



# DRAGON

*Korisnička i tehnička dokumentacija*

---

Autor: Iva Tadić, 8.r., OŠ Ante Kovačića

Mentor: Mladen Ćurić, dipl. ing. matematike



# 1 Uvod

## 1.1 O meni

Učenica sam 8.a razreda Osnovne škole Ante Kovačić u Zagrebu.

Volim matematiku, programiranje, zmajeve, Rubikovu kocku i Yu-Gi-Oh. Volim rješavati logičke zadatke i osmišljavati algoritme za njihovo rješenje

Počela sam se aktivnije baviti programiranjem u 7. razredu. Prve programčice sam napisala u programskom jeziku Python. Željela sam oduvijek napraviti jednu zabavnu igricu.

## 1.2 Kratko o aplikaciji

Aplikacija DRAGON je igrica čiji je cilj

- zabaviti i
- poboljšati opće znanje korisnika (igrača).

Aplikacija je napravljena kao igra pogađanja. Pogađati može zmaj Drago ili korisnik (igrač).

Kada pogađa zmaj Drago, korisnik zamisli neki pojam iz određenog područja (npr. Europsku državu, lik iz TV serije Teorija velikog praska), a zmaj Drago onda postavlja razna pitanja s namjerom da pogodi što je korisnik zamislio.

Mogući odgovori na pitanja su Da, Ne i Ne znam, a korisnik može odgovoriti pritiskom odgovarajućeg gumba ili glasovno. Zmaj Drago će pokušati pogoditi, a korisnik treba potvrditi je li odgovor točan ili nije. Ako zmaj nije pogodio, korisnik je zatražen da otkrije što je zamislio.

Nakon što je odgovor poznat, na ekran se ispisuje usporedba korisničkih i stvarnih odgovora na pitanja s pojašnjenjima. Tako korisnik može vidjeti što je netočno odgovorio i koji su točni odgovori na pitanja za taj pojam.

Kada pogađa korisnik, zmaj Drago zamisli neki pojam iz određenog područja pa zadaje korisniku hintove (smjernice, asocijacije) kako bi korisnik mogao pogoditi što je zmaj Drago zamislio. Korisnik nakon svakog hinta može zatražiti novi hint ili pokušati pogoditi. Ukoliko korisnik pogodi, ispisuje se lista svih hintova s pojašnjenjima. Ukoliko korisnik nije pogodio, može se predati ili nastaviti čitati hintove sve dok se ne iskoriste svi mogući hintovi koje zmaj Drago zna. Kada se korisnik preda, ispisuje se točan odgovori te lista svih hintova s pojašnjenjima.

Aplikacija DRAGON je hibridna Ionic 4, Angular 7 aplikacija.



## 1.3 Motivacija

Željela sam stvoriti aplikaciju koja je u isto vrijeme zabavna i obrazovna.

## 1.4 Što je dodano u novoj verziji aplikacije

U aplikaciju su dodane nove funkcionalnosti pa je, u skladu s tim, dokument izmijenjen i dopunjen opisima tih funkcionalnosti. Izmjene su sljedeće:

- U poglavlju 1.2. *Kratko o aplikaciji* prilagođen je opis aplikacije jer su u novu verziju aplikacije dodane nove funkcionalnosti: pogađanje korisnika što je zamislio zmaj Drago, detaljnija pojašnjenja određenih odgovora za kandidate
- Poglavlju 2.3. *Odabir kategorije i pokretanje igre* je izmijenjen naslov u 2.3. *Odabir kategorije i tko će pogađati*. Unutar poglavlja je izmijenjena *Slika 5. Izbor kategorija* jer je na stranicu za odabir kategorije dodan lik zmaya Drage koji spava. U skladu s tom izmjenom, prilagođen je i popratni tekst u poglavlju. Poglavlje je dopunjeno opisom novog ekrana za odabir tko će pogađati te je za taj ekran dodana nova slika *Slika 6*.
- Poglavlju 2.4. *Igra pogađanja* je dopunjen naslov u 2.4. *Igra pogađanja – pogađa zmaj Drago* zbog mogućnosti da pogađa ili zmaj Drago ili korisnik, a opis u tom poglavlju se odnosi kao i prije samo na situaciju kada pogađa zmaj Drago. Sam tekst je malo dopunjen kako bi bilo jasno da se opisi odnose na situaciju kada zmaj Drago pogađa.
- Poglavlju 2.5. *Pogodak?* je izmijenjen naziv u 2.5. *Je li Drago pogodio?* kako bi bilo jasno da se opisi u poglavlju odnose na situaciju kada zmaj Drago pogađa.
- Poglavlju 2.6. *Analiza odgovora* je dopunjen naziv u 2.6. *Analiza odgovora – pogađa zmaj Drago* kako bi bilo jasno da se opisi u poglavlju odnose na situaciju kada zmaj Drago pogađa
- U poglavlju 2.6. je izmijenjena *Slika 13. Analiza odgovora* jer su na stranicu dodana pojašnjenja odgovora. Iz istih razloga je dopunjen tekst poglavlja.
- Dodana su nova poglavlja 2.7. *Igra pogađanja – pogađa korisnik*, 2.8. *Novi hintovi* i 2.9. *Pokušaj pogotka korisnika*.
- Dopunjeno i izmijenjeno je poglavlje 3.1.2. *Pohrana kategorija, pitanja i mogućih odgovora* zbog izmjena u strukturi baze podataka (*cathintrule, cqexplain, qhintno, qhintyes, qahintnolength, qahintyeslength, qhintno, qhintyes, qimg*). U skladu s tim, zamijenjene su i sve slike s prikazom strukture baze podataka unutar tog poglavlja.
- Napravljene su izmjene u poglavlju 3.2. *Struktura stranica aplikacije* jer u su dodane nove stranice u novoj verziji aplikacije
- Napravljene su manje izmjene u poglavlju 3.4. *Opis algoritma za pogađanje* zbog uvođenja ekrana s odabirom tko će pogađati. Sam algoritam pa tako i opis algoritma nije mijenjan.
- Dodano je novo poglavlje 3.5. *Opis algoritma za hintove* gdje se opisuje logika ispisa hintova na ekran.



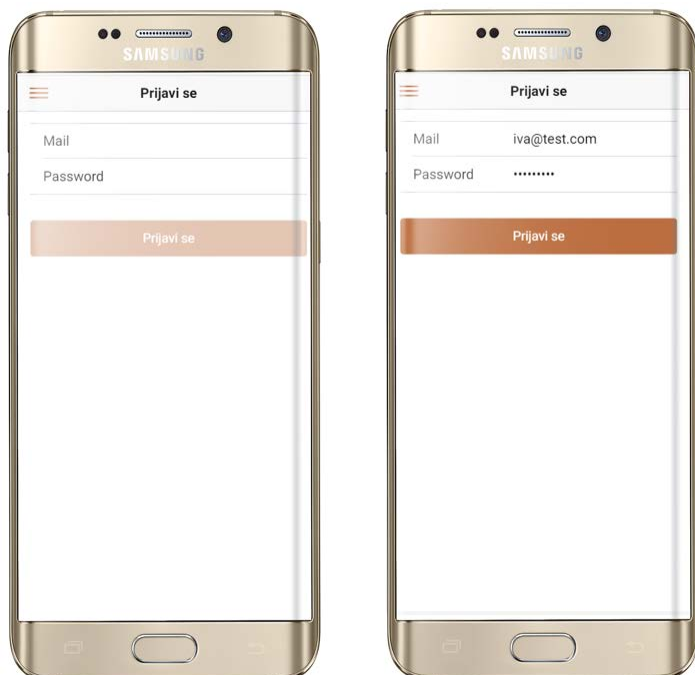
- U poglavlju 3.6. *Analiza odgovora* je izmijenjena *Slika 36.* zbog pojašnjenja koja se sada ispisuju uz analizu odgovora
- U poglavlju 5. *Planovi za budućnost* su izbačene dvije funkcionalnosti zato jer su ovom verzijom aplikacije ostvarene. Dodano je nekoliko novih planova za budućnost.



## 2 Korisnička dokumentacija

### 2.1 Prijava i registracija u aplikaciju

Nakon instalacije aplikacije na uređaj, prilikom prvog pokretanja je nužna prijava korisnika.

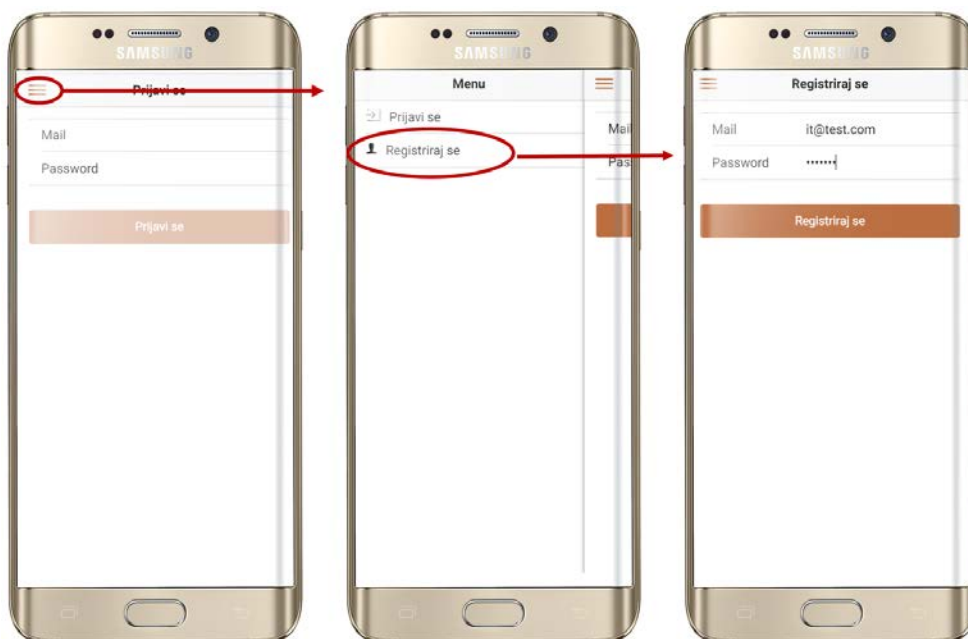


Ukoliko je korisnik već registriran, potrebno je na ekran za prijavu upisati pristupne podatke:

- adresu elektroničke pošte (email) i
- pripadnu lozinku.

Ukoliko korisnik još nije registriran, mora se registrirati. Kako bi se registrirao, potrebno je u gornjem lijevom kutu izabrati *Izbornik* te upisati željene pristupne podatke na ekranu koji se otvorio.

