CryptoNat0r

201509718 Christian Luke Pedersen, E

201509378 Niklas Meyer Møller Sørensen, IKT

18. maj 2018

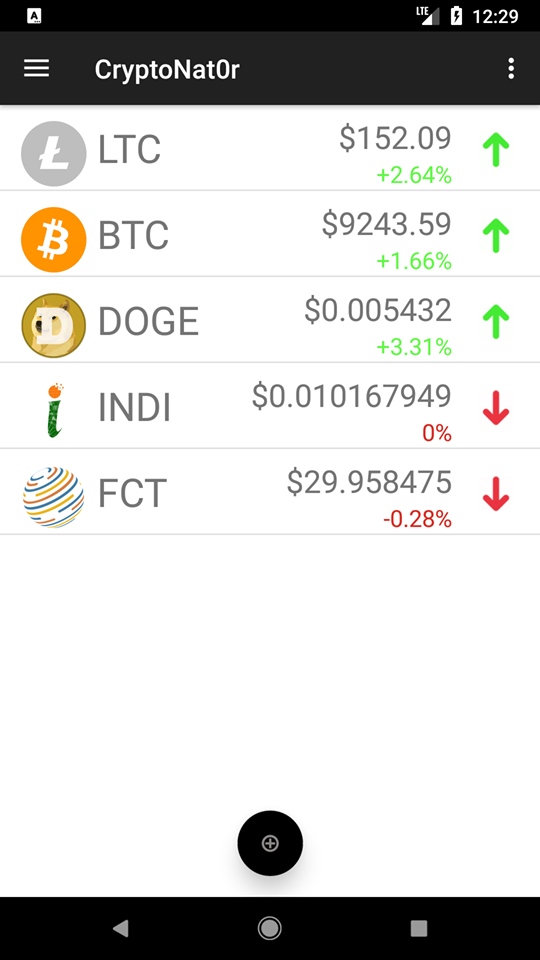


Table of Contents

[App vision 3](#_Toc514410468)

[Personal Vision 3](#_Toc514410469)

[Context 4](#_Toc514410470)

[Requirements specification 5](#_Toc514410471)

[UpdatingService overview 9](#_Toc514410472)

[Fetch coin list 9](#_Toc514410473)

[Fetch details 10](#_Toc514410474)

[Fetch historical data 11](#_Toc514410475)

[Conclusion 14](#_Toc514410476)

[List of known bugs, problems or future development ideas 14](#_Toc514410477)

[Work plan 14](#_Toc514410478)

# App vision

CryptoNat0r is an app which aims to make investments in cryptocurrencies easier for everyone. CryptoNat0r makes it possible for anyone to subscribe and keep track of their favorite cryptocurrencies. Subscribed currencies are shown in the main screen of the application along with their current rate and 24-hour percent change. If the user wants to view more detailed information about their favorite cryptocurrency, they can tap on it and the app will take them to a screen with more detailed information.

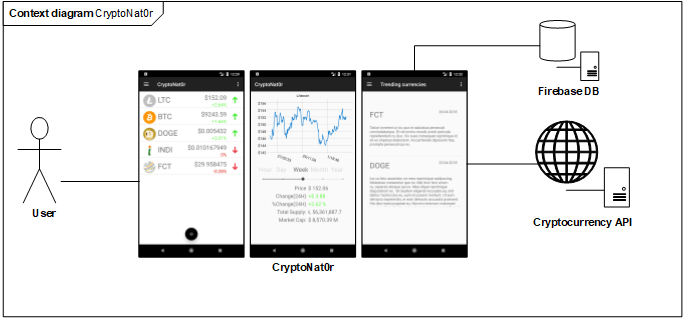
We also wish to make investing in cryptocurrency easier. That’s why the app has a screen with trending cryptocurrencies. These are typical popular, upcoming or highly promising coins recommended by our experts.

# Personal Vision

We think an app displaying various information about cryptocurrencies will give us the opportunity to explore different APIs and how to use them. In the long run we hope to create a popular app that can be of use for people interested in cryptocurrencies.

