CryptoNat0r

201509718 Christian Luke Pedersen, E

201509378 Niklas Meyer Møller Sørensen, IKT

13. maj 2018

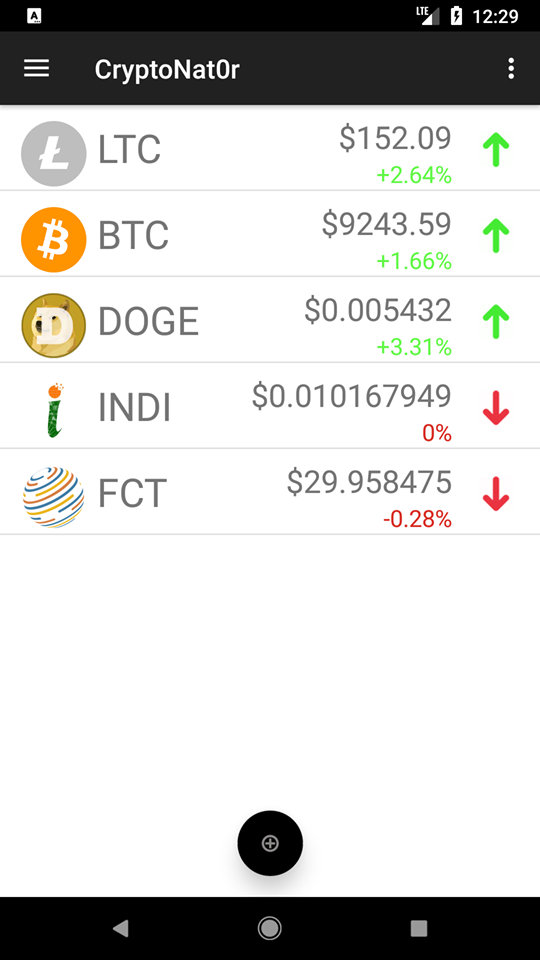


Table of Contents

[App vision 3](#_Toc513135678)

[Personal Vision 3](#_Toc513135679)

[Context 4](#_Toc513135680)

[Requirements specification 5](#_Toc513135681)

[Mandatory 5](#_Toc513135682)

[Optional 6](#_Toc513135683)

[UpdatingService overview 6](#_Toc513135684)

[Fetch coin list 7](#_Toc513135685)

[Fetch details 7](#_Toc513135686)

[Fetch historical data 9](#_Toc513135687)

[Conclusion 11](#_Toc513135688)

[List of known bugs and problems 11](#_Toc513135689)

[Work plan 11](#_Toc513135690)

# 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.

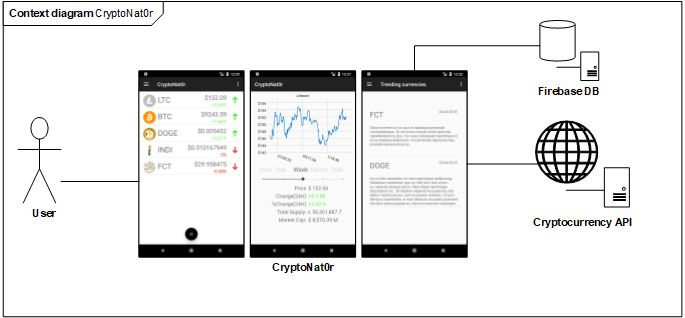


Figure 1. Context diagram for CryptoNat0r.