Slika 1 Prijava u aplikaciju

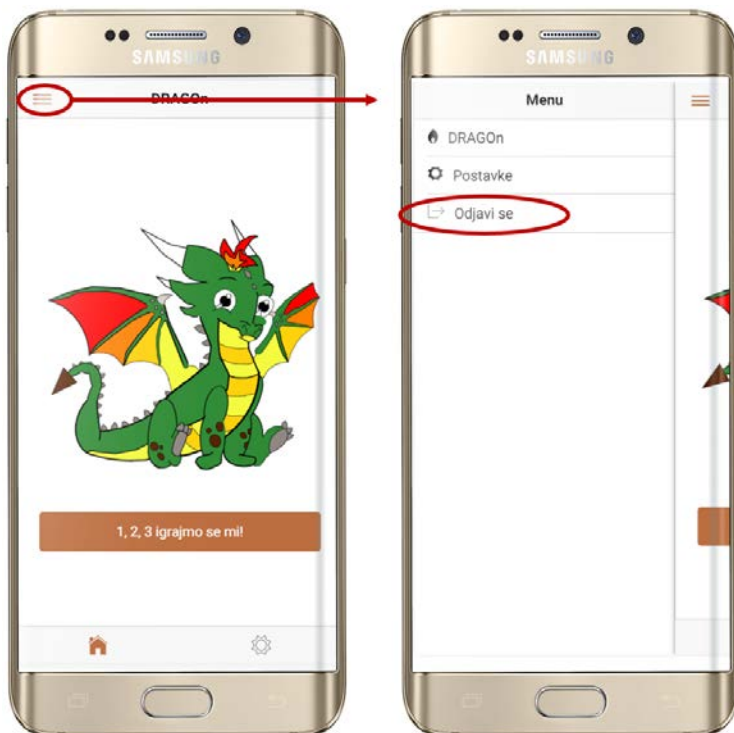


Slika 2 Registracija korisnika



Po završetku registracije, uređaj se odmah prijavi i otvara se početna stranica aplikacije.

Ukoliko se ne radi o prvom pokretanju aplikacije, prilikom otvaranja aplikacije se uređaj automatski spaja koristeći pritom pristupne podatke koje je korisnik upisao prilikom zadnje prijave ili prilikom registracije. Ako se korisnik želi prijaviti s nekim drugim pristupnim podacima, potrebno je da se prvo odjavi. To može učiniti odabirom *Izbornika*.



**Slika 3** Odjava korisnika



## 2.2 Početna stranica



Slika 4 Početna stranica aplikacije

Nakon prijave korisnika, aplikacija otvara početnu stranicu aplikacije.

U centru te stranice nalazi se animirani lik zmaja Drage.

Na vrhu se nalazi naslov i pokraj njega *Izbornik*.

Na dnu ekrana mogu se odabrati dvije opcije:



Gumb *Home* omogućuje korisniku povratak na početni ekran.



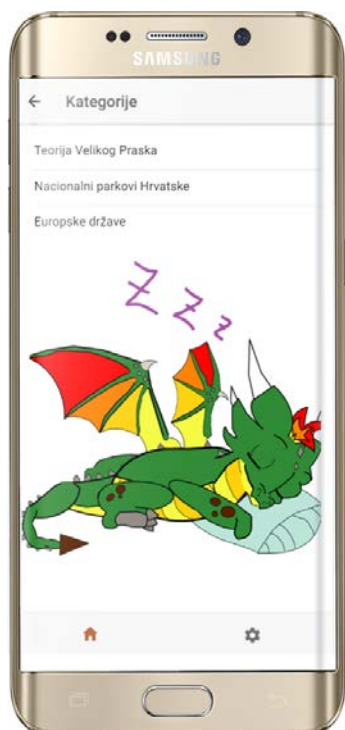
Gumb *Postavke* omogućuje korisniku namještanje boje gumba u igrici te uključivanje i isključivanje prepoznavanja glasa.

Naslov i izbornik na vrhu stranice te dvije opcije na dnu ekrana su prisutne na svim stranicama tijekom cijele igre.

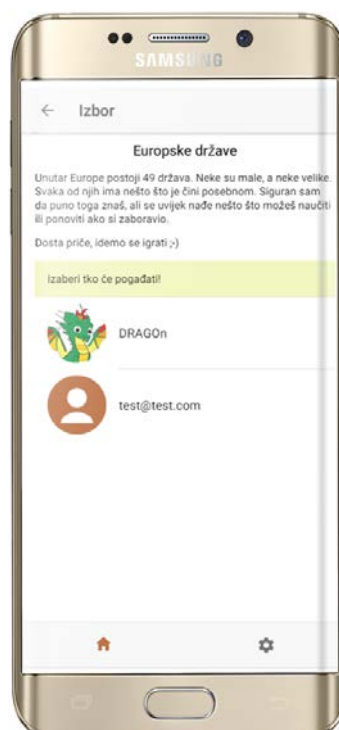
Pritiskom na gumb *1,2,3 igrjmo se mi!* otvorit će se stranica za odabir kategorije.



## 2.3 Odabir kategorije i tko će pogađati



Slika 5 Izbor kategorija



Slika 6 Odabir tko će pogađati

Na stranici za odabir kategorije, korisnik odabire kategoriju – temu čiji pojam će se pogađati (Slika 5).

Trenutno su dostupne tri kategorije:

- Teorija velikog praska
- Nacionalni parkovi Hrvatske
- Europske države

Nakon odabira kategorije, otvara se novi ekran na kojem korisnik odabire tko će pogađati – zmaj Drago ili korisnik (Slika 6).





## 2.4 Igra pogađanja – pogađa zmaj Drago

Nakon odabira kategorije i odabira da pogađa zmaj Drago, otvara se ekran na kojem se traži od korisnika da zamisli pojam iz odabrane kategorije.



**Slika 7** Početni ekran u situaciji kada pogađa zmaj Drago i kada je isključen speech recognition



**Slika 8** Ekran s pitanjem u situaciji kada je isključen speech recognition

Pritiskom na gumb *Započnimo*, započinje pogađanje.

Otvora se ekran na kojem se prikazuje prvo pitanje. Ispod pitanja su ponuđene tri mogućnosti odgovora na pitanje:

- Da
- Ne
- Ne znam

Na vrhu ekrana zapisan je naziv ekrana. U gornjem lijevom kutu nalazi se strelica za povratak na prethodnu stranicu (odabir tko će pogađati).

Ukoliko je korisnik uključio speech recognition, korisnik čuje i audio zapis pitanja te može glasovno odgovarati na pitanja. Na vrhu ekrana, odmah ispod naziva ekrana nalaze se dvije dodatne funkcionalnosti:



Gumb koji služi za ponovno uključivanje mikrofona. Koristi se u slučaju da je korisnik dugo šutio pa se mikrofona isključio.



Gumb koji služi za ponovno pokretanje audio zapisa.



**Slika 9** Početni ekran u situaciji kada pogađa zmaj Drago i kada je uključen speech recognition



**Slika 10** Ekran s pitanjem u situaciji kada je uključen speech recognition



## 2.5 Je li Drago pogodilo?



Slika 11 Pokušaj pogotka



Slika 12 Odabir zamišljenog pojma

Nakon što su postavljena sva potrebna pitanja za pogađanje zamišljenog pojma, otvorit će se stranica s pokušajem pogotka.

U centru stranice se nalazi naziv pojma za kojeg zmaj Drago misli da je zamišljen i slika vezana uz taj pojam.

Ispod toga se nalazi pitanje *Jesam li pogodio?* i od korisnika se traži da potvrdi ili opovrgne zmajev pogodak. Ukoliko je zmaj pogodilo, učitat će se stranica s analizom odgovora. Ukoliko zmaj nije pogodilo, korisnik se traži da otkrije koji je pojam zamislio pa se nakon tog radi analiza odgovora.



## 2.6 Analiza odgovora – pogađa zmaj Drago



Slika 13 Analiza odgovora

Na ovoj se stranici za sva postavljena pitanja uspoređuje točan odgovor za državu koja je rješenje s odgovorom koji je dao korisnik. Uz usporedbe odgovora, ispisuje se i pojašnjenje odgovora. U nekim slučajevima pojašnjenje odgovora ne postoji jer je očito i nema smisla pojašnjavati. (npr. *Radi li se o ženskoj osobi?*)

Tako je omogućeno da korisnik nauči ono što nije znao o toj državi ili da korigira ono što misli da je znao.

Ispod analize odgovora na sva pitanja nalazi se gumb

IGRAJMO SE OPET!

Pritiskom tog gumba počinje nova igra.

## 2.7 Igra pogađanja – pogađa korisnik

Nakon odabira kategorije i odabira da pogađa korisnik, otvara se ekran na kojem zmaj Drago kaže da je zamislio pojam iz odabrane kategorije. Pritiskom na gumb *Započnimo*, započinje pogađanje.



**Slika 14** Početni ekran u situaciji kada pogađa korisnik i kada je isključen speech recognition





**Slika 15** Ekran s hintom u situaciji kada je isključen speech recognition

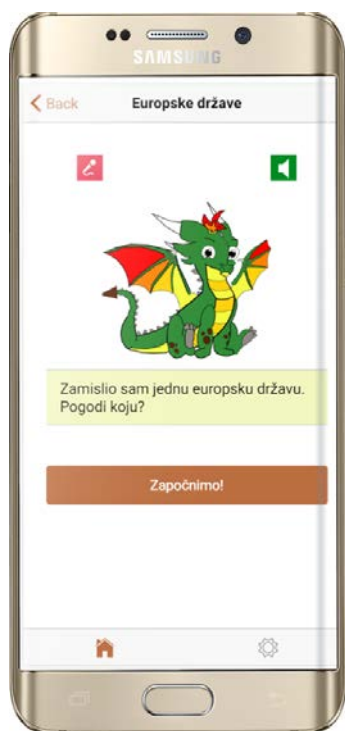
Otvora se ekran na kojem se prikazuje prvi hint. Ispod hinta su ponuđene dvije mogućnosti:

- Znam – odabirom ove mogućnosti, korisnik može pokušati pogoditi.
- Dalje – odabirom ove mogućnosti, zmaj Drago će ispisati novi hint.

Na vrhu ekrana zapisan je naziv ekrana. U gornjem lijevom kutu nalazi se strelica za povratak na prethodnu stranicu (odabir tko će pogađati).

Ukoliko je korisnik uključio speech recognition, korisnik čuje i audio zapis hinta te može glasovno zadavati daljnje korake (Znam ili Dalje). Na vrhu ekrana, odmah ispod naziva ekrana nalaze se dvije dodatne funkcionalnosti:

-  Gumb koji služi za ponovno uključivanje mikrofona. Koristi se u slučaju da je korisnik dugo šutio pa se mikrofona isključio.
-  Gumb koji služi za ponovno pokretanje audio zapisa.



**Slika 16** Početni ekran u situaciji kada pogađa korisnik i kada je uključen speech recognition



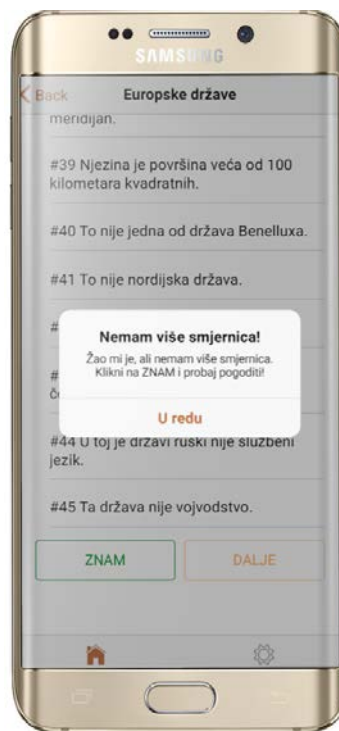
**Slika 17** Ekran s hintom u situaciji kada je uključen speech recognition



## 2.8 Novi hintovi



**Slika 18** Na ekranu ostaju zapisani svi hintovi



**Slika 19** Nestalo je hintova i korisnik mora pogađati

Svaki puta kada korisnik odabere mogućnost *Dalje*, zmaj Drago ispisuje novi hint. Hintovi se ispisuju sve dok zmaj Drago ne potroši sve svoje upute.