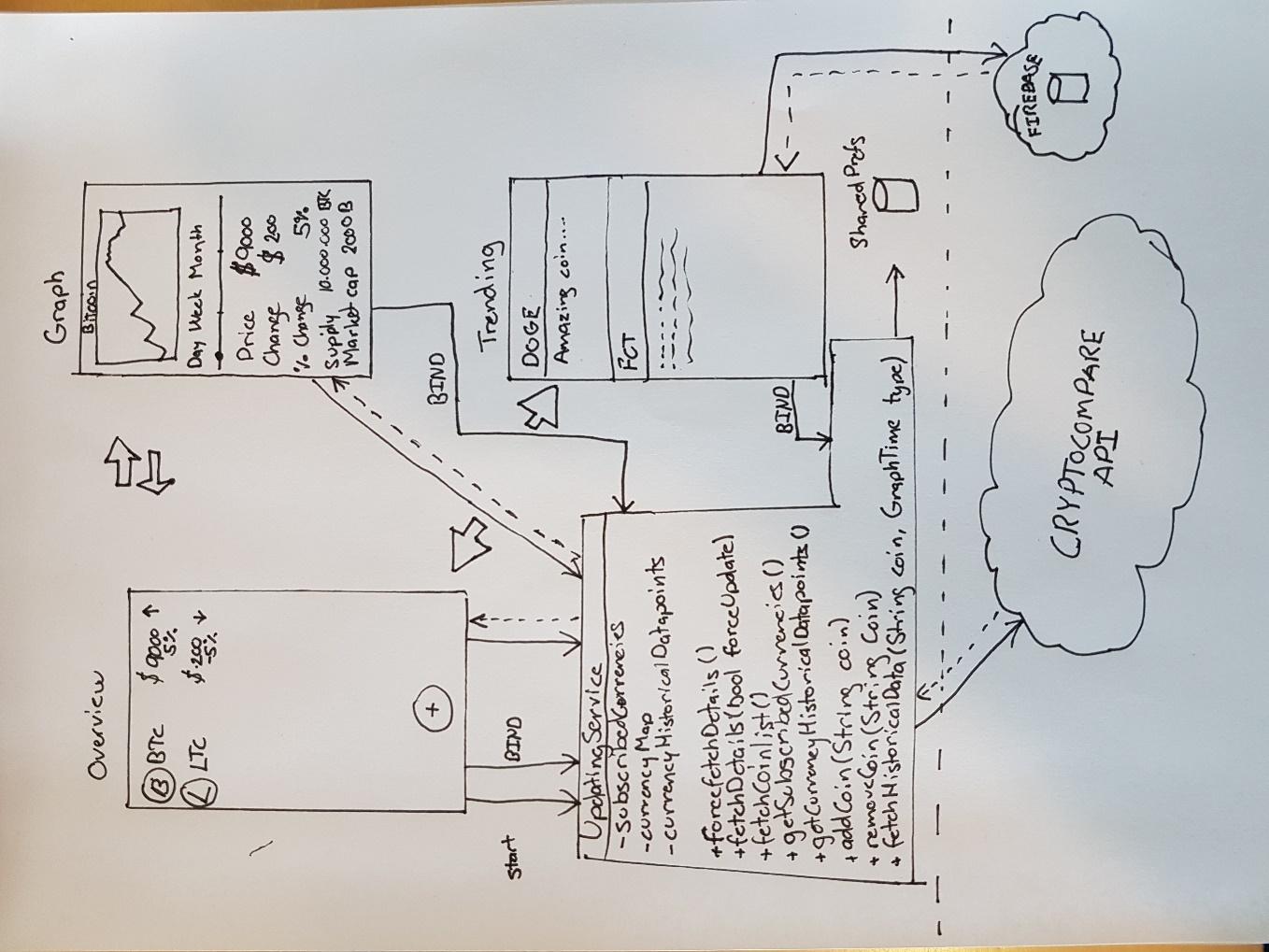
# Context

The context our app is used in is shown in Figure 1. A more detailed diagram is shown in Figure 2.



*Figure 1. Context diagram for CryptoNat0r.*

The user interacts with the app CryptoNat0r through an android phone. The app interacts with a Firebase database and the CryptoCompare API[[1]](#footnote-1).



