



FMT Reversi Android

by
[Francesco Benincasa](#)



Roadmap

- ▶ Intro
- ▶ The project
- ▶ Demo
- ▶ Let's dive into the code
- ▶ Conclusions



Introduction

- The aim of the project is to realize a Reversi implementation on Android platform:
 - Match on the same device
 - Match between two devices (each player on his own device)
- The game rules are based on [Reversi on Wikipedia](#) and [Federazione Nazionale Gioco Othello](#)



The project - what did we use?



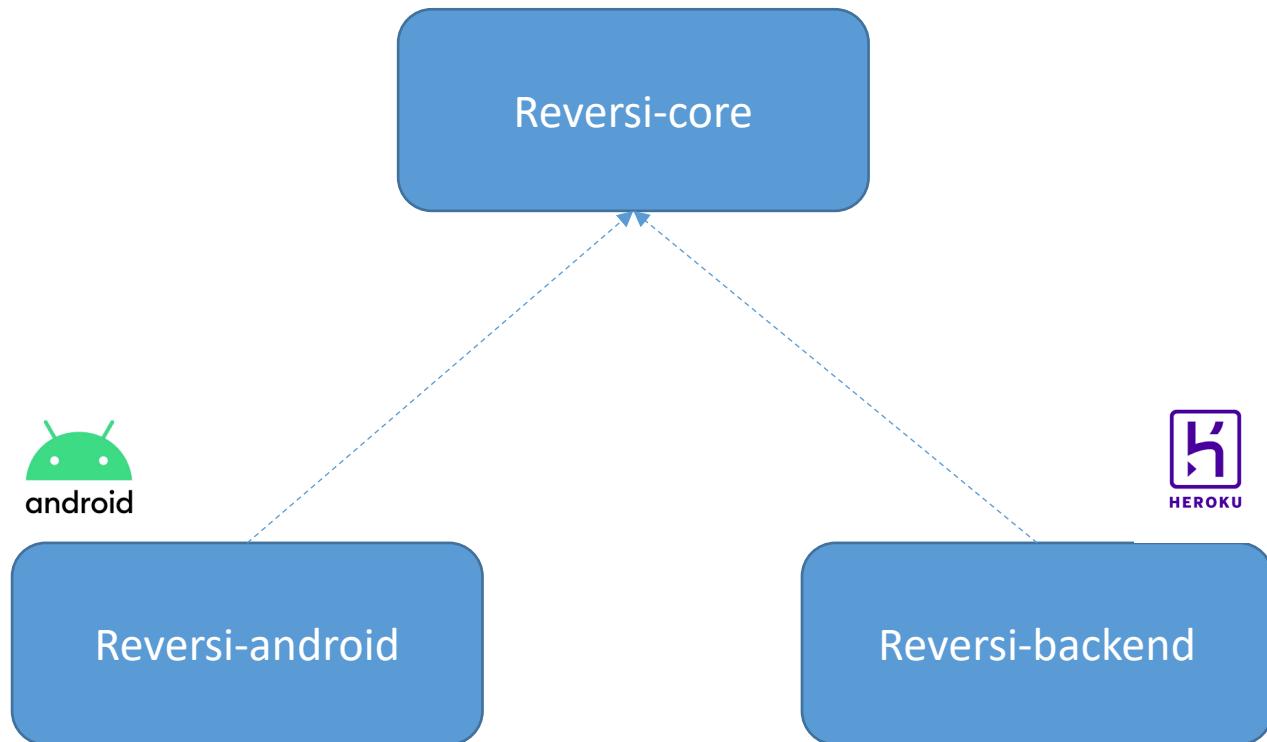
Maven[™]



ngrok



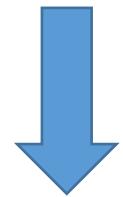
The project - modules





Build process

Android development



Backend development

A yellow downward-pointing arrow indicating the flow from Backend development to the final output.

ngrok

<https://836c360b507d.ngrok.io>



<https://fmt-reversi.herokuapp.com/>

Backend development

NGROK was used to expose local server on the web during development

The image shows a developer's environment with two main windows.