Svi hintovi su cijelo vrijeme ispisani na ekranu kako bi se olakšalo korisniku pogađanje. (Slika 16)

U trenutku kada zmaju Dragi ponestane daljnjih uputa, iskoči pop up prozor koji informira korisnika da mora pokušati pogoditi. (Slika 17)



## 2.9 Pokušaj pogotka korisnika

Odabirom mogućnosti *Znam*, korisnik pokušava pogoditi koji je pojam zmaj Drago zamislio. Na ekranu se ispisuju sve mogućnosti i korisnik treba odabrati jednu.



Slika 20 Pokušaj pogotka korisnika

Nakon što je korisnik odabrao jedan pojam, zmaj Drago ispisuje je li pogodak točan.

Ukoliko je korisnik pogodio, otvara se stranica s popisom svih hintova i s pojašnjenjima vezanim uz pojedini hint i uz pogođenog kandidata.

Ukoliko korisnik nije pogodio, korisnik ima mogućnost nastaviti pogađati ili se može predati. U slučaju daljnjeg pogađanja, zmaj Drago nastavlja davati hintove. U slučaju predaje, zmaj Drago će otkriti korisniku koji je pojam zamislio te će se na ekranu ispisati pojašnjenja za svaki hint osim gdje je očito i nema potrebe za pojašnjenjem.





Slika 21 Korisnik je pogodio



Slika 22 Korisnik nije pogodio

Možda da se vratiš i odigraš igru do kraja? ;-)



Slika 23 Korisnik se može 'predati' ili nastaviti igru dalje ...



## 3 Tehnička dokumentacija

### 3.1 Baza podataka

U aplikaciji DRAGON se koristi baza podataka Firebase Realtime Database i to za sljedeće potrebe:

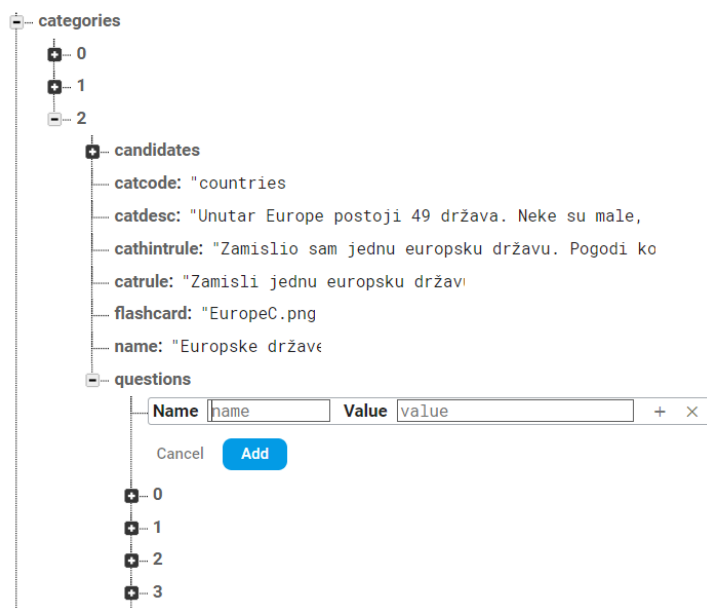
- autentifikaciju korisnika
- pohranu kategorija, pitanja i mogućih odgovora
- pohranu odgovora korisnika

Svi navedeni podaci su u bazu podataka spremljeni kao JSON objekti pa je baza podataka strukturirana kao stablo.



**Slika 24** Slika stablaste strukture u bazi podataka

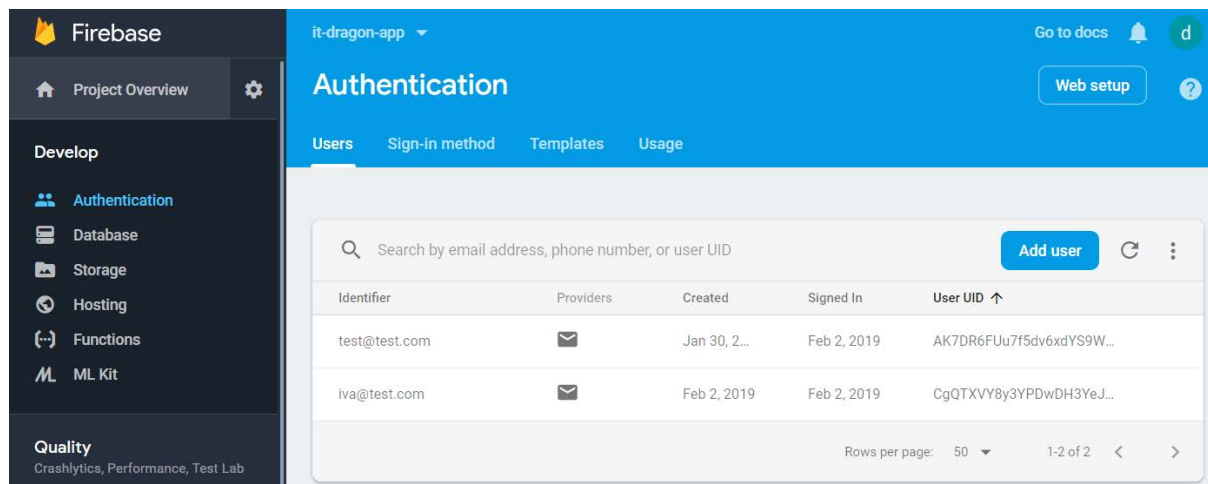
Dodavanjem novog podatka u bazu, stvara se novi čvor u stablu, koji ima određeni identifikator.



Slika 25 Dodavanje novog podatka u bazu podataka

### 3.1.1 Autentifikacija korisnika

Svi korisnici aplikacije su spremljeni u bazu u dio naziva *Authentication*. Kada aplikacija autentificira korisnika, on temeljem njegovih pristupnih podataka (mail adresa i lozinka) traži redak u listi svih postojeći korisnika. Ukoliko ga nađe, otvara token putem kojeg korisnik dalje koristi aplikaciju.



Slika 26 Lista korisnika aplikacije u bazi podataka

### 3.1.2 Pohrana kategorija, pitanja i mogućih odgovora

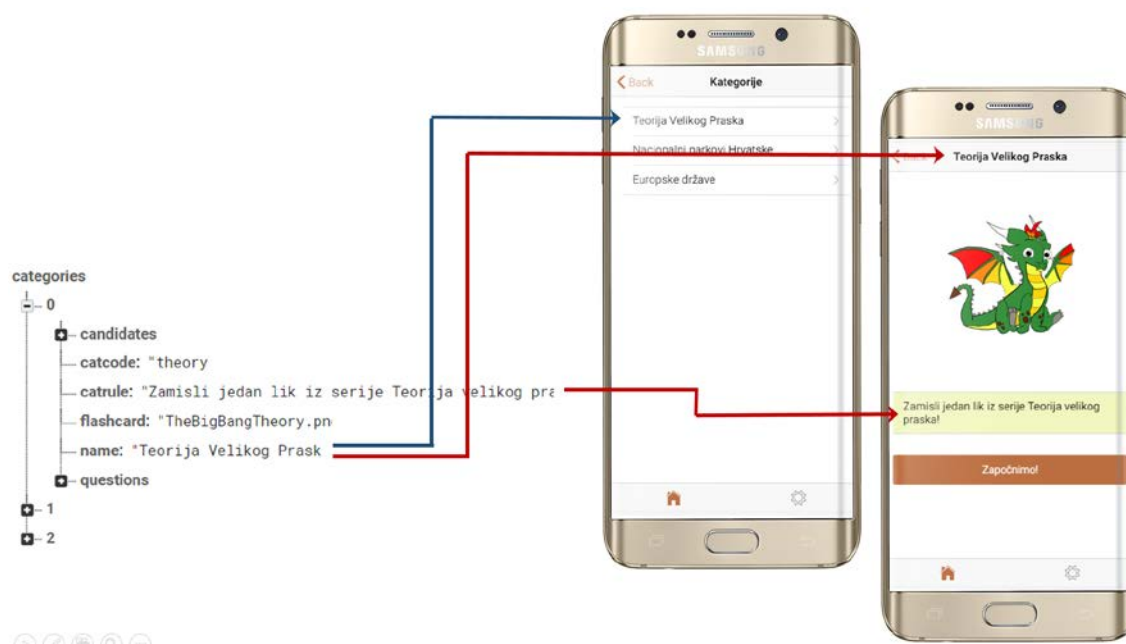
U bazi podataka su pohranjeni svi potrebni podaci za igru pogađanja.



Podatkovna struktura *categories* sadrži popis svih mogućih kategorija unutar kojih korisnik može pogađati.

Svaka kategorija ima svoj čvor kojem je dodijeljen identifikator. Podaci koji postoje za svaku kategoriju su sljedeći:

- *catcode* – šifra kategorije, koristi se za naziv mape u koju je spremljena dodatna medija vezana uz tu kategoriju, npr. zvučni zapisi, slike, ...
- *name* – naziv kategorije, pojavljuje se na ekranu kod odabira kategorije
- *flashcard* – naziv slike koja će se pojaviti iznad svakog pitanja unutar ove kategorije
- *catdesc*
- *catrule* – tekst koji se za tu kategoriju ispisuje na početku igre u slučaju kada zmaj Drago pogađa
- *cathintrule* – tekst koji se za tu kategoriju ispisuje na početku igre u slučaju da korisnik pogađa
- *candidates* – lista svih mogućih odgovora tj. skupina kandidata koje je moguće pogoditi; primjerice, za kategoriju *Europske države*, ovdje će biti lista svih država koje teritorijalno pripadaju u Europu; ovo je složena struktura (pojašnjena niže u tekstu)
- *questions* – lista svih mogućih pitanja/hintova koja zmaj Drago može postaviti/zadati korisniku tijekom pogađanja; ovo je složena struktura (pojašnjena niže u tekstu)



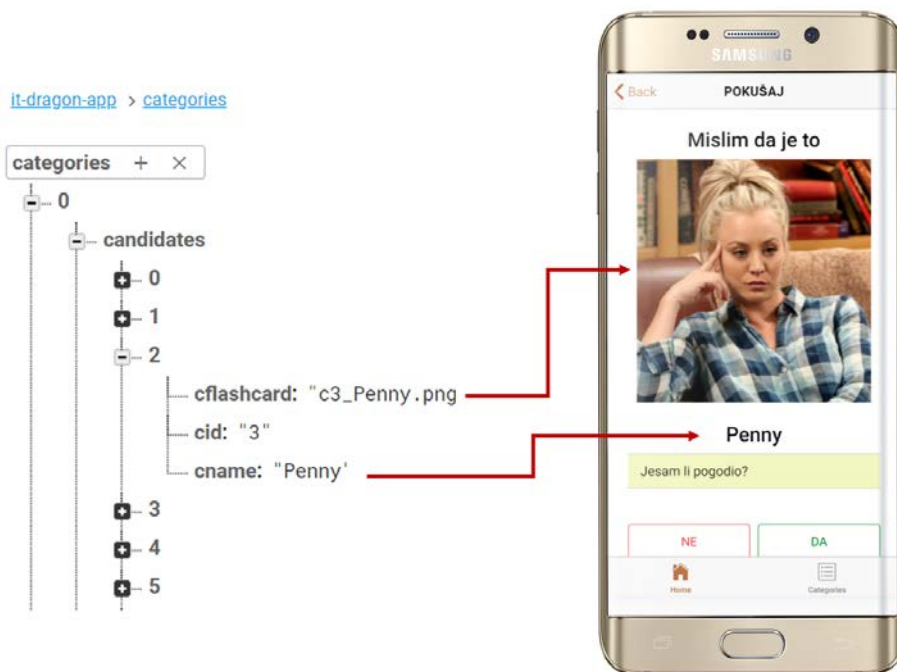
**Slika 27** Kategorije i podaci koji postoje unutar svake pojedine kategorije u bazi podataka