*Figure 2. Detailed context diagram for the curious reader.*

# Requirements specification

Below is an overview of the requirements for CryptoNat0r described as use cases. Most of the requirements were made in our synopsis, but we added a few more as we had some interesting features we wanted to implement as well.

|  |  |
| --- | --- |
| **Use Case #1: View current rates** | |
| Actors | The user |
| Description | User wants to view the current rates of the subscribed currencies. |
| Preconditions | 1. User is on the frontpage. 2. User has run through Use Case #2: Add coin to subscribed currencies |
| Flow of events |  |
| Postconditions | User has viewed the current rates of the subscribed currencies. |
| Evt. extensions (undtagelser) | None. |

|  |  |
| --- | --- |
| **Use Case #2: Add currency to subscribed currencies.** | |
| Actors | The user |
| Description | User wants to subscribe to a new currency to show it on the front page. |
| Preconditions | 1. User is on the frontpage. |
| Flow of events | 1. User clicks the “+”-sign at the bottom of the screen. 2. User enters the name of the currency he wishes to subscribe to. 3. User presses “Add”. |
| Postconditions | 1. User has subscribed to the currency he entered. |
| Evt. extensions (undtagelser) | **[Currency does not exist]**   1. User is met with an error message. 2. User proceeds from step 1 in main flow. |

|  |  |
| --- | --- |
| **Use Case #3: View details of a subscribed currency.** | |
| Actors | The user |
| Description | User wants to view details of a subscribed currency. |
| Preconditions | 1. User is on the frontpage. 2. User has run through Use Case #2: Add currency to subscribed currencies |
| Flow of events | 1. User clicks on the currency which he wants to view details for |
| Postconditions | User has been shown the details of a subscribed currency. |
| Evt. extensions (undtagelser) |  |

|  |  |
| --- | --- |
| **Use Case #4: View which currencies are trending** | |
| Actors | The user |
| Description | User wants to view which currencies are trending |
| Preconditions | 1. User is on the frontpage. |
| Flow of events | 1. User opens the menu drawer 2. User navigates to trending currencies by pressing the trending currencies menu item. |
| Postconditions | User has been shown information about the current trending currencies. |
| Evt. extensions (undtagelser) |  |

|  |  |
| --- | --- |
| **Use Case #5: Subscribe to Trending Currency** | |
| Actors | The user |
| Description | User wants to subscribe to a Trending Currency |
| Preconditions | 1. User has just completed use case #4: View which currencies are trending |
| Flow of events | 1. User long presses on a Trending Currency, and clicks “Subscribe”. |
| Postconditions | User has subscribed to a Trending Currency. |
| Evt. extensions (undtagelser) |  |

|  |  |
| --- | --- |
| **Use Case #6: Force update data for subscribed currencies by using Swipe-To-Refresh** | |
| Actors | The user |
| Description | User wants to force update data for subscribed currencies by using Swipe-To-Refresh |
| Preconditions | 1. User is on the frontpage. |
| Flow of events | 1. User swipes down on the subscribed currencies list to update prices |
| Postconditions | All data for subscribed currencies has been updated |
| Evt. extensions (undtagelser) |  |

|  |  |
| --- | --- |
| **Use Case #7: Force update data for subscribed currencies by pressing update-button** | |
| Actors | The user |
| Description | User wants to force update data subscribed currencies by using the update-button |
| Preconditions | 1. User is on the frontpage. |
| Flow of events | 1. User opens the context menu 2. User presses Refresh |
| Postconditions | All data for subscribed currencies has been updated |
| Evt. extensions (undtagelser) |  |

|  |  |
| --- | --- |
| **Use Case #8: Add price watch to a subscribed currency** | |
| Actors | The user |
| Description | User wants to add price watch to a subscribed currency to get a notification when the currency reaches a set price |
| Preconditions | * + - 1. User is on the frontpage |
| Flow of events | 1. User long presses a subscribed currency 2. User presses “Add watch” 3. User enters a price in the dialog 4. User presses OK |
| Postconditions | A price watch has been setup which creates a notification if the currency reaches the set price. |
| Evt. extensions (undtagelser) |  |