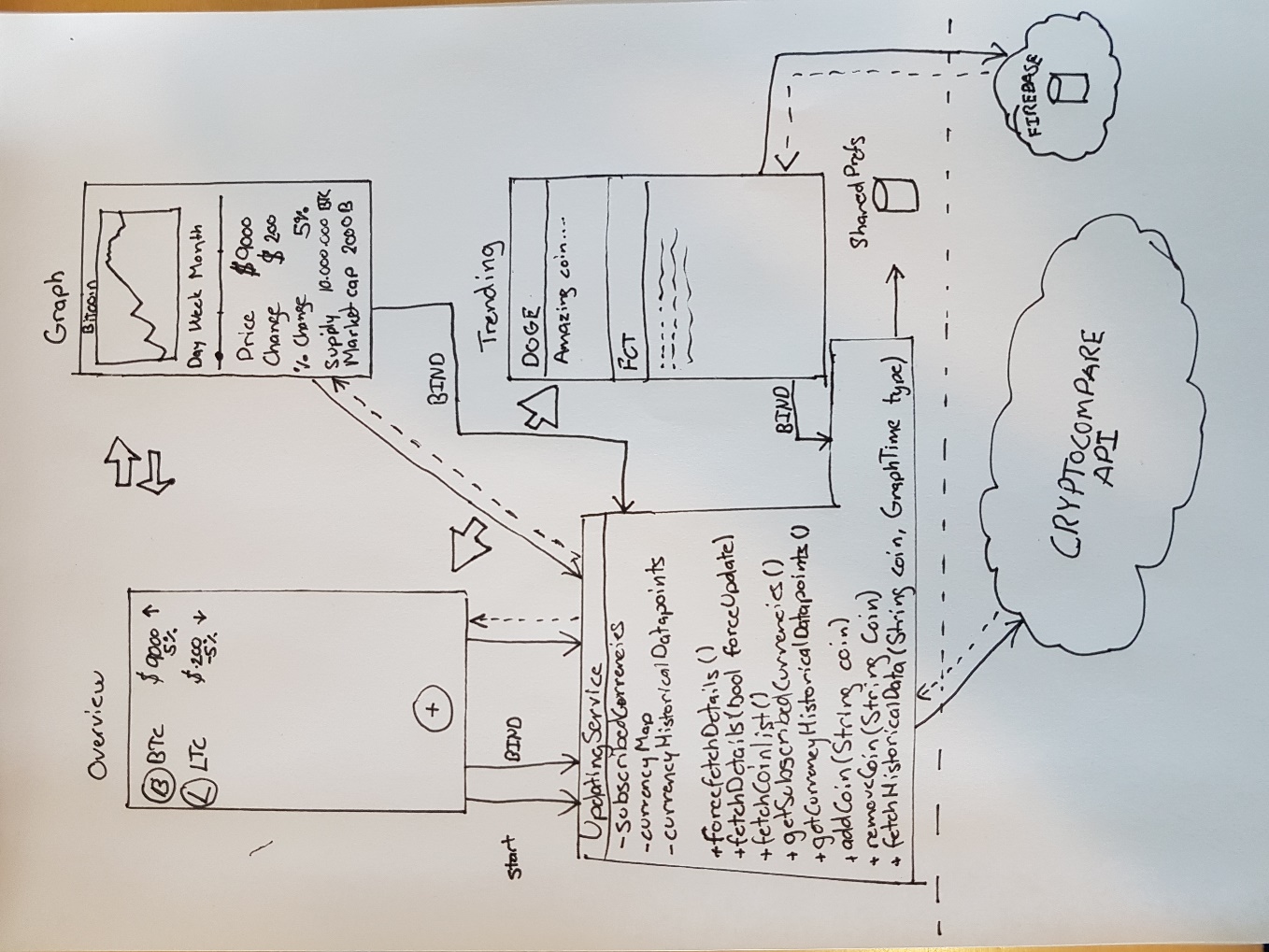


Figure 2. Detailed context diagram for the curious reader.

# Requirements specification

Below is a quick overview on how the required specifications for the app project was achieved in CryptoNat0r.

## Mandatory

* At least two activities
  + Achieved with MainActivity, GraphActivity and TrendingActivity
* Use of Intents to send data between components
  + MainActivity sends CurrencyData to GraphActivity when starting the activity.
* Persisting data through SharedPreferences and/or an SQLite database. Using Firebase is also permitted.
  + Subscribed currencies are saved in SharedPreferences. TrendingActivity retrieves trending currencies and information about these from a Firebase database.
* Some element of communication using internet, Bluetooth and/or WIFI
  + Volley is used for HTTP requests to all API calls.
* Use at least one Service
  + The bound service, UpdatingService, is being used throughout the app to retrieve data from our API.
* Use Asynchronous processing
  + Asynchronous processing is used whenever an API call is made.
* Use proper resource externalization
  + Resource externalization is used for all text strings being used by the layout. This is how the app is also available in Danish.
* Support at least two languages including English
  + The app is available in Danish and English.
* Use Layouts that adapt to at least two different screen sizes (phones and tablets)
  + Layouts has been made for both phones and tables.
* Have a custom app icon
  + The app has a custom app icon.
* Have app specific color scheme
  + The app has been made in a custom dark theme using guidelines from material.io.