Podatkovna struktura *candidates* sadrži popis svih mogućih odgovora (kandidata za pogodak) unutar određene kategorije. Svaki kandidat ima svoj čvor koji sadrži sljedeće atribute:

- *cid* – identifikator kandidata koji se koristi za povezivanje s određenim pitanjima
- *cname* – naziv kandidata koji se ispisuje na ekran prilikom pogotka



- *cflashcard* – naziv slike koja se prikazuje iznad teksta odgovora
- *cqexplain* – lista koja sadrži pojašnjenja odgovora na svako pitanje za tog kandidata; ovo je složena struktura (pojašnjena niže u tekstu)



Slika 28 Kandidati i dodatni elementi kandidata

Podatkovna struktura *cqexplain* sadrži listu parova:

- *qid* – identifikator pitanja
- *explain* – tekst s pojašnjenjem odgovora na pitanje s identifikatorom *qid* za kandidata *cid* unutar kojeg se nalazi cijela lista *cqexplain*

Podatkovna struktura *questions* sadrži listu svih mogućih pitanja koja zmaj Drago može postaviti korisniku tijekom pogađanja. Svako pitanje sadrži sljedeće atribute:

- *qid* – identifikator pitanja, koristi se unutar algoritma pogađanja te za dohvat odgovarajućih multimedijских zapisa, npr. audio zapis za pitanje s *qid*=8 je *q8.m4a*
- *qtext* – tekst pitanja koji se ispisuje na ekranu
- *qinit* – oznaka ima li smisla ovo pitanje postaviti kao prvo pitanje; kako bi algoritam radio optimalno, ima najviše smisla prvo postaviti upravo ona pitanja koja kandidate podijele u skupove s približno jednakim brojem članova. Takav pristup omogućuje najmanji broj koraka do rješenja. Stoga ima smisla npr. pitati *Radi li se o ženskoj osobi?* ukoliko pogađamo osobe iz BBT jer je ukupan broj kandidata 36, od čega 18 ženskih osoba i 18 muških osoba.



- *qclist* – lista s popisom identifikatora kandidata (*cid*) za koje je odgovor na ovo konkretno pitanje DA; ova je lista vrlo značajan za rad samog algoritma (opisano u poglavlju 0)
- *qalength* – informacija o duljini audio zapisa koja se koristi kako bi se znalo kada treba uključiti mikrofonski (detaljnije opisano u poglavlju 0)
- *qhintno* – tekst *pozitivnog* hinta tj. tekst hinta u slučaju kada je odgovor na pitanje DA (npr. za pitanje 'Protječe li kroz tu zemlju Dunav?', pozitivni hint bi bio 'Kroz tu zemlju protječe Dunav.')
- *qhintyes* - tekst *negativnog* hinta tj. tekst hinta u slučaju kada je odgovor na pitanje NE (npr. za pitanje 'Protječe li kroz tu zemlju Dunav?', negativni hint bi bio 'Kroz tu zemlju ne protječe Dunav.')
- *qahintnolength* – duljina audio zapisa za tekst pozitivnog hinta
- *qahintyeslength* – duljina audio zapisa za tekst negativnog hinta
- *qimg* – naziv slike zmaja koja se prikazuje uz pojedino pitanje

[it:dragon-app](#) > [categories](#)

categories

```

- 0
- 1
- 2
  - candidates
    - catcode: "countries"
    - flashcard: "EuropeC.png"
    - icon: "boat"
    - name: "Europske države"
  - questions
    - 0
    - 1
    - 2
    - 3
    - 4
    - 5
      - qalength: "3"
      - qclist
      - qid: "6"
      - qinit: "No"
      - qtext: "Je li poznata po proizvodnji čokolac"
    - 6

```



Slika 29 Pitanja i podaci koji postoje unutar svakog pitanja u bazi podataka



Slika 30 Struktura *qclist*

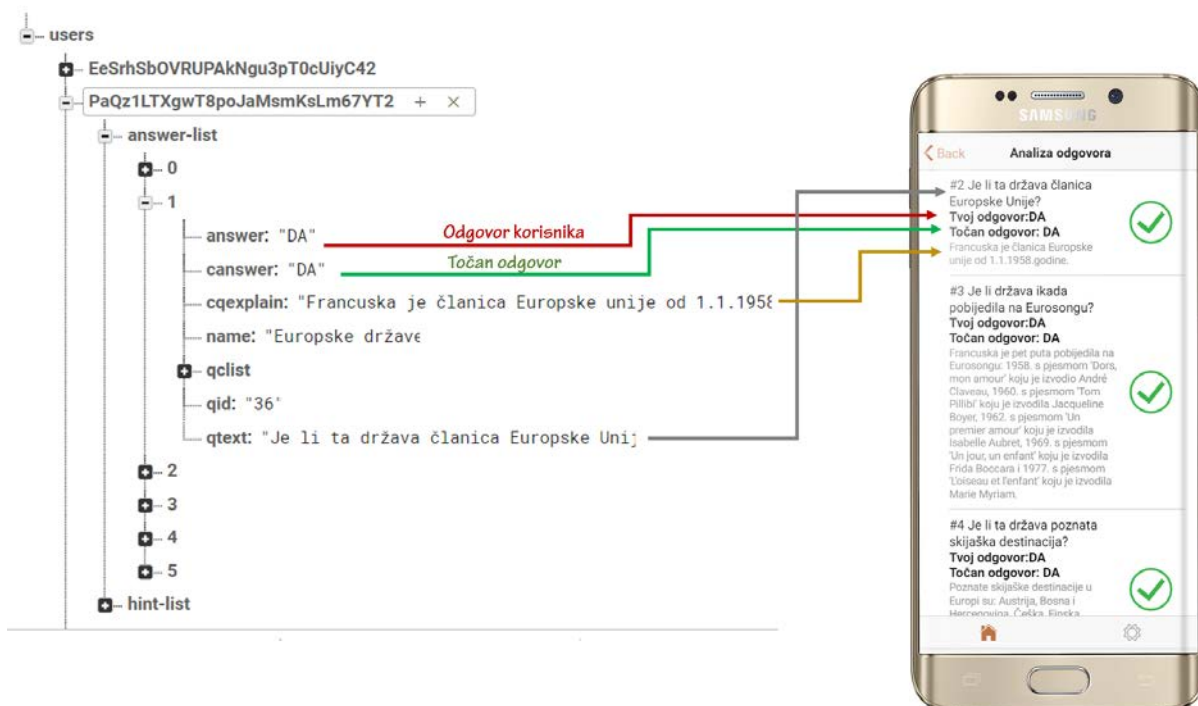
### 3.1.3 Pohrana odgovora korisnika

Kada korisnik odgovori na sva pitanja i zmaj Drago misli da zna odgovor, svi odgovori se spremaju u bazu podataka u složenu podatkovnu strukturu *user/firebase\_identifikator\_usera/answer-list*, npr. na Sliku 20 je to *user/AK7DR6FUu7f5dv6xdYS9WQKTs7c2/answer-list*).

Podatkovna struktura *answer-list* sadrži:

- *name* – naziv kategorije
- *qid* – identifikator pitanja
- *qtext* – tekst pitanja
- *qclist* – popis identifikatora kandidata (*cid*) za koje je odgovor na ovo pitanje DA
- *answer* – odgovor kojeg je dao korisnik na ovo pitanje
- *answer* – točan odgovor na ovo pitanje za kandidata koji je rješenje potvrđeno od korisnika
- *cxexplain* – pojašnjenje vezano uz pojedino pitanje i pojedinog kandidata

Podaci se koriste za izradu *Analize odgovora*.

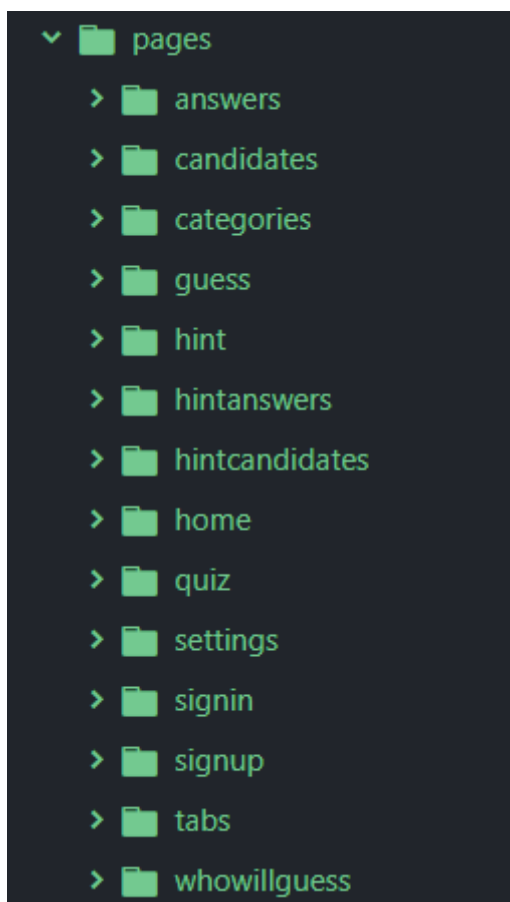


Slika 31 Podatkovna struktura `answer-list` u bazi podataka





## 3.2 Struktura stranica aplikacije



Osnovne stranice u aplikaciji su:

- home – Ovo je početna stanica na kojoj se nalazi lik zmaja Drage i gumb koji nas vodi na sljedeću stranicu categories.
- categories – Ovo je stranica koja prikazuje popis svih postojećih kategorija i omogućuje odabir jedne. Prilikom punjenja ove stranice, aplikacija se spaja na bazu podatak i puni podatkovne strukture unutar koda aplikacije. Odabirom pojedine kategorije, aplikacija vodi na sljedeću stranicu quiz
- quiz – Ovo je stranica koja je napravljena kao slider.
- guess – Ovo je stranica na kojoj se ispisuje pogodak.
- hint – Ovo je stranica koja prikazuje hintove na ekranu.

Njihova struktura je prikazana na slici.



### 3.3 Podatkovna struktura unutar koda aplikacije

Unutar koda aplikacije, podatkovna struktura je definirana unutar mape *models* i odgovara podatkovnoj strukturi koja je definirana unutar baze podataka.

- **candidate.ts** definira klasu *Candidate* koja se koristi za dohvat i korištenje podataka o kandidatima iz baze podataka. Svojom strukturom odgovara podatkovnoj strukturi *candidates* iz baze podataka koja je detaljno opisana u poglavlju 3.1.2.
- **question.ts** definira klasu *Question* koja se koristi za dohvat i korištenje pitanja iz baze podataka. Svojom strukturom odgovara podatkovnoj strukturi *questions* iz baze podataka koja je detaljno opisana u poglavlju 3.1.2.
- **category.ts** definira klasu *Category* koja se koristi za dohvat i korištenje kategorija iz baze podataka. Svojom strukturom odgovara podatkovnoj strukturi *categories* iz baze podataka koja je detaljno opisana u poglavlju 3.1.2.
- **answer.ts** definira klasu *Answer* koja se koristi za upis i dohvat odgovora korisnika. Svojom strukturom odgovara podatkovnoj strukturi *answer-list* iz baze podataka koja je detaljno opisana u poglavlju 3.1.3.3.1.2
- **questionvalue.ts** definira klasu *QuestionValue* koja se koristi unutar algoritam za odabir sljedećeg pitanja (više o tome u poglavlju 0).