Java IDE (Left): A screenshot of an IDE (IntelliJ IDEA) showing the code for `WebPathConstants.java`. The code defines several static final strings for URLs related to user moves, ready status, and topic matching.

```
public static final String WS_USER_MOVES_URL_SEGMENT = "/users/{uuid}/moves";
public static final String WS_USER_READY_URL_SEGMENT = "/users/{uuid}/ready";
public static final String WS_USER_NOT_READY_URL_SEGMENT = "/users/{uuid}/not-ready";
public static final String USER_READY_URL_SEGMENT = "{uuid}/ready";
public static final String USER_NOT_READY_URL_SEGMENT = "{uuid}/not-ready";
public static final String USER_MATCH_URL_SEGMENT = "{uuid}/match";
public static final String WS_TOPIC_USER_MATCH_DESTINATION = TOPIC_PREFIX + "/user/{id}/match";
```

Browser (Right): A screenshot of a web browser displaying a page from `https://katesapp.ngrok.io`. The page says "Welcome to Kate's Site! It's currently under development...". Below this, it shows the command used to start the tunnel: `./ngrok http 3000` and `ngrok by @inconsreveable`.

Session Status:

- Account: Kate Libby (Plan: Pro)
- Web Interface: <http://127.0.0.1:4040>
- Forwarding: <http://katesapp.ngrok.io> -> localhost:3000
- Forwarding: <https://katesapp.ngrok.io> -> localhost:3000

Backend development

To deploy backend server, github, travis and heroku were used

The image displays three screenshots illustrating the deployment process for a backend application:

- Github Repository (xcesco/reversi-backend):** Shows the repository structure with files like .mvn/wrapper, docs, src, .gitignore, .travis.yml, HELP.md, LICENSE, README.md, mvnw, mvnw.cmd, pom.xml, reversi-backend.iml, and system.properties. A recent commit by xcesco is shown.
- Heroku Application (fmt-reversi):** Shows the Heroku dashboard for the app, including sections for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. It indicates no add-ons are installed and shows deployment history.
- Travis CI Build Log (xcesco/reversi-backend):** Shows the build status as "passing". The log details a successful build (#50) for the master branch, comparing commits 9897118 and 3c462ee5, running on an AMD64 machine.



Android development

- ▶ View model & live data
- ▶ Shared preferences
- ▶ Navigation
- ▶ Intent
- ▶ Retrofit
- ▶ Websocket & STOMP
- ▶ Recycler View
- ▶ Jackson
- ▶ RX java
- ▶ Timber
- ▶ Firebase
- ▶ Dagger2



websocket navigation timber model_view rx java live_data firebase jackson intent dagger2 view retrofit recycler-view room view_binding



About Android development

- ▶ Retrofit: <https://square.github.io/retrofit/>
- ▶ Okhttp: <https://square.github.io/okhttp/>
- ▶ Websocket & STOMP protocol: <https://github.com/NaikSoftware/StompProtocolAndroid>
- ▶ Jackson: <https://github.com/FasterXML/jackson>
- ▶ ViewBinding: <https://developer.android.com/topic/libraries/view-binding#java>
- ▶ Dagger: <https://github.com/google/dagger>
- ▶ Navigation component: <https://developer.android.com/guide/navigation/navigation-pass-data>
- ▶ ModelView & Live data components:
<https://developer.android.com/topic/libraries/architecture/livedata>
- ▶ Timber: <https://github.com/JakeWharton/timber>
- ▶ Firebase
<https://firebase.google.com/docs/crashlytics/get-started?authuser=0&platform=android>

Why WebSocket?

