QUIZ APP

Programiranje za suvremene procesore
2018./19.

Mihaela Gamulin
Valentina Gavranić
Dajana Jerončić
Luka Valenta

Asistenti: Lovro Rožić i Luka Mikec

Kazalo

1. Općenito o aplikaciji

Aplikacija je zamišljena kao igra u kojoj je cilj prijeći cijeli kviz, odnosno prijeći sva tri levela iz svake od pet kategorija. Korisniku je omogućeno registrirati se te ulogirati na aplikaciju, što mu pruža mogućnost da bude ulogiran u isto vrijeme na više uređaja.

Neke od mogućnosti unutar aplikacije su provjera vlastitog napretka, dohvat liste globalno najboljih igrača te samo igranje kviza. Kod igranja kviza bira se kategorija te nakon toga kreće kviz u kojem su zadana pitanja iz odabrane kategorije za korisnikov trenutni level. Ukoliko su sva pitanja točna, level se ažurira, te kod ponovnog odabira te kategorije dohvaćaju se nova, teža pitanja.

Pitanja su zbog bržeg dohvata spremljena lokalno na mobitelu koristeći bazu ObjectBox, dok je baza s informacijama svih korisnika spremljena na backendu, odnosno Ruby on Rails aplikaciji koja se nalazi na heroku-u, zbog čega je potrebno prije pokretanja aplikacije posjetiti stranicu https://androidquiz.herokuapp.com/ kako bi se aktivirati dyno-i te omogućili api pozivi.

Aplikacija je testirana na sljedećim uređajima:

- Emulator: Nexus 5X API 26
- Mobitel: Huawei P9 Lite (Android 7.0, API 24)
- Tablet: Samsung Galaxy Tab 3 Lite (Android 4.4.4)