Nakon što je u aplikaciji pritisnut gumb *1,2,3 igrajmo se mi!*, učitava se stranica s popisom kategorija (*categories*). Prilikom učitavanja stranice s popisom kategorija, aplikacija se spaja na Firebase Realtime Database i dohvaća sve podatke koji se nalaze ispod čvora *categories*. Podatke spremi u objekt **categories**.

Tim se podacima aplikacija koristi tijekom cijele igre pogađanja.



## 3.4 Opis algoritma za pogađanje

Algoritam za pogađanje se nalazi u mapi *quiz*, u datoteci *quiz.ts*.

### Početak algoritma (START)

Algoritam započinje kada korisnik na stranici *whowillquess* izabere da će zmaj Drago pogađati pojam kojeg će on (korisnik) zamisliti. Počinje učitavanje nove stranice *quiz* i njoj se prosljeđuju svi podaci vezani uz odabranu kategoriju iz objekta *categories*. Pokreće se metoda [ngOnInit](#).

### Kreiranje pomoćnih listi

Za rad algoritma potrebne su dvije nove liste. One se kreiraju i postavljaju na početne vrijednosti unutar metode [ngOnInit](#).

- Kreira se nova lista *myQuestions* koja će sadržavati identifikatore mogućih pitanja tj. onih pitanja koja ima smisla postaviti. Lista će se kasnije tijekom igre ažurirati na način da se iz nje izbacuju već postavljena pitanja, ali i ona pitanja koja još nisu postavljena, ali ih više nema ni smisla postaviti. Primjerice, ako je zmaj Drago već pitao korisnika radi li se o ženskoj osobi i dobio odgovor DA, tada nema smisla pitati radi li se o muškoj osobi. Ova se lista u algoritmu koristi za odabir sljedećeg pitanja uz iznimku prvog pitanja koje se ne bira iz ove liste. Na početku se lista *myQuestions* napuni identifikatorima svih pitanja jer prije ijednog postavljenog pitanja ima smisla postaviti bilo koje pitanje iz ukupne liste pitanja (*category.questions*).
- Kreira se nova lista za moguće kandidate, *myCandidates*, koja će sadržavati identifikatore svih onih kandidata koji su možda baš onaj kojeg je korisnik zamislio. Na samom početku, kada nema još ni jednog odgovora, traženi zamišljeni kandidat može biti bilo koji od ukupne liste kandidata pa se lista *myCandidates* napuni identifikatorima svih kandidata iz ukupne liste kandidata (*category.candidates*).

```
// lista mogućih pitanja
this.myQuestions=[];
for (let item of this.category.questions) {
  this.myQuestions.push(item.qid);
}
console.log("Početna lista pitanja:"+this.myQuestions);

// lista mogućih kandidata
this.myCandidates=[];
for (let item of this.category.candidates) {
  this.myCandidates.push(item.cid);
}
console.log("Početna lista kandidata:"+this.myCandidates);
}
```



## Odabir prvog pitanja

Odabir prvog pitanja događa se unutar metode `ngOnInit`.

- U zasebni objekt naziva `category.candidates` izdvoje se sva moguća pitanja (zajedno sa svim atributima).
- Kreira se nova lista `initQuestions` u koju se spremne identifikatori svih pitanja koja mogu biti početna. Početna pitanja su ona pitanja koja su u bazi podataka i u objektu `category.questions` označena sa `qinit=„Yes“`. Ne postoji samo jedno početno pitanje nego je takvih pitanja više. Razlog tome je da igra bude zanimljivija tj. da ne počinje uvijek od jednog te istog pitanja. Nisu ni sva pitanja označena kao početna jer se želi izbjeći situacija u kojoj je prvo pitanje neko tipično samo za jednog kandidata (npr. *Je li ta država vojvodstvo?*<sup>1</sup>).
- Slučajnim odabirom se odabere jedno od početnih pitanja iz objekta `initQuestions` i njegov identifikator se spremi u varijablu `initQuestion`.

Kada korisnik pritisne gumb „Započnimo“ na ekranu quiz, pokreće se metoda `goToSlide`. Ulazni parametar metode je identifikator pitanja. Kod prvog poziva, taj identifikator je upravo onaj iz varijable `initQuestion` tj. to je identifikator odabranog prvog pitanja.

## Postavljanje pitanja

Unutar metode `goToSlide`, prema dobivenom ulaznom parametru se dohvati pitanje sa svim detaljima te se promijeni slide kako bi korisnik vidio odgovarajuće pitanje.

Unutar objekta `category.candidates` pronađe se pitanje s identifikatorom koji je zaprimljen kao ulazni parametar i spremi se u novi objekt `sQuestion` koji čuva trenutno odabrano pitanje sa svim njegovim detaljima.

Otključa se mogućnost pomicanja slide-ova, pomakne se na odgovarajući slide - onaj koji sadrži pitanje s dobivenim ulaznim identifikatorom i na kraju se ponovo zaključa mogućnost pomicanja slide-ova (korisnik ne može potezom prsta po ekranu promijeniti prikaz).

Korisnik sada vidi novo pitanje na ekranu (ili prvo pitanje ukoliko se radi o prvom pozivu ove metode).

Ukoliko korisnik ima u postavkama uključen Speech Recognition, pokrene se audio zapis pitanja.

Ispod prikazanog pitanja, korisnik može pritiskom na gumb ili glasovno (ukoliko je u postavka uključen speech recognition) odgovoriti na pitanje. Kada je uključen speech recognition, sami gumbi su neaktivni do trenutka završetak audio zapisa.

---

<sup>1</sup> Luksemburg je jedina država u Europi koja je vojvodstvo.



## Spremanje odgovora

Kada je korisnik odgovorio na pitanje (pritiskom na gumb ili glasovno) poziva se metoda `selectAnswer` kojoj se prosljeđuju identifikator pitanja na koje je upravo odgovoreno (*qid*) i sam odgovor (*DA, NE ili NE ZNAM*).

U objekt `alService` se pospremi pitanje iz `sQuestion` i dobiveni korisnički odgovor. Vrijednosti ovog objekta će se koristiti kasnije za upis u bazu podatak i za analizu odgovora.

## Izbacivanje postavljenog pitanja iz liste mogućih pitanja

Unutar metode `selectAnswer` se iz liste mogućih pitanja `myQuestions` izbacuje pitanje koje je upravo postavljeno i na koje je dobiven odgovor. Ovo je nužno napraviti kako se ne bi dogodilo da zmaj Drago ponovo postavlja pitanje na koje je već dobio odgovor.

## Smanjivanje broj kandidata nakon odgovora

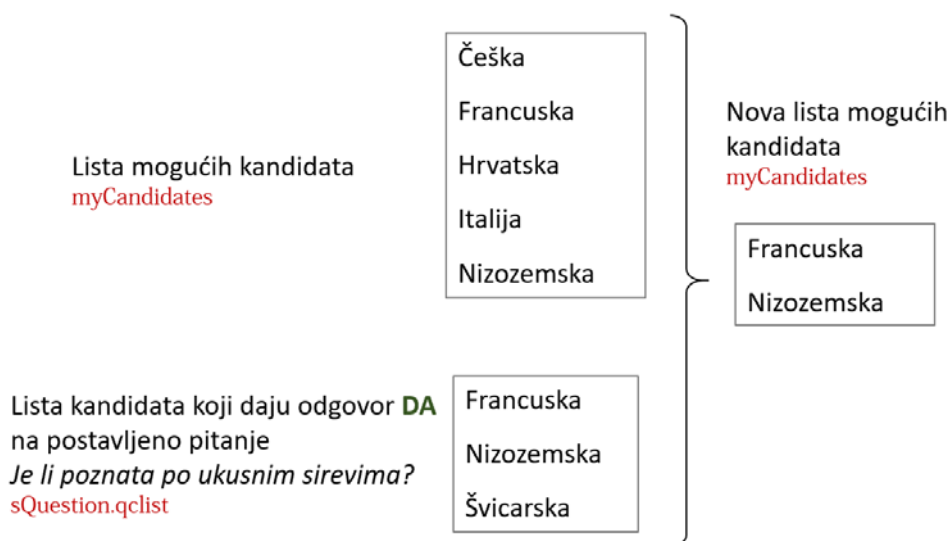
Unutar metode `selectAnswer` se mijenja popis mogućih kandidata koji se nalaze unutar liste `myCandidates`. Način promjene popisa ovisi o samom odgovoru korisnika.

### Odgovor na postavljeno pitanje je DA.

```
this.myCandidates=this.myCandidates.filter(item => -1 !== this.sQuestion.qclist.indexOf(item));
```

- Uspoređuju se redom svi kandidati iz liste mogućih kandidata `myCandidates` s kandidatima koji su zapisani unutar `sQuestion.qclist` (pitanje na koje je dobiven odgovor), a to su kandidati za koje vrijedi da je odgovor na postavljeno pitanje DA.
- Kandidati koji ne postoje u obje liste se izbacuju iz liste, a kandidati koji postoje u obje liste se pospremaju u novu vrijednost liste `myCandidates`.

Zapravo se radi presjek između moguće liste kandidata (oni koji su mogući odgovor) i onih koji preostaju kada se dobije odgovor.



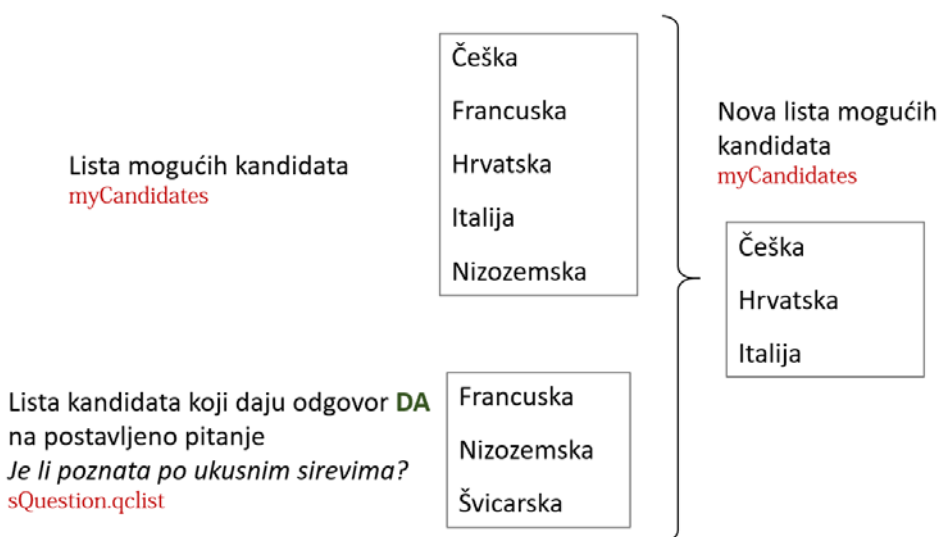
**Slika 32** Osvježavanje liste mogućih kandidata u slučaju odgovora DA

Odgovor na postavljeno pitanje je **NE**.

```
this.myCandidates=this.myCandidates.filter(item => this.sQuestion.qclist.indexOf(item) < 0)
```

- Iz liste mogućih kandidata `myCandidates` izbacit će se svi kandidati koji su zapisani unutar `sQuestion.qclist` tj. kandidati za koje vrijedi da je odgovor na postavljeno pitanje DA i koji više nisu kandidati za pogodak jer je odgovor na pitanje NE.

