Mobile Application Design 2017 project

µCoaching

team: Filip Gulán, xgulan00
Jan Herec, xherec00
Marek Marušic, xmarus05

Project Goal

- Do you have some goals or dreams for example learn new language or progress with your diploma thesis? Are those dreams pushed out of your focus by some other "important" stuff like procrastination or you just forget about them for a long time?
- This app and your friends can help you to focus more and to get more motivation for accomplishment of your goals by micro - coaching.
- You can create new goal and ask your friend to micro-coach you.
- App has two modes: performer and coach. Performer sets goal and coach helps or motivates him to achieve this goal.
- Performer can set milestones for goal, and then after they are done he can check them.
- Your goals are shown in the performer screen where you can see your todos, graph with your progression and messages from your coach.
- Performer is pushed towards by coach (mostly his friend) to achieve his goal
- Performer rates his progress by emoticons within conversation with coach and this progress is visualized in graph.
- Our app provide bidirectional conversation respective brief exchange between performer and the coach.
- User of the app can of course coach and perform more goals.

Used Technologies

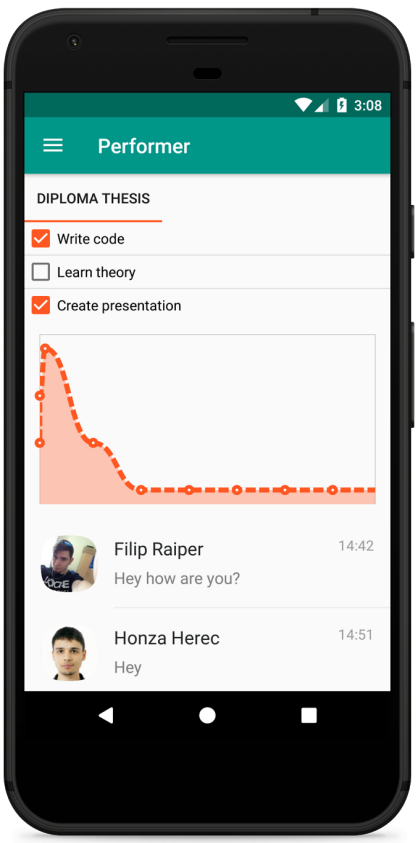
- Platform: Android (used in the app)
- IDE: Android studio
- Programming language - Kotlin
- Version control: Gitlab
- DB: Firebase realtime database (used in the app)
- Other libraries
    - MPAndroidChart: library to draw various advanced charts
    - ChatKit: library to make awesome chat user interface