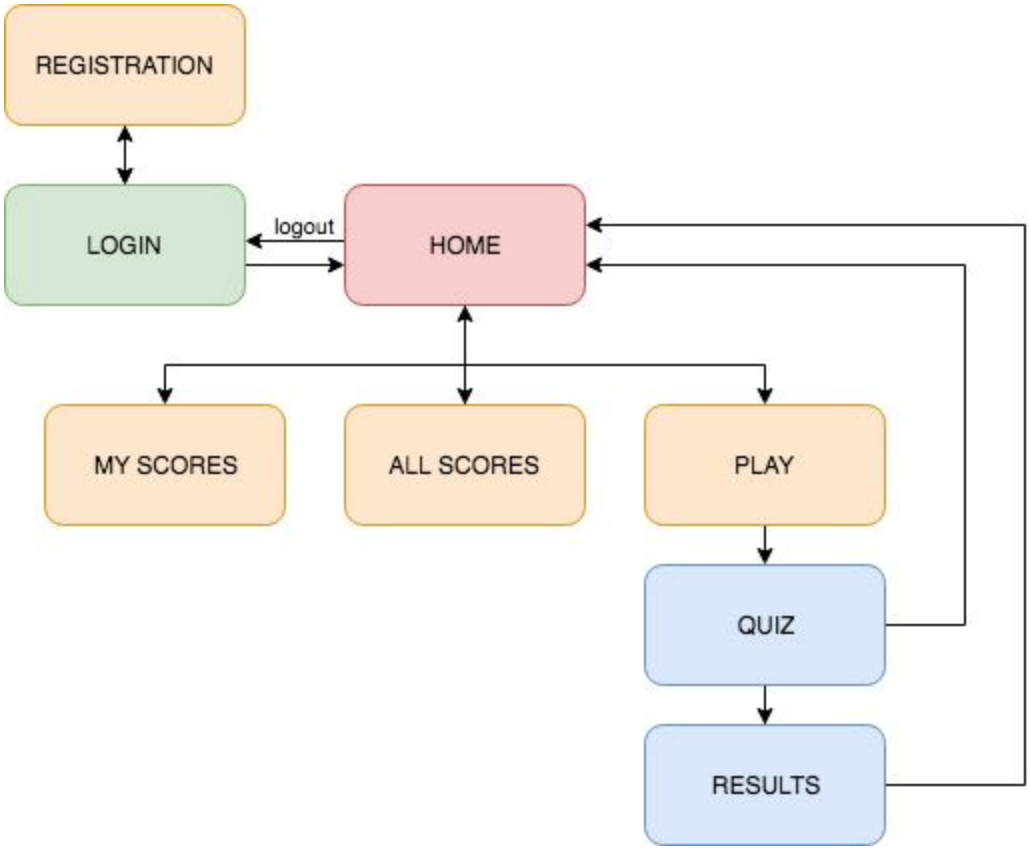
2. Tijek aplikacije

Osnovne aktivnosti aplikacije prikazane su na activity dijagramu (Slika 1). Nakon uspješne registracije u **RegisterActivity** šalje se mail u kojem je potrebno kliknuti na priloženi link kako bi se potvrdila registracija. Zatim je moguće ulogirati se pomoću **LoginActivity** u aplikaciju koristeći korisničko ime i password. Kod logina, u shared preferences se spremaju bitne informacije o korisniku kao što su token te trenutni leveli po kategorijama.

U glavnom izborniku Home activityja **MainActivity** moguće je vidjeti vlastiti progress - **MyScoresActivity**, dohvatiti rezultate najboljih igrača - **AllScoresActivity** ili igrati kviz.

Kod odabira kviza, bira se jedna od 5 mogućih kategorija u **CategoryActivity**, a zatim se dobije niz od 10 pitanja na koje je potrebno odgovoriti. Nakon što kviz završi dobije se potpuni izvještaj po pitanjima koje je točno/netočno u **GameFinishedActivity**.

Moguće se odlogirati iz aplikacije pri čemu se brišu podaci o korisniku iz shared preferences te se prikaže ponovno **LoginActivity**.