Ovo je zapravo razlika liste `myCandidates` i liste `sQuestion.qclist`.



**Slika 33** Osvježavanje liste mogućih kandidata u slučaju odgovora NE

Odgovor na postavljeno pitanje je **NE ZNAM**.

Lista `myCandidates` se ne mijenja jer zapravo nemamo novu informaciju od korisnika koju bi iskoristili za smanjenje broja mogućih kandidata.



### Provjera broja preostalih pitanja

Provjerava se duljina liste mogućih pitanja `myQuestions`. Namjera je odrediti ima li barem još jedno pitanje koje bi zmaj Drago mogao postaviti. Naime, moguće je da je korisnik odgovorio „*Ne znam*“ na dovoljno veliku količinu pitanja te da se lista mogućih kandidata nije smanjila na jednog kandidata, a zmaj je već postavio sva smisljena pitanja. U tom slučaju, pogađanje će stati, a sljedeći korak je 'završetak pogađanja' tj. algoritam završava, a zmaj će pokušati pogoditi. Ukoliko je duljina liste strogo veća od nule, znači da postoji još pitanja i igra pogađanja se nastavlja.

### Provjera broja preostalih mogućih kandidata

Provjerava se duljina liste mogućih kandidata `myCandidates` (i dalje unutar metode `selectAnswer`) i sprema se u varijablu `n`. Ukoliko je duljina liste jednaka 1, znači da je preostao samo jedan mogući kandidat i zbog toga, zmaj Drago treba pokušati pogoditi. U tom slučaju se prekida pogađanje i sljedeći korak algoritma je 'završetak pogađanja'.

Ukoliko je duljina liste mogućih kandidata strogo veća od 1, znači da postoji više kandidata koji su moguća rješenja pa je potrebno odabrati novo pitanje.

### Određivanje liste s mjerom podjele kandidata

Dohvaćaju se identifikatori preostalih pitanja iz liste `myQuestions` i za svakog se dohvati pitanje sa svim detaljima iz `category.questions` te se traži presjek od `qclist` tog pitanja s listom mogućih kandidata `myCandidates`. Tom se presjeku odredi duljina (varijabla `nintersect`). Ako presjek postoji (`nintersect > 0`):

- u novu listu `candQuestions` spremi se identifikator pitanja (`qid`) –ovo lista se kasnije koristi kako bi se iz liste mogućih pitanja izbacila sva pitanja čiji `qclist` nema presjek s listom `myCandidates`
- u novu listu `lookUpQuestions` se spremi kao novi član uređeni par koji se sastoji od:
  - identifikatora pitanja (`qid`),
  - mjere udaljenosti od idealne podjele skupa na dva dijela s istim brojem članova (u daljnjem tekstu: mjera podjele kandidata)

Mjera podjele kandidata se određuje na sljedeći način:

$$\text{apsolutna vrijednost} \left( \text{duljina presjeka} - \frac{\text{duljina liste mogućih kandidata}}{2} \right)$$

tj.

$$\text{abs}(\text{nintersect} - \frac{n}{2})$$



Što je mjera podjela kandidata manja, to je pitanje s tom mjerom pogodnije za nastavak pogađanja.

Naime, kako bi algoritam radio optimalno, želi se uvijek postaviti pitanje koje će bez obzira na to je li odgovor *DA* ili *NE*, odbaciti maksimalni broj preostalih mogućih kandidata. Ako na početku imamo skup od 200 mogućih kandidata, želimo postaviti pitanje koje će i u slučaju odgovora *DA* i u slučaju odgovora *NE* odbaciti 100 kandidata. Za taj je slučaj mjera podjele kandidata jednaka nuli.

### Izbacivanje pitanja koja nemaju smisla iz liste mogućih pitanja

Nakon punjenja novih listi, mijenja se još jednom lista *myQuestions* i to na način da u njoj ostaju samo oni identifikatori pitanja koji postoje i u listi *candQuestions*. Na taj način su iz liste mogućih pitanja *myQuestions* izbačena ona pitanja čije *qclist* nema presjek s preostalim mogućim kandidatima tj. ona koja nisu smisljena. (Nema smisla pitati radi li se o muškoj osobi ako su svi preostali kandidati žene.)

### Odabir sljedećeg pitanja

Nakon što je lista *lookUpQuestions* napunjena, lista se sortira od najmanjeg prema najvećem po mjeri podjele kandidata.

Kao sljedeće pitanje odabire se ono s *qid* od prvog člana iz liste *lookUpQuestions* (to je ono s najmanjom mjerom podjele kandidata nakon sortiranja) te se poziva metoda *goToSlide*. Algoritam se vraća na korak 'postavljanje pitanja'.

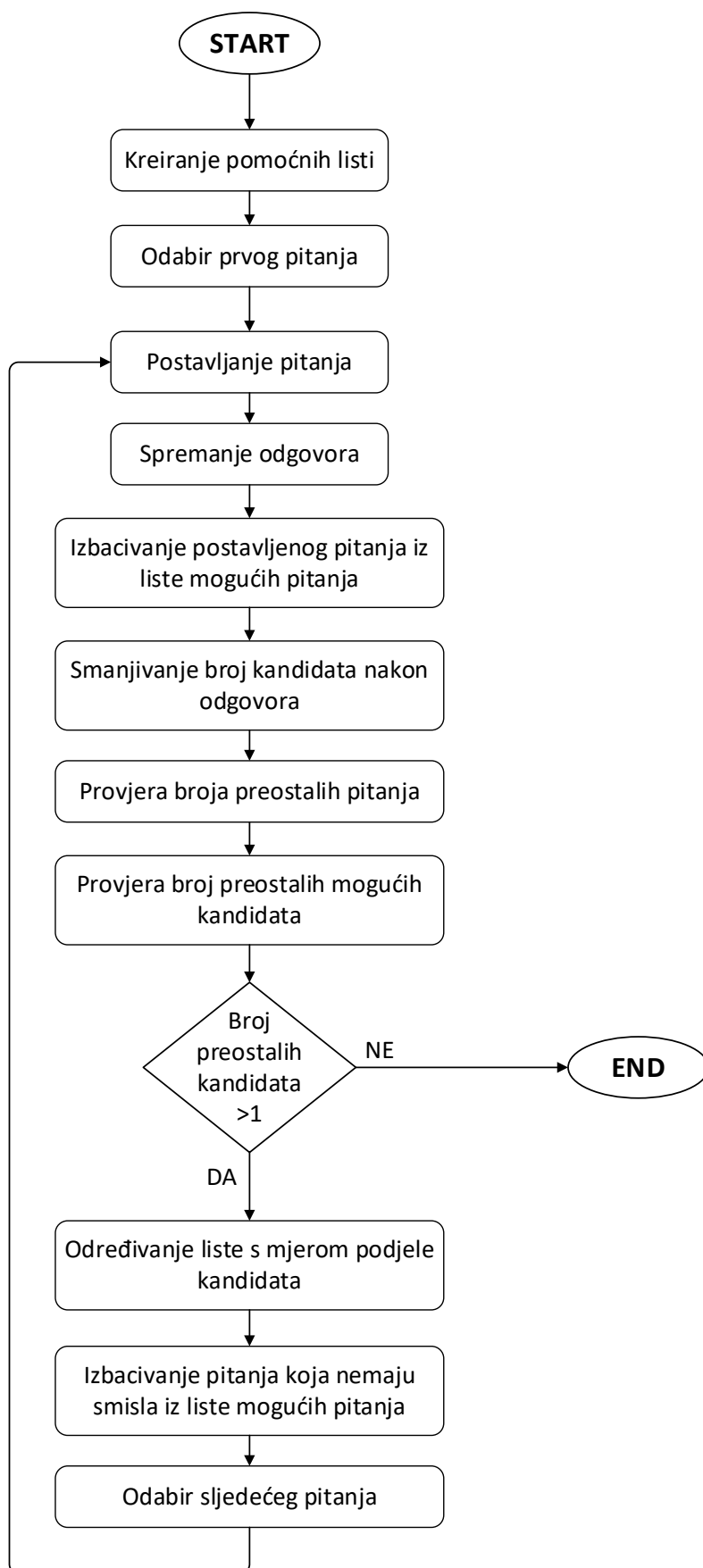
### Završetak algoritma (END)

U novi objekt *myCandidate* se sprema prvi kandidat iz liste mogućih kandidata (ako je ostao točno jedan kandidat, onda je to upravo taj jedini preostali kandidat). Kandidat se dohvaća iz *category.candidates* prema prvom članu tj. identifikatoru iz liste *myCandidates*

Pozivaju se metode:

- *storeAnswerList* koja će spremi sve odgovore u bazu podataka
- *tryToGuess* i prosljeđuje joj se kandidat iz *myCandidate* te se poziva otvaranje stranice *guess*.





Slika 34 Dijagram toka



## 3.5 Opis algoritma za davanje hintova

Algoritam za davanje hintova se nalazi u mapi *hint*, u datoteci *hint.ts*. Posebnost ovoga smjera igre je da se većina algoritma odradi kod pokretanja same stranice (izabere kandidat i definira redoslijed hintova), dok se tijekom same igre provjerava je li igrač spreman na pogođanje, odnosno jesu li potrošeni svi hintovi.

### Početak algoritma (START)

Algoritam započinje kada korisnik na stranici *whowillquess* izabere da će on pogađati pojam kojeg će zamisliti zmaj Drago. Počinje učitavanje nove stranice *hint* i njoj se prosljeđuju svi podaci vezani uz odabranu kategoriju iz objekta *categories*. Pokreće se metoda `ngOnInit`.

### Postavljanje brojača na vrijednost nula

Još prije pokretanja metode `ngOnInit`, sve potrebne pomoćne varijable se postavljaju na početne vrijednosti. Jedna od tih varijabli je brojač *cnt* koji je ključan za rad algoritma jer se koristi za prolaz kroz listu hintova i pomaže kako bi se znalo kojeg dohvatiti – hintovi se hvataju po redu kako su posloženi unutar pomoćne liste, a brojač se tijekom algoritma povećava za po jedan nakon svakog ispisa pojedinog hinta.

### Kreiranje liste *myCandidates*

Unutar metode `ngOnInit`, kreira se nova lista za moguće kandidate, *myCandidates*, koja će sadržavati identifikatore svih onih kandidata koje zmaj Drago može zamisliti.

### Slučajni odabir kandidata (ono što zmaj Drago zamisli)

Slučajnim odabirom, odabire se jedan kandidata iz liste *myCandidates*. Taj kandidat predstavlja pojam kojeg je zmaj Drago zamislio.

```
// lista mogućih kandidata
this.myCandidates=[];
for (let item of this.category.candidates) {
  this.myCandidates.push(item.cid);
}
this.n=this.myCandidates.length;

// slučajnim odabirom izaberem jednog kandidata
this.myCandidate=this.category.candidates[Math.floor(Math.random()*this.category.candidates.length)];
// za razvoj i testiranje uzмимо da je npr. cid=15 Grčka ili cid=48 Vatikan
// this.myCandidate = this.category.candidates.find(i => i.cid === "48");
console.log("Zamišljeni kandidat:"+this.myCandidate.cid +", "+this.myCandidate.cname);
```



### Kreiranje liste `myQHints`

Unutar metode `ngOnInit`, kreira se pomoćna lista `myQHints`. Lista se sastoji od identifikatora svih pitanja (hintova). Na početku se u ovoj listi nalaze svi mogući identifikatori hintova. Kako se koji hint ispisuje na ekran, iz liste `myQHints` će se izbaciti njegov identifikator kako bi se znalo koliko je mogućih hintova preostalo. Kada ova lista ostane prazna, korisniku se neće ispisati novi hint (jer su svi već ispisani) već će se ispisati poruka kako nema više hintova i da se mora pogađati.