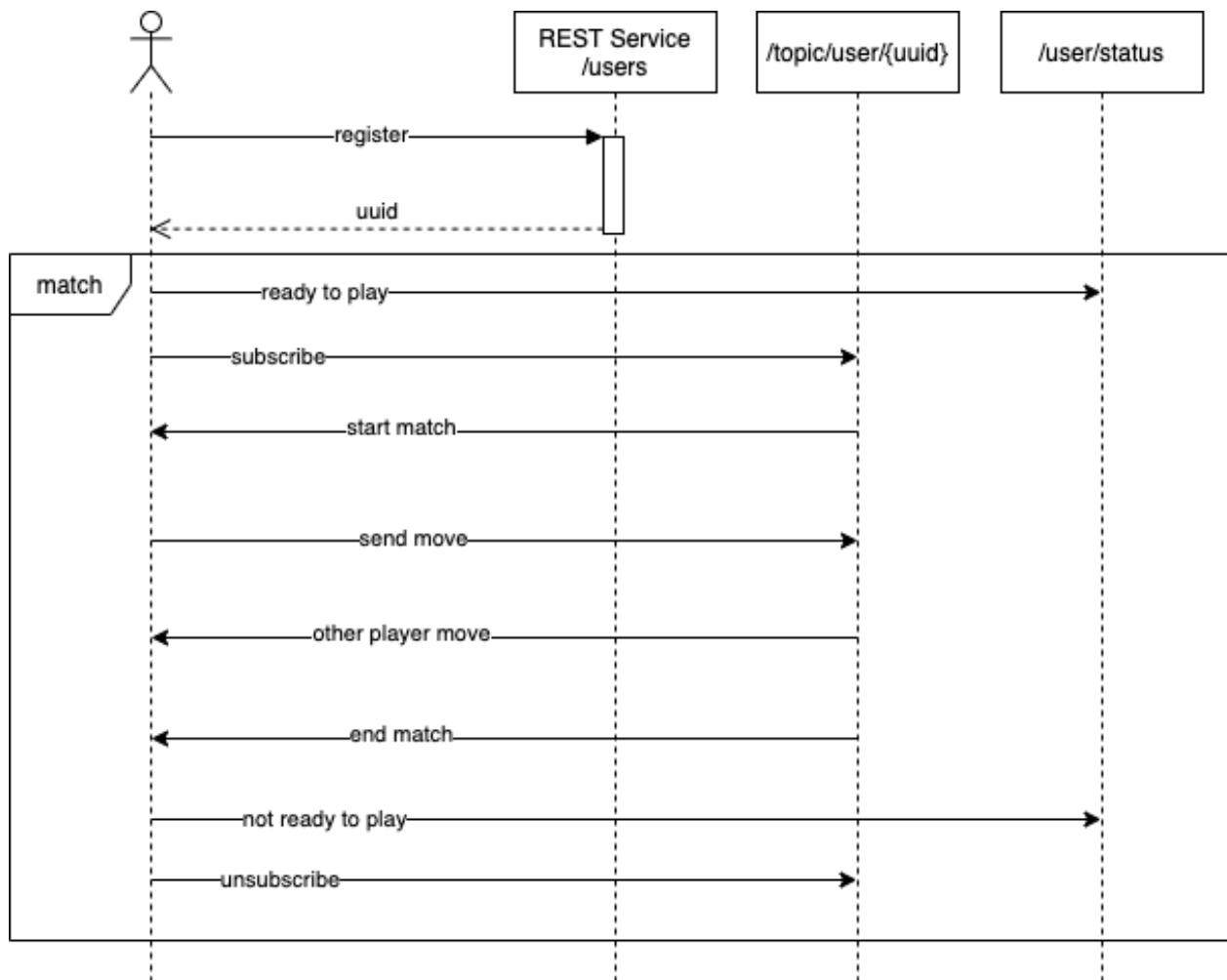
- ▶ Game server contains match status
- ▶ Network matches require in some case server starts communication with clients.
- ▶ The choice was Simple (or Streaming) Text Oriented Message Protocol (STOMP) on WebSocket

```
D/AbstractConnectionProvider: Receive STOMP message: MESSAGE
    type:MATCH_START
    destination:/topic/user/c0056fbe-9cbc-44f7-9c82-44930b477f52
    content-type:application/json
    subscription:066fc93-cfa5-49c8-b2f9-690a8b6c4b1c
    message-id:246aa132-e4b5-d2fb-b1fc-56a7111962a9-1
    content-length:257

    {"player1Type":"NETWORK_PLAYER","player1Name":"BOT","player2Type":"HUMAN_P
```