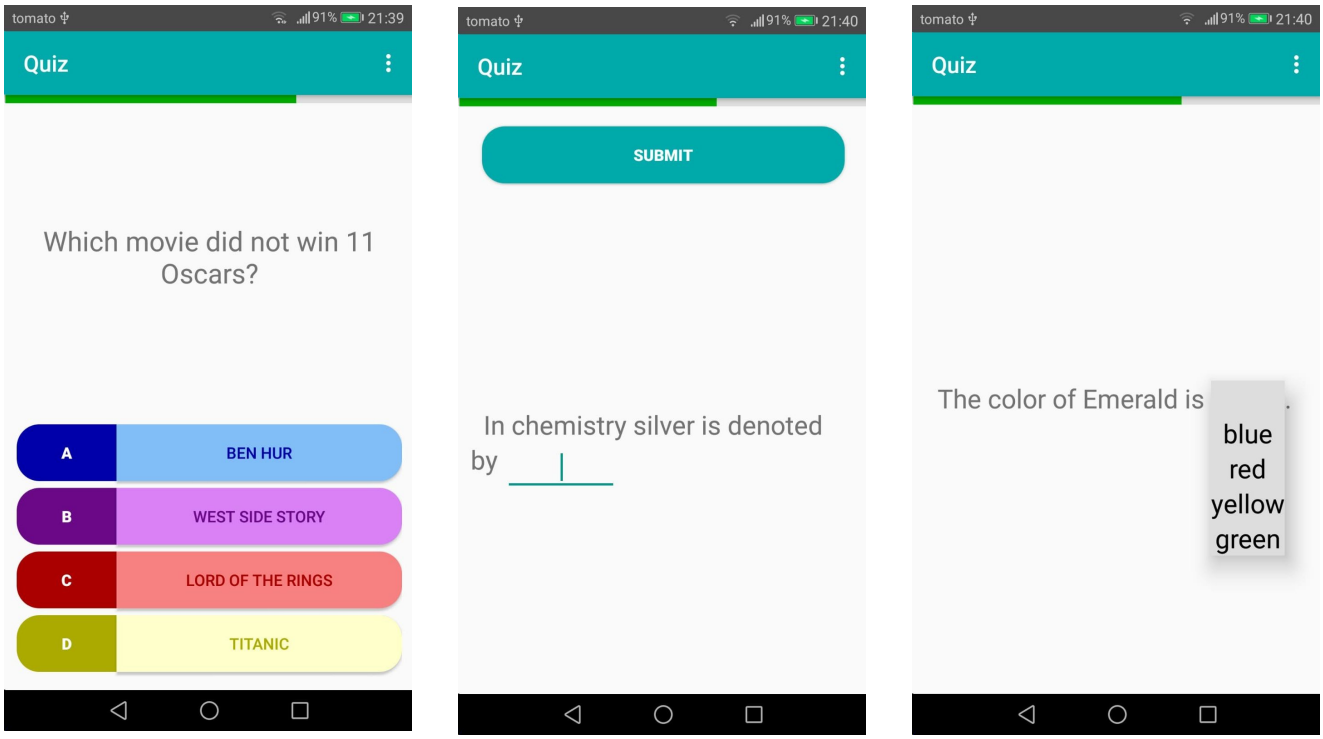


Slika 1. Dijagram prijelaza između activity-ja

3. Kviz

Kod svakog ulaska u kviz, dohvaćaja se 10 pitanja u ovisnosti o odabranoj kategoriji te trenutnom levelu korisnika za tu kategoriju. Svako pitanje može biti jednog od tri tipa: select, input, dropdown. Korisnik ima 15 sekundi za odgovoriti na pitanje, a ukoliko ne odgovori, odgovor se računa kao netočan. Na kraju kviza korisniku je prikazan cijeli izvještaj (Slika 2) po pitanjima gdje može vidjeti koje mu je pitanje točno, odnosno netočno. Izgled pitanja prikazan je na Slici 3.

Slika 2. Primjer izvještaja po pitanjima

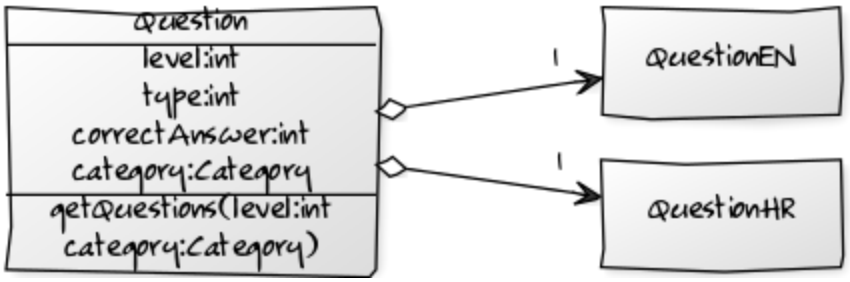


Slika 3. Izgled pitanja select, input, dropdown redom

4. Baza podataka

Baza podataka korištena za pohranu pitanja je **ObjectBox** – objektno-orjentirana NoSQL baza podataka. Entiteti koji su nam bili potrebni su **Question**, **QuestionHR** te **QuestionEN** s obzirom da se u kvizu dohvaćaju pitanja ovisno o postavljenom jeziku uređaja.

Svako pitanje ima svoj $level \in \{1, 2, 3\}$, $type \in \{1, 2, 3\}$ (odgovara načinu na koji se odgovara na pitanje, može biti select, dropbox, input), $correctAnswer \in \{1, 2, 3, 4\}$ (redni broj odgovora koji je točan), $category \in \{history - art, sport, science, geography, movie\}$. Svako pitanje ima svoju inačicu na **hrvatskom** i **engleskom** jeziku što je ilustrirano sljedećom slikom 4.



Slika 4. Prikaz lokalne baze

Na svakom jeziku je zadano 5 atributa, a to su *text* (tekst pitanja), te 4 moguća odgovora: *answer*1, *answer*2, *answer*3, *answer*4.
Dohvat svih pitanja za određeni level te kategoriju omogućen je metodom *getQuestions*().

Kod pokretanja aplikacije, ukoliko u bazi nema pitanja, ili su u međuvremenu promijenjena, baza se inicijalizira odnosno ažurira metodom *App* :: *populateDatabase*().

5. Komunikacija s rails aplikacijom

Backend je pisan u **Ruby on Rails** aplikaciji, te je korišten **heroku** za deploy.
(Ponovo napomena da je potrebno posjetiti https://androidquiz.herokuapp.com/.)

Baza se sastoji od dvije tablice: **users** i **tokens**. U tablici users se nalaze sve informacije o korisniku kao username, email, password_digest, registration_token (korišten za potvrdu registracije), has_registered, score za svaku od kategorija te vrijeme za svaki level po kategoriji.

Svaki korisnik može imati više tokena, te je veza one-to-many ostvarena pomoću tablice tokens gdje je spremljen id usera te token kao string.