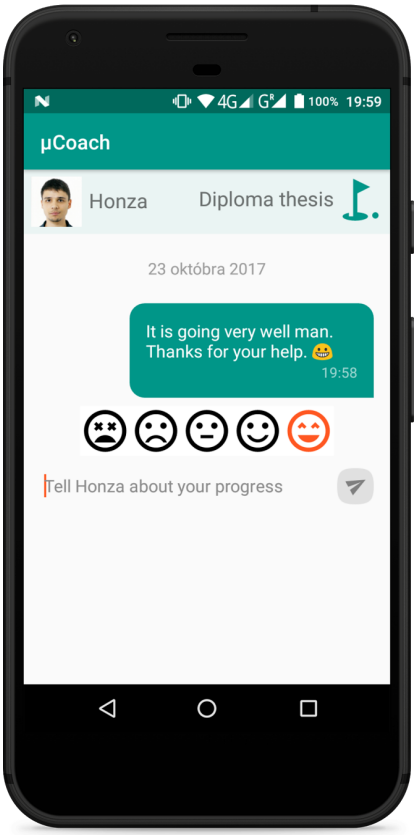
Used Resources

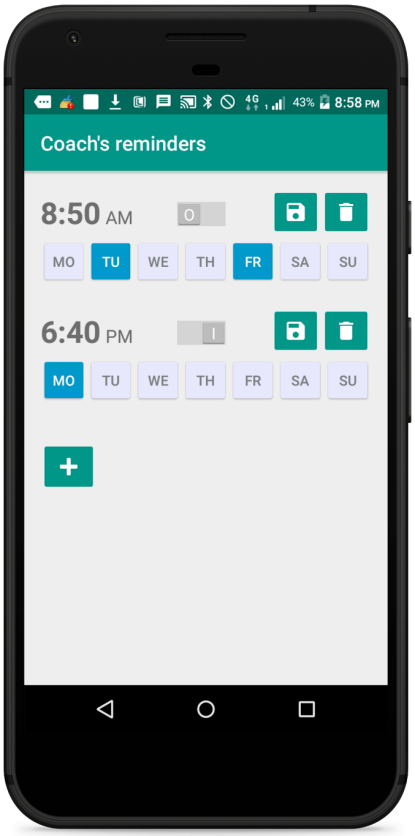
- Study material and existing code: mainly Android developers page and stackoverflow hints to solve particular problems
- Images: graphic form standart android library

Most Important Achieved Results

- All important about performed goal clearly showed in one screen - goal name, milestones, progression graph, coaches (names and photos), and last message from conversation within coaches. Performer can check milestones, move in progression graph, switch between his goals, start a conversation with coach.
- Realtime conversation between performer and coach. Last message is displayed with possibility to go throught the message history exploring. Performer rates his progress by emoticons within conversation. Everything important clearly showed - goal name, coach name (and their photos), last message, and field to write message.
- Coach´s reminders - for selected goal coach can set reminders which reminds him to coach this goal. Setting reminders is impressively easy, joyful and intuitive. Every remider and his setting is compactly showed in one screen.







Controlling of Created Program

Performer screen

- Can show navigation drawer with options to switch to coach mode or create/edit goal.
- Can move between tabs which represent performers goals.
- For selected goal can move in progression graph to past.
- For selected goal can check milestones.
- For selected goal can open conversations with goal coaches.

Coach screen

- Can show navigation drawer and there are options to switch to performer mode or create/edit reminders.
- Can move between tabs which represent coached goals.
- For selected goal can move in progression graph to past
- For selected goal can open conversations with performer.

Messaging screen

- Coach or performer can exchange messages or can show messages history.
- Performer can rate his progress with emoticons.

Create/edit goal screen

- Performer can set/edit/remove goal name, milestones.
- Performer can choose/remove coach for goal by typing his name. Coach must be registered user in application.

Create/edit reminders screen

- Coach can set multiple reminders for the goal he coaches. Reminders are implemented using android notification system. For particular reminder coach can set time and days in week when he wishes to be notified. Reminder can be active or inactive. Reminder is weekly by default.

Experience with the Selected Platform

- Android - app publishing process is complicated, restrictions due Android's obsession of saving battery are annoying, many OS variety and different restriction for each in compare with iOS

- Kotlin - has null safety, but there missing promises functionality
- Firebase - a lot different experience than the relational databases (json data modelling, asynchronicity - due missing promises in Kotlin we experienced real callback hell)
- Difficult testing on various devices.

Distribution of the Work in the Team

- **Filip Gulán - formal team leader**
  - graphs fragment
  - message chat partners list fragment
  - message activity and functionality
  - message push notifications
  - mockups design
  - database design
  - deployment of project
  - icon of app
  - Google play beta preview
  - documentation & presentations
- **Jan Herec**
  - user registering
  - creating and editing goal
  - reminders screens and functionality
  - mockups design
  - database design
  - documentation & presentations
- **Marek Marušic**
  - performer and coach activity
  - integrations of fragments into activities
  - mockup design
  - database design
  - documentation & presentations

Our team is rather peer-to-peer based, than master-slave. So we have rather formal leaders than true leaders..

What Was the Biggest Challenge

Before we started coding we were designing app and it took quite lot of time:

- On several common physical meetings we discussed app UI and possible technologies to use
- Then everyone created his own mockups of app based on previously discussed app design and we were finding the best consensus (using video/audio conference)
- Next everyone created his own UML DB draft based on consensus mockup and the JSON data model (due firebase) based on his own UML DB draft. Then we searched consensus in JSON data model (using video/audio conference)
- After that, when we have designed app from UI and data model perspective we divided work and finally started to code

From coding perspective we were working in parallel on different project parts with similar speed with no insuperable issues. So everyone faced different challenges and there is no consensus what was the biggest complication, or what took unreasonable time -- everyone see it differently. In any case, we did code reviews, and it took objectively some time -- to check colleague's code and test behaviour of application before approving and merging collegue's contribution to master branch.