### Kreiranje liste `LookupHints`

Unutar metode `ngOnInit`, kreira se još jedna pomoćna lista `LookupHints`. `LookupHints` je lista kod koje svaki član sadrži 3 elementa:

- `qid` – predstavlja identifikator pitanja tj. hinta
- `typeofhint` - govori je li riječ o pozitivnom ili negativnom hintu
- `ndist` – govori koliko dobro navedeni hint dijeli listu kandidata

`LookupHints` lista je ključna lista algoritma.

### Određivanje redoslijeda pojavljivanja hintova

`LookupHints` se koristi za određivanje redoslijed ispisa. Svi elementi liste se prvo nasumično rasporede. Nakon toga se lista posloži na način da se prvo pojavljuju pozitivni hintovi, a zatim negativni hintovi. Tako je postignuto da hintovi ne počinju na isti način svaki puta kada se izabere isti kandidat, a negativni hintovi omogućuju da neovisno o zamišljenom kandidatu igra može trajati jednako dugo.

```
// slučajno premještanje unutar LookupHints liste.  
this.lookupHints.sort((a,b) => 0.5 - Math.random());  
// namještanje LookupHints liste tako da prvo idu oni koji imaju typeofhint=hintyes  
this.lookupHints.sort(a => (a.typeofhint=="hintyes") ? -1 : ((a.typeofhint=="hintno") ? 1 : 0));
```

### Dohvat i ispis hinta liste `lookupHints` s pozicije brojača

Iz liste `lookupHints` se dohvaća element koji se nalazi na poziciji s vrijednošću brojača `cnt` (ako je vrijednost brojača npr.5, dohvatiti će se element koji se nalazi na 5. mjestu u listi). Za dohvaćeni `qid` se poziva metoda `goToHint` koja dohvati sve potrebne atribute iz ukupne liste pitanja/hintova (`category.questions`) te tekst samog hinta ispiše na ekran.



### **Izbacivanje ispisanog hinta iz liste myQHints**

Nakon ispisa hinta na ekran, iz liste **myQHints** se izbacuje taj element. Time je omogućeno da se zna koliko je mogućih hintova preostalo. Kada ova lista ostane prazna, korisniku se neće ispisati novi hint (jer su svi već ispisani) već će se ispisati poruka kako nema više hintova i da se mora pogađati.

### **Provjera je li korisnik odlučio pogađati**

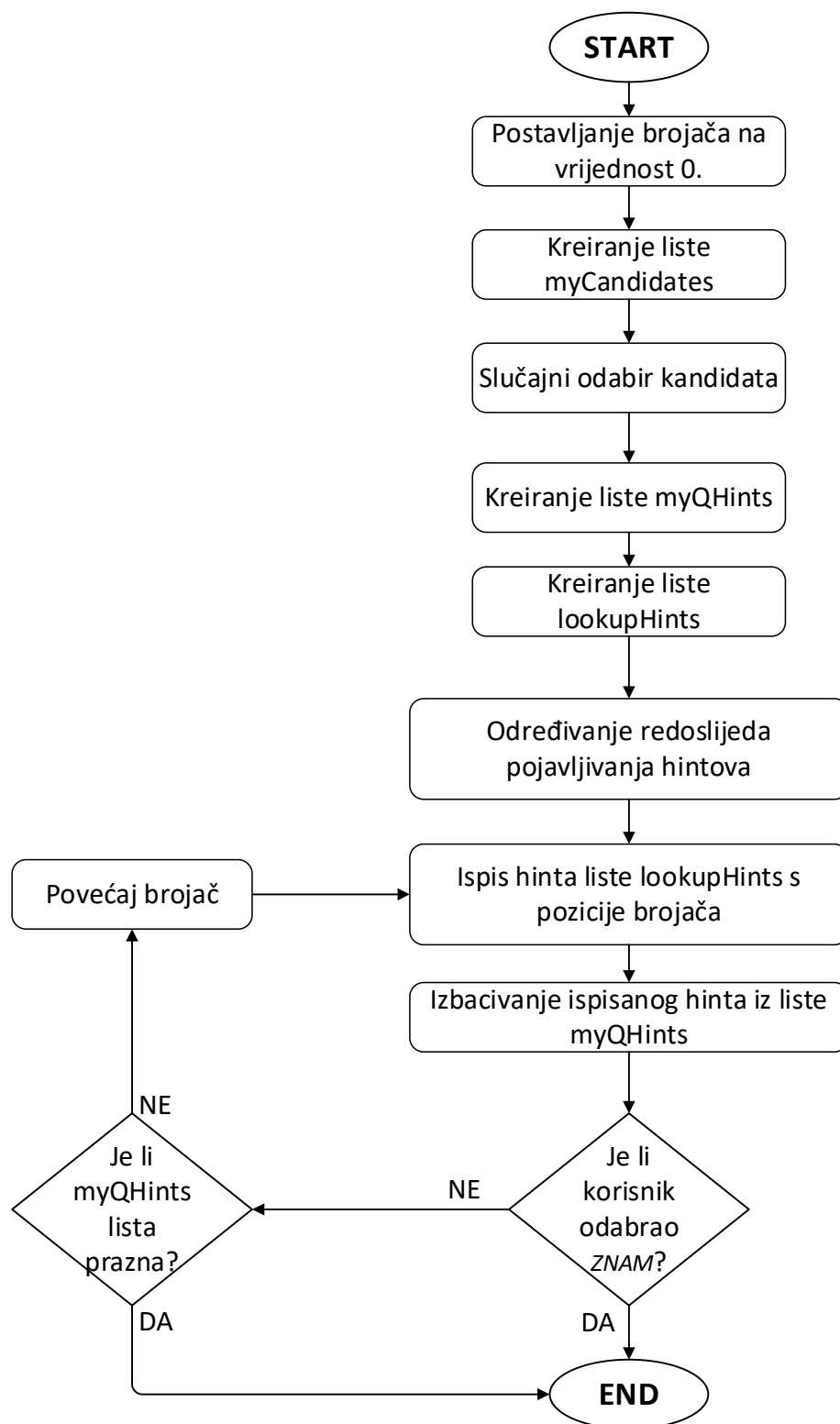
Nakon svakog ponuđenog hinta, korisnik može pokušati pogoditi (klik na gumb ZNAM) ili može zatražiti novi hint (klik na gumb DALJE). Ukoliko korisnik ne odluči pogađati nego zatraži novi hint, kao sljedeći korak algoritma se pokreće provjera je li lista **myQHints** prazna. Ukoliko je korisnik odlučio pogađati, otvara se ekran za pogađanje.

### **Provjera je li lista prazna**

Prije dohvata sljedećeg hinta, uvijek se provjerava ima li još elemenata u listi **myQHints** tj. ima li još hintova koje bi zmaj Drago mogao dati korisniku. Ako lista nije prazna (hintova ima), sljedeći korak algoritma je povećavanje brojača cnt. Ukoliko je lista prazna, korisniku se ispisuje poruka da mora pokušati pogoditi, a sljedeći korak je pogađanje tj. kraj algoritma.

### **Završetak algoritma (END)**

Igra traje dok korisnik ne pogodi zamišljeni pojam, odnosno dok ne ponestane hintova.



Slika 35 Dijagram toka



## 3.6 Analiza odgovora

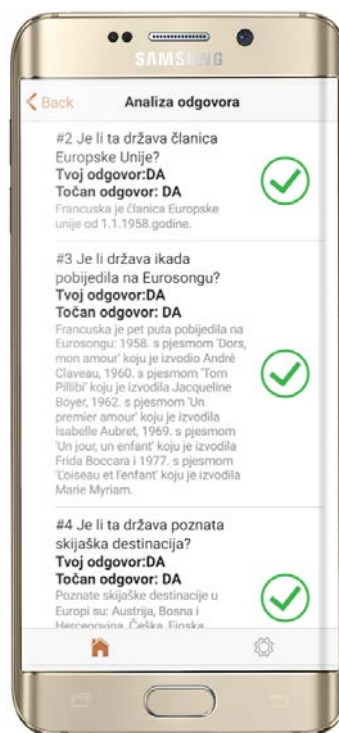
Nakon što je završila igra pogađanja, aplikacija ispisuje o kojem se pojmu radi prema odgovorima koje je dobila. Od korisnika se očekuje da potvrdi je li pogodak točan ili nije.

Ukoliko je pogodak bio točan, unutar metode `guessTrue` unutar `guess.ts` u folderu `guess`, dohvaćaju se točni odgovori na sva pitanja koja su postavljena i za pogodeni pojam. Ovo se radi kako bi se znali točni odgovori i na pitanja koja su odgovorena s „Ne znam“. Sve zajedno se spremi u bazu ispod čvora `users/ime_korisnika`.

Ukoliko je pogodak bio netočan, unutar metode `guessFalse` unutar `guess.ts` u folderu `guess`, pokrenut će se nova stranica `categories`. Na toj stanici se ispisuju svi mogući kandidati. Korisnik izabere onog koji je zamišljeni pojam (unutar `categories.html`). Kada ga izabere poziva se metoda `radioChecked` koja se nalazi u `categories.ts`. Unutar nje dohvaćaju se točni odgovori na sva pitanja koja su postavljena i za pojam kojeg je korisnik označio kao zamišljen. Sve se spremi u bazu ispod čvora `users/ime_korisnika`.

```

└─ users
  └─ EeSrhSbOVRUPAkNgu3pT0cUiyC42
    └─ PaQz1LTxgwT8poJaMsmKsLm67YT2
      └─ answer-list
        └─ 0
          └─ 1
            ├── answer: "DA"
            ├── canswer: "DA"
            ├── cqexplain: "Francuska je članica Europske unije od 1.1.1958"
            ├── name: "Europske države"
            └─ qclist
              └─ qid: "36"
                 qtext: "Je li ta država članica Europske Unij"
          └─ 2
            ├── answer: "DA"
            ├── canswer: "DA"
            ├── cqexplain: "Francuska je pet puta pobijedila na Eurosongu:"
            ├── name: "Europske države"
            └─ qclist
              └─ qid: "32"
                 qtext: "Je li država ikada pobijedila na Euroson"
          └─ 3
            ├── answer: "DA"
            ├── canswer: "DA"
            ├── cqexplain: "Poznate skijaške destinacije u Europi su: Austr"
            ├── name: "Europske države"
            └─ qclist
              └─ qid: "37"
  