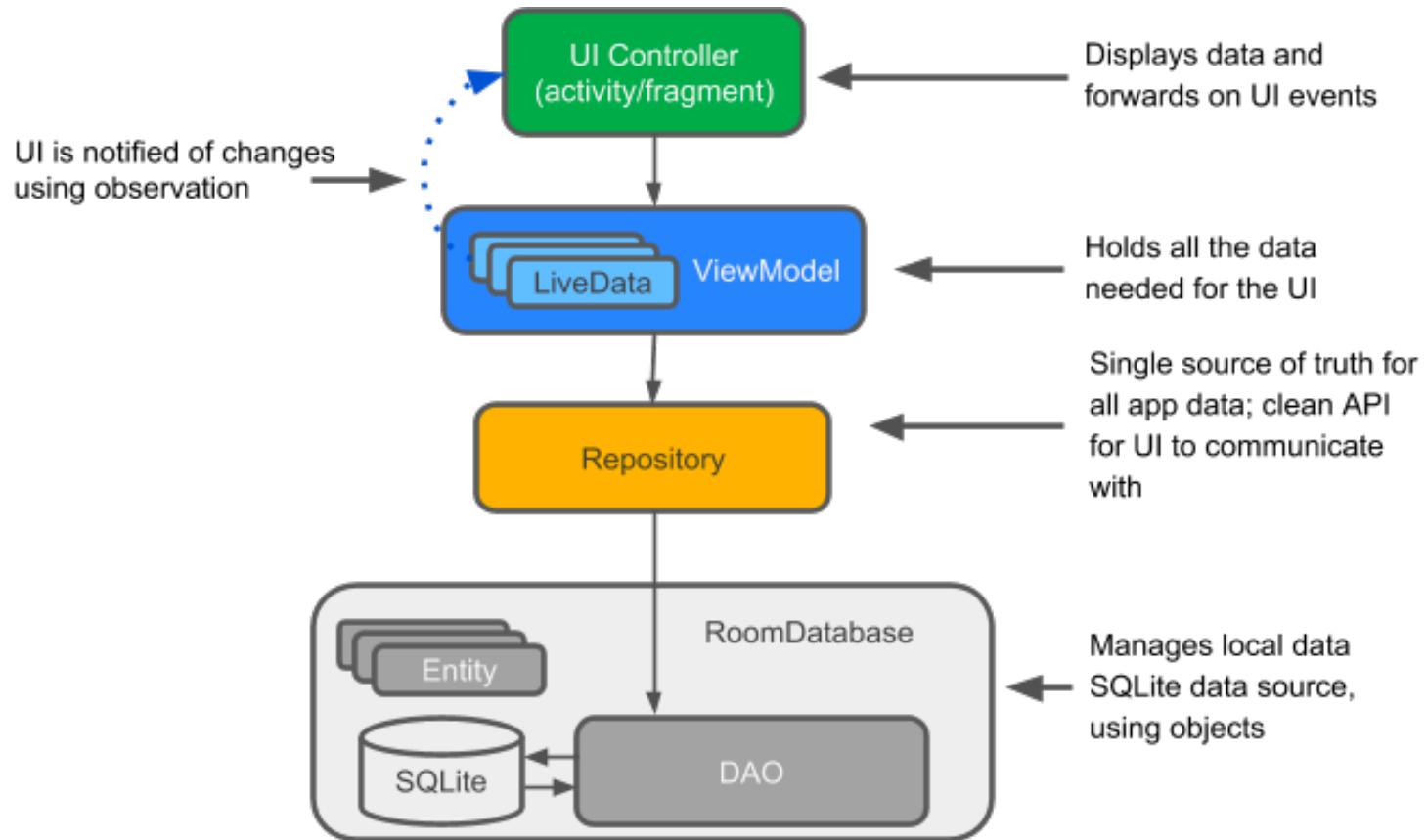
HTTP	WebSocket
Duplex	
Half	Full
Messaging Pattern	
Request-response	Bi-directional
Service Push	
Not natively supported. Client polling or streaming download techniques used.	Core feature
Overhead	
Moderate overhead per request/connection.	Moderate overhead to establish & maintain the connection, then minimal overhead per message.
Intermediary/Edge Caching	
Core feature	Not possible
Supported Clients	
Broad support	Modern languages & clients

Why WebSocket?