## Optional

* Use AppBar and/or MenuDrawer
  + Both is used in the app. The AppBar is being used for manually refreshing subscribed currencies and the MenuDrawer is being used for navigating between Overview and Trending activity.
* Create custom views / UI elements
  + A graph library is being used to help create the graphs in GraphActivity. The ListViews in OverviewActivity and GraphActivity uses custom list items.
* Use BroadcastReceivers either internal/local or external
  + A BroadcastReceiver is used in the OverviewActivity so it knows when the UpdatingService has new information.

# 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 responsible for retrieving the latest rates and the historical data shown on the graphs in the GraphActivity and storing this information.

## 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. A sequence diagram describing fetching the coin list is shown in Figure 2.

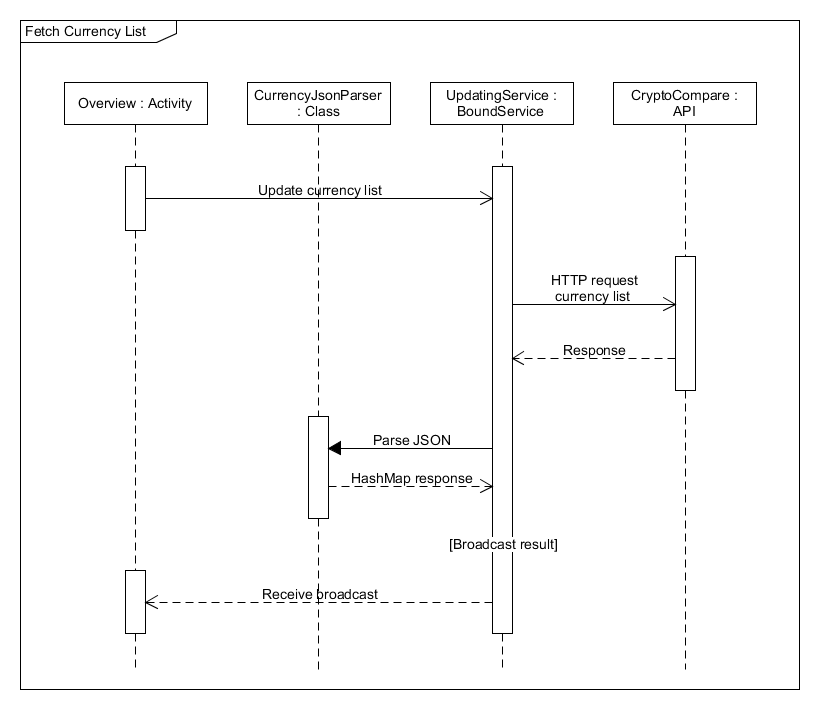


Figure 3. Fetch currency list sequence diagram.

