Architecture Design Document

Special Notes:

- The app fetches the supported forex exchanges. Based on these exchanges the user can pick for which one the app should load symbols.
- The price variation is computed based on the latest readed value
- I noticed too late during the development that a search functionality would be appreciated, instead of this search feature, I've added a "watch/unwatch" feature that allows the user to get/ignore symbols data.

Architecture: MVVM
State Management: Riverpod
Navigation: go_router

I imagined this project split in multiple layers, that follow the MVVM architecture. The data source layer, which is taking care of where to fetch the data, the repo layer that takes care of the business logic, the view model layer which decides what to displays to the user and accepts feedback from the user, and the view layer where all the UI is created.

For this project, the data source layer was more special, since we have to take care of the WebSocket implementation too, which requires handling for connect/disconnect and provides data via a stream not a request/response format. All this handling is done in this layer.

After the data source layer is done, repo layer can take the data and manipulate it and deliver it to the VM.

After the VM gets the data, it can translate it into "ui" data that will be displayed on screen.

Future Improvements

- Handle the errors on the UI side, all the errors are forwarded to the UI, but no widget is displayed.
- Make the handling of loading/data/error abstract and reusable
- Test on web.
- Add more test coverage, I had some difficulties mocking the WebSocket lib classes.