation into the ID.

Once we have the ID, we get to the second method getCryptoValue(), that will use the API endpoint and parse the market value into a JSON object. In the end will return the market value of the crypto.

```
private static final String COIN_LIST_API =
"https://api.coingecko.com/api/v3/coins/list";
private static final String VALUE_API =
"https://api.coingecko.com/api/v3/simple/price?ids=%s&vs_currencies=usd";
```

Calculator

This class will handle all the math in the web applications, it has 3 arguments, the first one will be the APR value (in the code is shown as "String text"), the second and third arguments are the amount value and the cryptocurrency inputed by the user.

It starts with executing the CryptoValueFetcher class using the cryptocurrency as the argument and then proceeds to do calculations so that we can get the Annual and Weekly

Value in $ and the Annual Value in the cryptocurrency chosen.

Not as important but it also uses the Java DecimalFormat library.

```
public static String[] calculations(String text, String amountValue,String crypto)
throws IOException {

        double mc = CryptoValueFetcher.getCryptoValue(crypto);
        DecimalFormat decimalFormat = new DecimalFormat("#.#####");
```

Binance and CRO WebDriving Classes

These 2 classes are the key for our web application. They both receive all the inputs mentioned in the "How to use" section.

Using Selenium Web Driving library, the main objectives was to Drive through both websites and try to return the APR value of each one, to do this it is necessary take a list of steps.

- Accept cookies and press any button or agreement so that we can stay in the main website.
- Look for the specific crypto
- Look for the specific Duration
- In CRO case look for the LockUp inputed.

Once all the necessary steps are completed, the class retrieves the APR value as a String. Subsequently, it invokes the *_Calculator_* class to perform further calculations and processing based on the extracted APR value. After concluding the calculations it will return a String array containing the four numbers that are intended to be displayed on the website, (Reminder that both classes will return 4 numbers, so in the end it will be 8).

```
System.setProperty("webdriver.chrome.driver", "C:\\Program Files\\Apache Software
Foundation\\Tomcat 10.1\\webapps\\Driver\\chromedriver.exe");

ChromeOptions options = new ChromeOptions();
options.addArguments("--headless");

//Code here

WebDriver driver = new ChromeDriver(options);
driver.get("https://crypto.com/eea/earn");
WebElement amount = driver.findElement(By.className("css-105r9ql"));
        amount.click();
        amount.clear();
        amount.sendKeys(amountValue);

//More code


String[] Calculators = Calculator.calculations(text, amountValue, cryptoName);
        String ganhoAnual = Calculators[0];
```

```
        String weekly = Calculators[1];
        String adaAnual = Calculators[2];
        driver.quit();
        return new String[]{text, ganhoAnual, weekly, adaAnual};
```

JSP and Spring

Like it was mentioned before, this application utilizes both JSP and Spring, JSP as the frontend using HTML CSS and JavaScript in it, and uses the Spring as a controller of the APP.

Index Controller

This is the class that will control the application, it is a simple task because it only controls one step that is when the Compare button in the jsp file is clicked it will execute the program and display it.

```
@Controller
public class IndexController {

    @Autowired
    Executer executer;

    @GetMapping (value="/")
    public ModelAndView index(){
        ModelAndView mav = new ModelAndView("index");
        return mav;
    }

    @PostMapping("/compare")
    public ModelAndView compare(@RequestParam("amount") String amount,
@RequestParam("crypto") String crypto,
                                @RequestParam("duration") String duration,
@RequestParam ("lock") String lock) throws InterruptedException, IOException {
        ModelAndView mav = new ModelAndView("index");
        String[] results = executer.execute(crypto, amount, duration, lock);
        String APRCRO = results[0];
        String AnualCRO = results[1];
        String SemanalCRO = results[2];
        String anualCryptoCRO = results[3];
```

As you can see, it has 2 views and one is this initial one and the other is the after the button was clicked.

```
<div class="input-container">
    <form action="/compare" method="post">
        <input type="text" name="amount" placeholder="Enter amount" />
        <!-- - more inputs here -->
        <input type="submit" id= "#compare" name="compare" value="Compare" />
    </form>
</div>
```