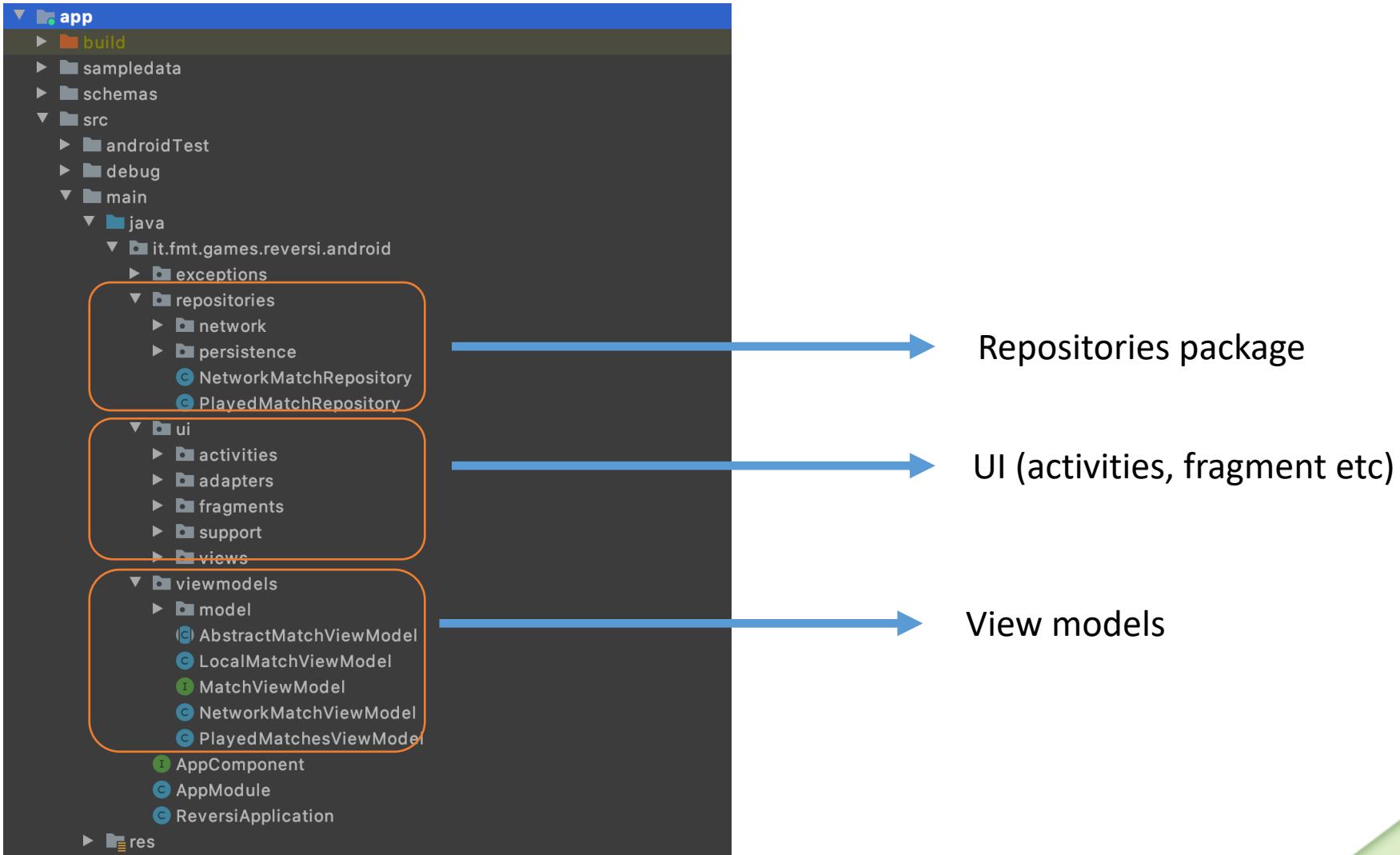
- ▶ Client connects to server with a web service (synchronized request-response)
- ▶ Client receives an UUID will be used from server to identificate it.
- ▶ Client subscribes to match topic to receive server communication.
- ▶ JSON message, based on STOMP over WebSocket.
- ▶ Clients send data to server and receive async message from server, until match finish.