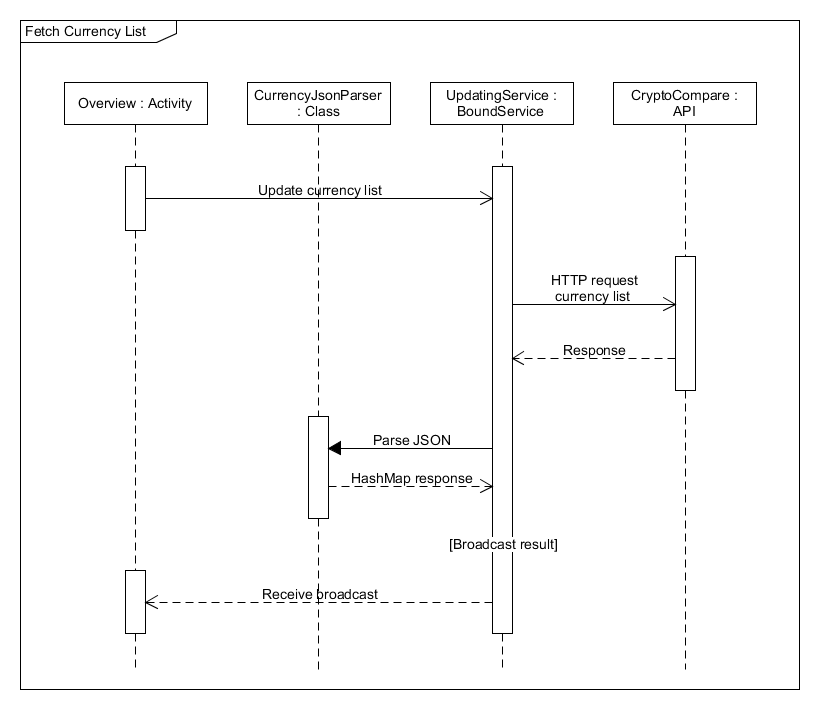
# UpdatingService overview

The UpdatingService is a background service in our app. It’s the most important aspect of the app because it keeps track of all subscribed currencies. It’s also responsible for retrieving the latest rates and the historical data shown on the graphs in the GraphActivity and storing this information correctly.

All the other activities use the UpdatingService to some extend by binding to it.

## Fetch coin list

After having started the UpdatingService the first thing it does is fetch the coin list. The coin list is a HashMap with the currency name as the key and name, short-name and image URL as the value. The HasMap is used when the user adds a subscription to a currency. We check if the currency exists in the HashMap and if it does we add it to the list of subscribed currencies. A sequence diagram describing fetching the coin list is shown in Figure 2.

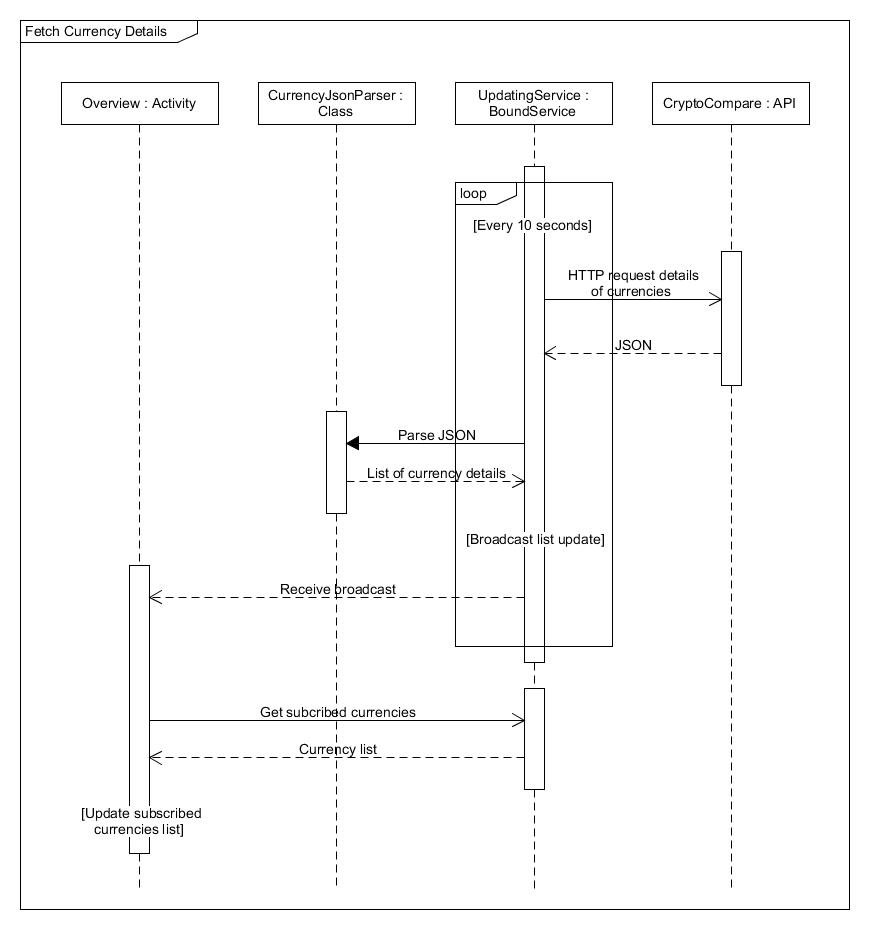


*Figure 3. Fetch currency list sequence diagram.*

## Fetch details

Fetch details is responsible for getting the latest price for all subscribed currencies. We fetch details every 10 seconds or when the user forces an update. 10 seconds is chosen because the CryptoCompare API caches data every 10 seconds, making updating more often than this of no use.