## 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.

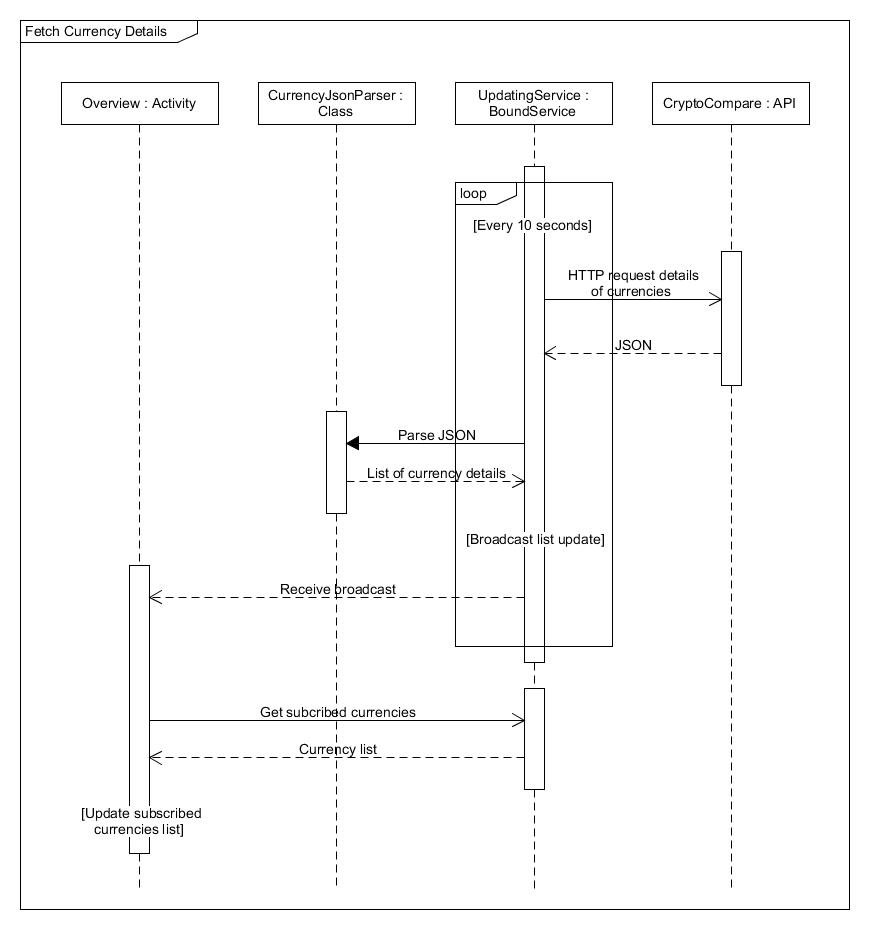


Figure 4. Fetch details sequence diagram.

## 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 4.

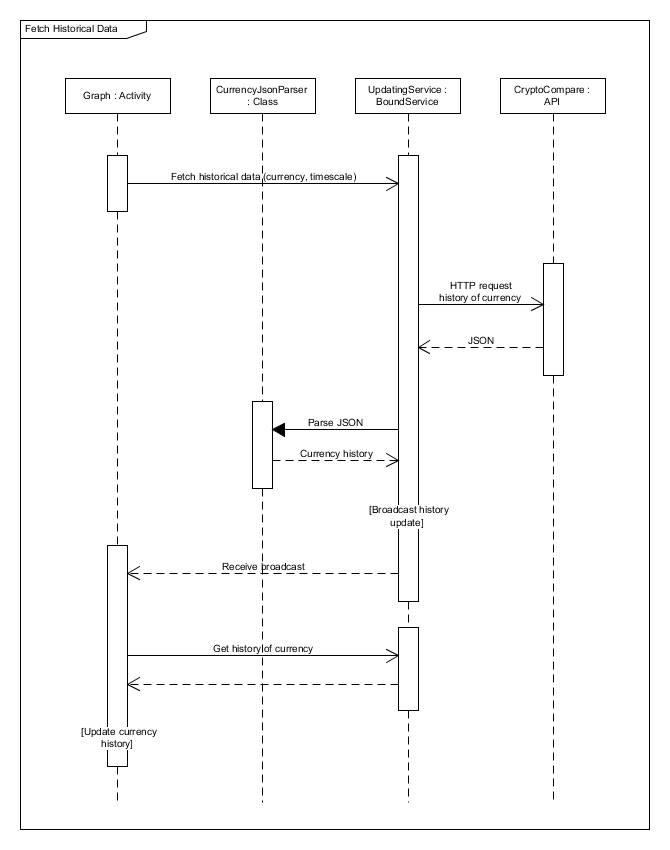


Figure 5. Fetch historical data sequence diagram.

# 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 and problems

None.

# Work plan

During the development of CryptoNat0r the different responsibilities were primarily divided by the different activities. 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 |
| Overview Activity | Shared |
| GraphActivity | Christian |
| TrendingActivity | Niklas |
| JsonParser | Christian |
| App layouts | Shared |
| UpdatingService | Niklas |

Table 1. Table showing who's responbility for which tasks.