Why Live Data and View Model?

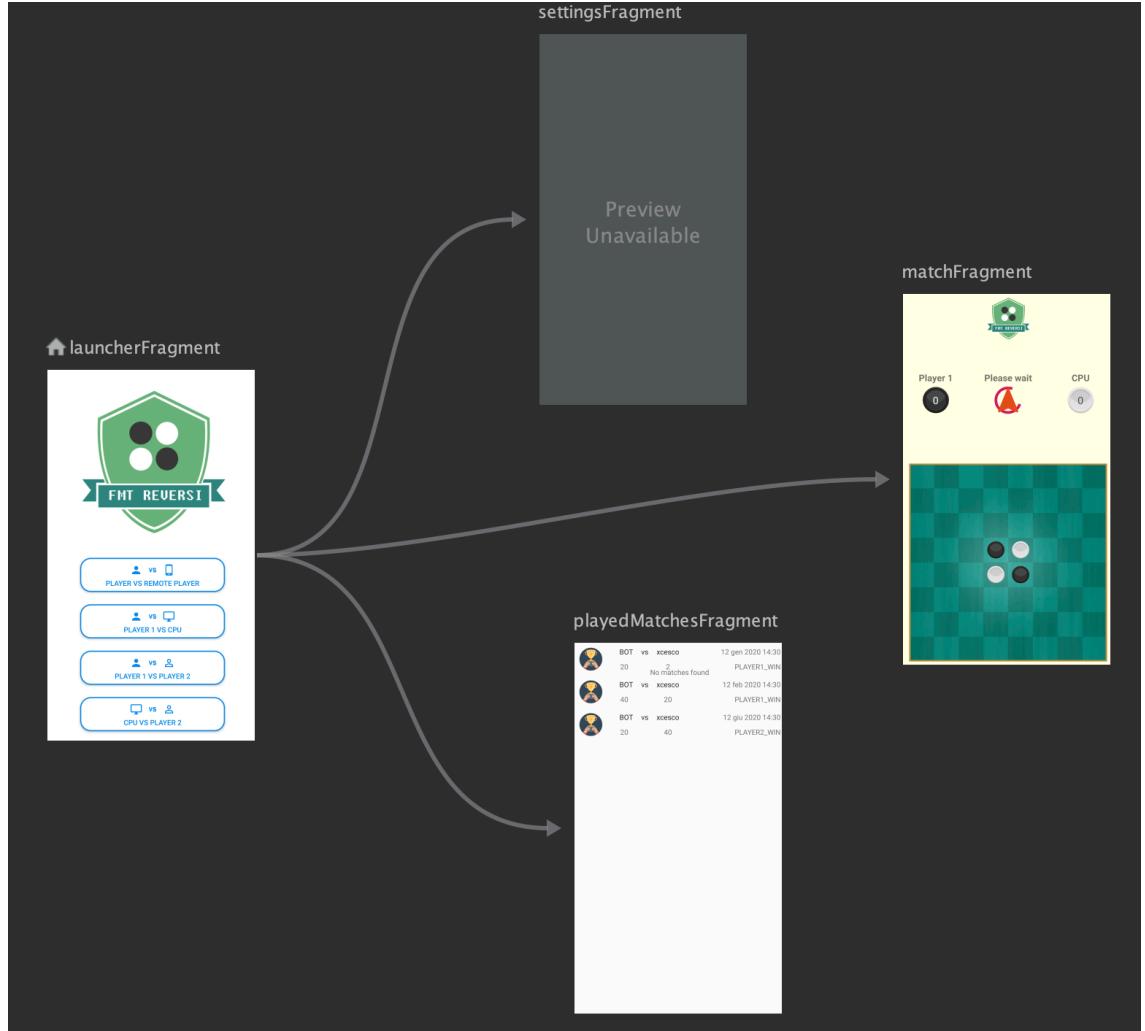


- ▶ Live data and view models allow to manage model changes on UI respecting UI components lifecycle

