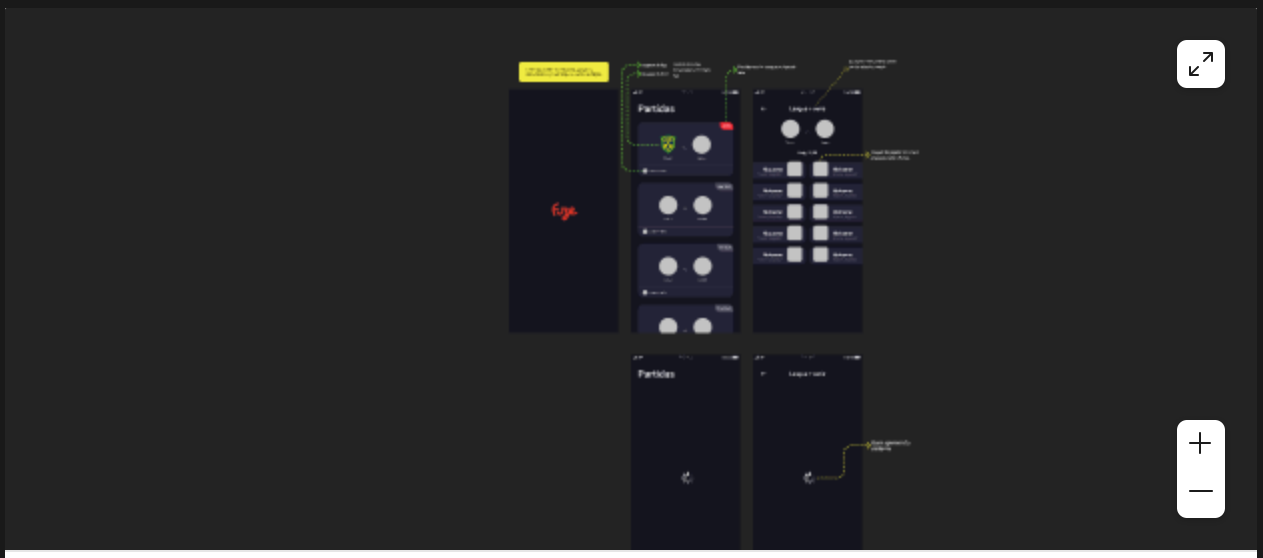
Android Dev (EN)

The candidate must build an app that will display a list of CS: GO matches in a given period of time. For that, the PandaScore API will be used.

The design spec is here: https://www.figma.com/file/OeNVxV2YkHXMgzky8YNQQO/Desafio-CSTV?node-id=0%3A1



Desafio CSTV · App Edited 2 years ago

What will be evaluated, in order of importance:

- Overall app performance and usability
- Adherence to the spec
- Code quality
- Project and code organization
- Git usage
- Optionals (**listed below**) will count positively to the final evaluation

App Spec:

CSTV is an app that displays CS: GO matches happening across several worldwide tournaments

- The app will display matches beginning from the current date. It will also include a “match detail” screen that will show details like the team names, roster, and match time
- There are 3 screens: splash screen, main screen (matches list), as well as the match details screen
- UI must follow the Figma spec above. Attention to detail is a must.
- The PandaScore API will be used for listing the matches as well as fetching match details
- Dates shown should follow the device local time

Splash Screen

This screen will show the app logo, and should be displayed every time the app is launched for a given amount of time before the main screen is shown.

Main Screen (Matches List)

This screen will display a list of matches, beginning from the current date. The list should be sorted so that currently playing matches will appear on top. Each match will show basic data like the League, Teams, date and time, as well as its status. The status can be: Ended, In Progress, and Scheduled.

Match Detail Screen

This screen will appear when the user selects a match from the main screen. Aside from the data shown in the matches list, this will also display the players from each team, with their respective names, nicknames, and pictures.

Requirements:

- The candidate must provide us with a link to a repository that includes:
  - A working project
  - A README file describing briefly what was implemented (libraries used, any architectural decisions the developer would like to explain), as well as instructions to run the project
- The app must have pull-to-refresh in the main screen to reload the matches list

- Accepted libraries:
  - Retrofit
  - kotlinx.serialization
  - gson
  - Coil
  - Glide
  - Android Jetpack Libraries

Optionals:

- Unit tests
- MVVM architecture
- Pagination support
- Reactive programming
- Responsiveness