Experience Gained From the Project

- Presentation of project before large audience is is good experience
  - We learned that we have to work on our presentation skills a lot
- Kotlin language - new modern official language from android platform

- Working with android fragments
- Learn data modelling for firebase NoSQL database, learn how to create a data model which scale, learn working with firebase in kotlin
- Search for consensus within team and doing compromis
- Firebase cloud messaging and push notifications and We know now, that is not best solution for user to user push notifications, because as We know it is not directly supported.

Autoevaluation

Technical Design: 90%

Before coding we analyzed problem, design UI using mockups and create DB model in UML and JSON. Selected tools and technologies was appropriately.

Programming: 70%

Code is structured in many class, often commented. Reusability is questionable, some parts of code (some modules/class) are, some aren't. We did code review and pull requests and merging.

Usability of the Created Solution: 70%

Application is practical usable even if there aren't all features we wanted. GUI is nice, but some things user says aren't very intuitive and need user experience improvements.

Use of Resources: 90%

Often used existing libraries, used existing code and ideas how solve problems.

Time Management: 70%

Project isn't in phase we exactly wanted (some functionality isn't implemented yet and some user feedback isn't incorporated), however app is functional. We started developing during summer holidays, it was good choice, because there was not much time during semester.

Team Cooperation: 90%

Teamwork really worked. Work was balanced, communication almost on daily basis. Can't imagine better team.

Chances of Publishing the App: 100%

App was published 23. 11. 2017 on Google play but only in beta for course presentation, so Google play presentation is not tailored and it is only temporal.

Overall Impression: 80%

Lot of work was done, lot of lines of source code was written, lot time was on project spended, lot of experiences gained. App is quite useful for microcoahing. Goal was almost achieved, but not all is done as we planned, not all user feedback is incorporated and we need improve GUI and UX.

Five Main Questions of TAM

What drives this sector of IT?

- VR
- Simple apps and games
- Wearables
- IoT.

What it will be like in five years?

Mobile will have better batteries, folding display, augmented reality used to provide information about surroundings, 3d screens.

What slows it down, speeds it up?

Who knows...

What ideas are dead (though they appeared great once)?

- Windows mobile platform is dead
- NFC payment are dead.

Where do new ideas come from?

Research teams - Google, IBM, Universities etc.

Recommendation for Assigning Future Projects

- Within project registration take in account that windows phone is dead a left registration to this platform as optional and provide enough android slots to register, because not everyone has mac computer to develop for iOS platform… However we are satisfied, because we were assigned to Android platform.
- UX lecture did not take place :-(
- Workshops are quite fine, students improve presentation skills before a large audience.

Recommendation for Future Students

- Quite important: If you are thinking about app database - firebase or traditional relation, there is no free lunch. Firebase as NoSQL is suited for particular app data type better than relational db, and vice versa. Every technology has advantages and disadvantages and you have to make sure, that for your problem it has significantly more advantages than disadvantages. For example if you use firebase, you don't have to maintain your own server. You don't have to worry about scaling because it scales very good if the right design is used. It is better for non-highly structured data than relational databases and it provides a real-time synchronization across connected clients, so for real-time application it is the best choice. But for a lot of highly structured data firebase can't offer you more advantages than sql queries. Last but not least you rely on technology, which is not in your hand. If it is shut down you can't do nothing.
- Quite important: In presentation use rather images than simple text (avoid slides with lots of text)
- Very important - select some interesting problem to solve and focus on few things deeply than create complex application - there is no time for that in semester
    - The simpler the better (Just remember that the simplest - fidget spinner in phone was one of the most successful projects and it was very simple one.)
- And most important: win in clickwar platform for which you have suited computer - if you haven't got a mac, then avoid iOS platform, and at all cost avoid windows mobile platform which is dead.

Miscellaneous