Dodali smo sljedeće dependencyje kako bismo mogli izvršavati API pozive te serijalizirati/deserijalizirati objekte koje šaljemo, odnosno primamo.

```
implementation 'com.google.code.gson:gson:2.8.5'
implementation 'com.squareup.retrofit2:retrofit:2.4.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
```

5.1. Registracija

Prvi poziv backenda doziva se već kod registracije koristeći **api.models.User**. Ukoliko je prošla validacija unesenog korisničkog imena, email-a te passworda, poziva se api za stvaranje korisnika POST metodom, te se vraća status code **201 Created** ukoliko je korisnik uspješno stvoren, odnosno **400 Bad Request** ukoliko su email ili username već zauzeti. Kod uspješnog stvaranja korisnika, šalje se mail za potvrdu registracije.

5.2. Login

Za login su potrebni korisničko ime i password. Nakon što je validacija uspješno provedena, poziva se api za kreiranje sessiona POST metodom koristeći model **api.models.Session**. Ukoliko su username ili password neispravni, vraća se status code **400 Bad Request**. Ukoliko su username i password ispravni, međutim korisnik još uvijek nije potvrdio registraciju, vraća se status code **422 Unprocessable Entity**. Te za kraj, ukoliko je mail potvrđen te poslani podaci ispravni, vraća se status code **201 Created**, koji osim trenutnih levela za svaku kategoriju u jsonu šalje i token za trenutni session. U callbacku api calla, u shared preferences se spremaju podaci o korisniku: username, token, te leveli za pojedinu kategoriju.

5.3. Ažuriranje levela

Po završetku kviza, ukoliko je korisnik riješio svih 10 pitanja točno, potrebno je ažurirati level, odnosno povećati ga za 1, koristeći model **api.models.Result**.

Za odigranu kategoriju, level te vrijeme koje je bilo potrebno za rješavanje tog levela, POST metodom se poziva api za update levela. Šalje se kategorija, level, vrijeme te u headeru **Authorization** token korisnika kako bi se mogao utvrditi njegov identitet. Ukoliko je korisnik valjan te je update levela prošao uspješno vraća se status **200 OK**. Ukoliko authorization header nije prisutan ili nije ispravan, vraća se **401 Unauthorized**. Inače, ukoliko je neki podataka neispravan status **400 Bad Request**. Nakon uspješnog updatea levela, level se također updatea i u shared preferences kako bi bio u toku s onim podacima na backendu.

5.4. Lista najboljih igrača

Kod dohvata liste najboljih igrača koristi se model **api.models.LeaderBoard**. Api poziv se vrši metodom GET, koji vraća json statusa **200 OK** koji se sastoji od po 3 najbolja igrača za svaku od 5 kategorija. Najbolji igrač u nekoj kategoriji je definiran kao onaj koji je na najvećem levelu, a potom ako su neki na istom levelu gleda se ukupno vrijeme koje im je bilo potrebno za stići na taj level. Ukoliko ima manje od 3 igrača koja su prošla barem jedan level u toj kategoriji, umjesto imena igrača ispisuje se " - ". U callbacku calla se učitavaju podaci u pripadne TextView-ove.

Na sljedećoj slici 5 se vidi način na koji se traže 3 najbolja korisnika kategorije.

```
def self.best(category)
  User.order("#{category} desc")
      .order("COALESCE(#{category}_1, 0) + COALESCE(#{category}_2, 0) + COALESCE(#{category}_3, 0) asc")
      .where("#{category} > 1").limit(3)
end
```

Slika 5. Pretraga najboljih korisnika kategorije

5.5. Logout

Logout se ostvaruje brisanjem trenutnog sessiona pomoću api calla metodom DELETE. U headeru **Authorization** je potrebno poslati token korisnika koji se logouta, te se taj token briše iz baze na backendu. Neovisno o uspješnosti api call-a, token te ostale informacije o korisniku se brišu iz shared preferences te je user odjavljen iz aplikacije nakon čega se prikazuje ponovno ekran za login.

6. Kontrola verzije koda

Kako bismo mogli istovremeno raditi na projektu koristili smo sustav za verzioniranje git. Cijeli projekt nalazi se na **GitHub**-u: https://github.com/valental/AndroidQuiz

Za polaznu točku napisali smo dvije wiki stranice za GitHub-u.
Jedna se odnosila na aktivnosti u aplikaciji i prijelaze između njih, a druga na lokalnu bazu podataka i bazu podataka na serveru. Tijekom izrade projekta došlo je do nekih promjena, ali na wiki stranicama još se može vidjeti koje su nam bile polazne ideje.