In the start the service made an API call for each subscribed currency. This was later optimized to only one call for all subscribed currencies. This was found to be an important optimization, as the service fetches details quite often. The fetch details sequence is described in Figure 3.



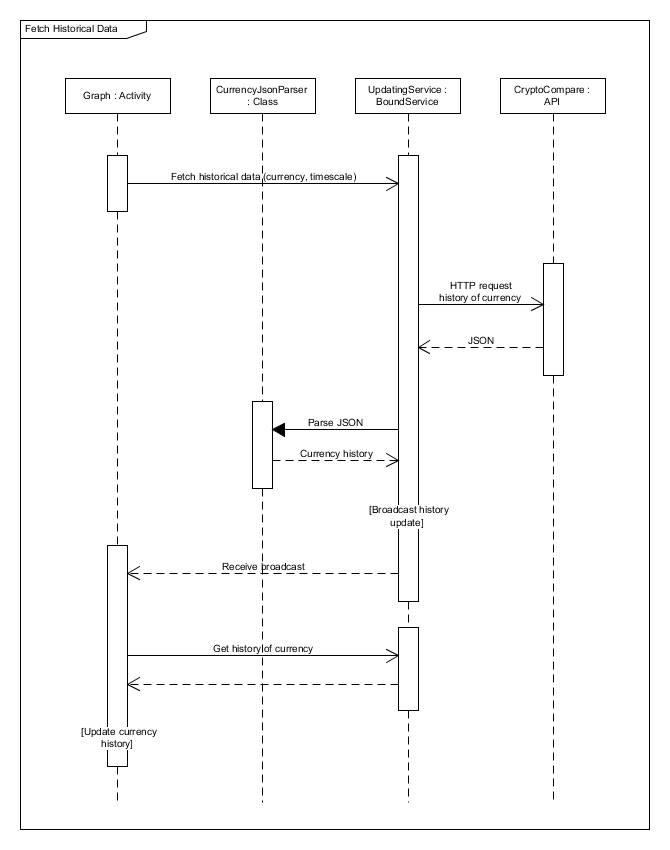
*Figure 4. Fetch details sequence diagram.*

## Fetch historical data

Fetch historical data is responsible for retrieving historical data for a currency. This data is used by the GraphActivity showing graphs for a currency. Most of the data is for populating data in the different graphs.

This API call caused some issues because the JSON response doesn’t contain information about which coin the data is for. Using the standard Volley library there was no information about which request the response was for. This issue was solved by creating a custom Request called HistoricalDataRequest by extending the request class. The HistoricalDataRequest saves the HTTP request URL and appends it to the JSON response. Using our custom JSON parser we parse the request URL as a currency using a regex expression.

The fetch historical data sequence is described in Figure 5.



*Figure 5. Fetch historical data sequence diagram.*

# Conclusion

The app Cryptonat0r has achieved to make investments easier for everyone. The app makes it possible to subscribe to any currency and follow its price, market cap and other useful details in real time. The Trending Currencies Activity is a proof of concept but will provide the user an overview of the current trending currencies, to optimize the user’s profits.

# List of known bugs, problems or future development ideas

If you make a price watch for a currency it will only trigger when the currency is within 2 USD of the price you set. This was done because you a currency will rarely hit the exact price you set. But the algorithm deciding whether the price watch should be triggered or not could be made better.

The app could show notifications when a new trending currency has been added.

# Work plan

During the development of CryptoNat0r the different responsibilities were primarily divided by the different activities, see Table 1. While one worked on getting the API calls right, another worked a lot on the JSON parser. From the beginning it was prioritized getting our data structures and the data that was put in them right, because it was essential for showing anything in the app. This prioritization turned out to work great.

|  |  |
| --- | --- |
| Task | Responsible |
| OverviewActivity (currency overview) | Shared |
| GraphActivity (detailed currency view) | Christian |
| TrendingActivity (trending currencies view) | Niklas |
| JsonParser (parsing API responses to objects) | Christian |
| App layouts | Shared |
| UpdatingService (background service making API requests) | Niklas |

*Table 1. Table showing who's responsible for which tasks.*

1. https://www.cryptocompare.com/api/# [↑](#footnote-ref-1)