Source code organization

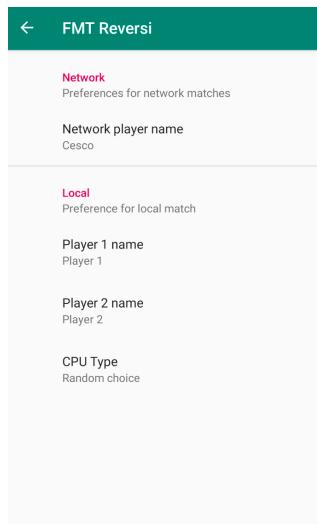


Navigation between fragments

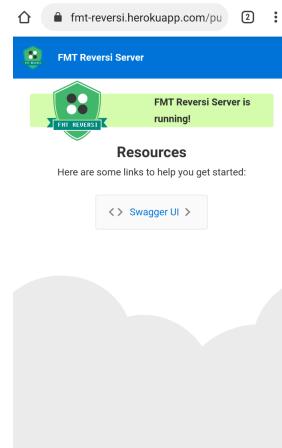
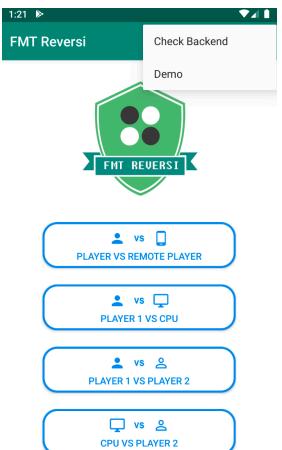
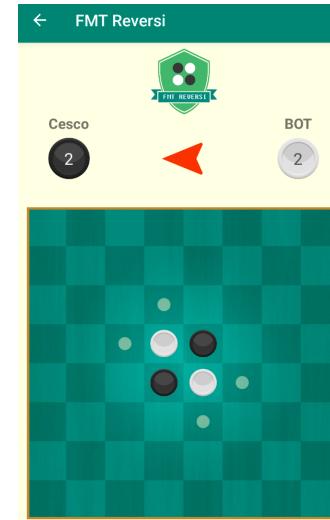


1 activity, 4 fragments, 1 navigation graph

Navigation between fragments



FMT Reversi		
Network	Cesco	vs BOT
	39	25
	PLAYER1 WIN	
Network player name		
Local	Cesco	vs BOT
	29	35
	PLAYER2 WIN	
Preference for local match		
Player 1 name	Player 1	
Player 2 name	Player 2	
CPU Type		
	Random choice	



The screenshot shows the API documentation for the Reversi FMT API. It includes sections for Match Controller and User Controller, with various HTTP methods listed for each endpoint.

match-controller
GET /api/v1/public/matches

users-controller
DELETE /api/v1/public/users
GET /api/v1/public/users
GET /api/v1/public/users/{uuid}/match
PATCH /api/v1/public/users/{uuid}/ready
PATCH /api/v1/public/users/{uuid}/not-ready
POST /api/v1/public/users

Demo



It's time to play!

Android version available on [Google Play Store](#)

<https://play.google.com/store/apps/details?id=it.fmt.games.reversi.android>



Network match demo on <https://youtu.be/RUfBwd1IXWg>





Source code on GitHub

- ▶ [FMT Reversi Android source code on Github](#)
- ▶ [FMT Reversi Backend source code on Github](#)
- ▶ [FMT Reversi source code on Github](#)



Conclusions

- ▶ FMT Reversi can be improved:
 - ▶ Support for other platform (web)
 - ▶ Improved IA for CPU players
 - ▶ Google play game service
 - ▶ PS4 version cooming soon!
- ▶ Any question?

Thanks!