```



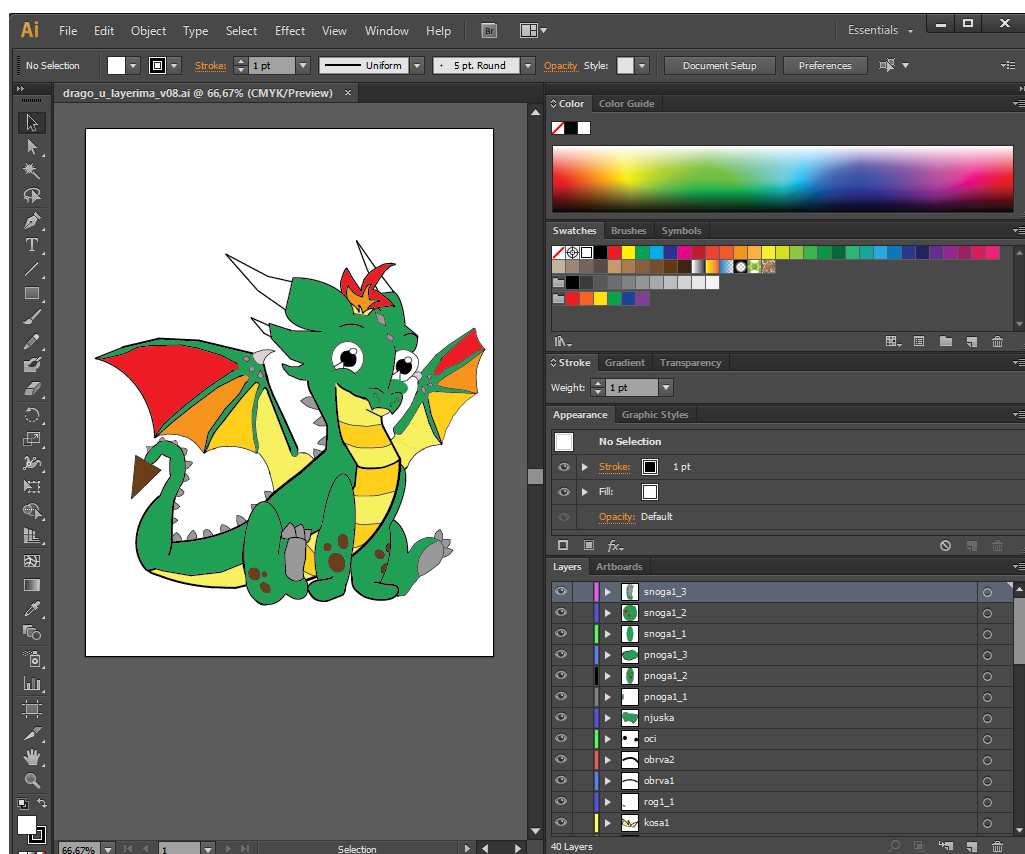
Slika 36 Analiza odgovora

Na stranici `answers`, dohvaćaju se iz baze iz dijela ispod čvora `users/ime_korisnika`, potrebni podaci za ispis na ekran.



### 3.7 Izrada animiranog lika DRAGOn

Animirani lik zmaja Drage je napravljen unutar alata Adobe Illustrator CS6. Zmaj je nacrtan na način da su njegovi pojedini dijelovi nacratni na zasebnim *layerima*, npr. na posebnim layerima su njuška, oči, lijeva obrva, desna obrva, mali pramen kose, veći pramen kose, veliki pramen kose, lijevi rog, desni rog, jezik .... Nacrtani su i neki dijelovi zmaja koji se ne vide na slici kada je uključena vidljivost svih layera (npr. jezik u ustima), ali su potrebni kako bi se mogli kasnije koristiti za animaciju. Ima oko 40 layera.



**Slika 37** Prikaz crteža zmaja Drage u Adobe Illustratoru kada su vidljivi svi layeri.



Slika 38 Prikaz crteža zmaja Drage u Adobe Illustratoru kada su vidljivi samo neki layeri.

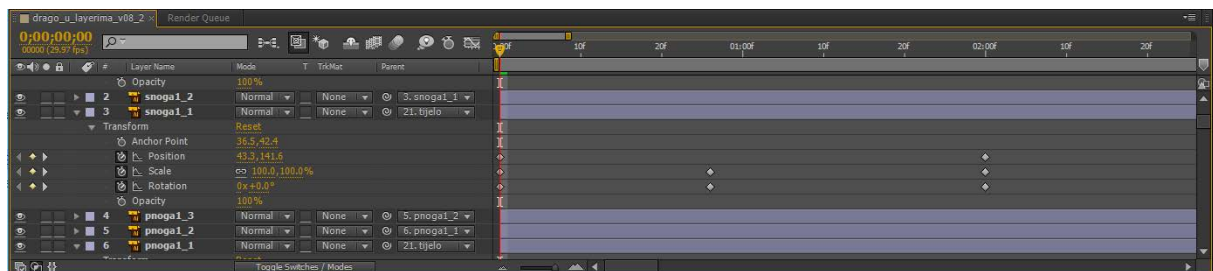
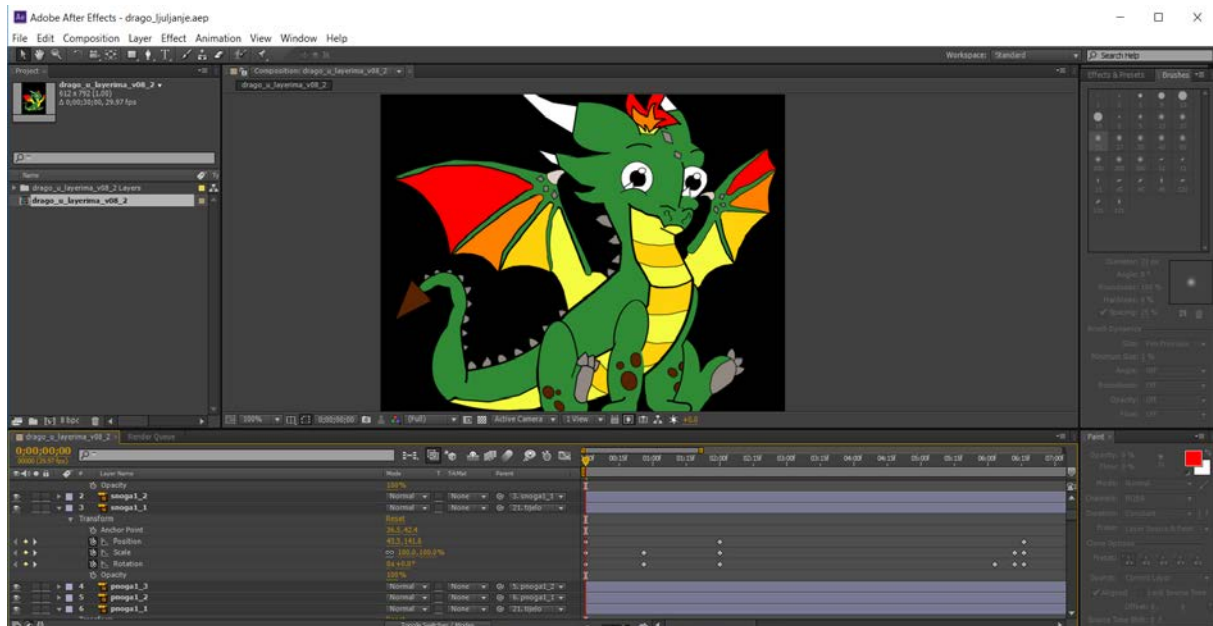


Slika 39 Popis layera





Zmaj je nacrtan na taj način kako bi se omogućila njegova animacija. Animacija lika je napravljena u alatu Adobe After Effects CS6. U njega je učitani nacrtani lik u layerima. Tamo su pojedini dijelovi (slike na zasebnim layerima) prvo spojene, a onda je za svaki layer određena točka oko koje se taj dio lika može okretati. Na dnu after effect nalazi se timeline za svaki dio slike. Na tom timeline-u su određene animacije – odabrane su dvije točke u vremenu i za svaku je definirana pozicija, after effects je dodao animaciju između temeljem definicije položaja u te dvije točke.



Svake animacija je nacrtana posebno.

Animacije su exportirane u format Quick time (.mov) sa brojem frame-ova u sekundi 29.97.



---

## 3.8 Glas zmaja Drage i prepoznavanje govora

U aplikaciju je dodana komponenta Speech Recognition.

Time je omogućeno da korisnik glasovno odgovara na pitanja. Aplikacija snimi odgovor, spoji se na google servis za prepoznavanje govora, vrijednost uspoređuje sa traženim 'da', 'ne', 'ne znam'. Ako je neka od njih prepoznata, aplikacija nastavlja rad.

Obzirom da korisnik može glasovno odgovarati na pitanje, očekuje se i da sam animirani lik zmaja može govoriti. Glas za zmaja Dragu je snimljen unaprijed pa se prilikom prikaza novog pitanja uvijek pušta odgovarajući audio zapis.

Odmah po završetku svakog pojedinog audio zapisa, uključuje se mikrofoni koji osluškuje odgovore.



## 4 Korišteni alati i tehnologija



- Ionic version 4.10.1

je framework za izradu hibridnih mobilnih aplikacija tj. koji omogućuje da se napiše samo jedan programski kod koji se onda prevede u kod za različite operativne sustave mobilnih uređaja (Android, iOS).



- Angular version 7.2.3

je framework koji se temelji na JavaScriptu i služi za izradu dinamičnih web aplikacija, a može koristiti standardni HTML i CSS.



- Angular CLI: 7.2.4

je alat kojeg kreira inicijalni Angular projekt sa svim komponentama, a služi za razvoj angular aplikacija putem terminala (command prompt).



- Node JS 11.9.0

je platforma, napravljena pomoću JavaScript-a, koja je pogodna za izradu brzih real-time aplikacija.



- Typescript 3.1.6

je nadskup od JavaScripta - može sve što može i JavaScript, ali ima i svoje dodatne funkcionalnosti.



- Webpack 4.28.4

se koristi za objedinjavanje svih modula.



- RxJS 6.3.3

dodatna komponenta koja omogućuje da se iz baze podataka dohvati cijela kolekcija umjesto samo pojedinog elementa



- Apache Cordova platform android 7.1.4
- Apache Cordova platform ios 4.5.4

- Cordova Plugin for Speech Recognition 1.2.0

- Cordova Native Audio Plugin

- Android Studio & Gradle

je alat koji se koristi za razvoj native android aplikacija, a u ovom projektu su korištene samo neke njegove funkcionalnosti potrebne za deploy aplikacije na sam android uređaj. Za istu potrebu je u Android Studio dodana i komponentu Gradle.





- **Firestore Database**  
je NoSQL baza podataka u oblaku koja sprema podatke u formatu JSON. Kao što i samo ime kaže, ovo je baza podataka u stvarnom vremenu pa se svaka promjena unutar baze podataka na poslužitelju odmah prenosi i na sam uređaj. Time je omogućen brz rad aplikacije koja koristi ovu bazu podataka.

Ovu bazu podataka se u ovom projektu koristi za spremanje popisa korisnika aplikacije (zbog autentifikacije) te za spremanje kategorija, pitanja i mogućih odgovora sa svim potrebnim elementima (za potrebe kviza).



- **Atom 1.34.0**  
alat koji je korišten za razvoj aplikacije DRAGON

Za izradu animiranog lika zmaja Drage, korišteni su:



- **Adobe Illustrator CS6**  
je alat koji se koristi za izradu vektorskih slika.



- **Adobe After Effects CS6**  
je alat koji se koristi za izradu animacija.



- **Adobe Photoshop CS6**  
je alat koji se koristi za izradu slika koje se temelje na pixelima.



## 5 Planovi za budućnost

Moji planovi za daljnji razvoj aplikacije su sljedeći:

1. Dodavanje novih kategorija.
2. Izrada kvalitetnih animacija zmaja Drage koje su prilagođene svakom pitanju i to za svako pitanje nekoliko različitih animacija te prilagodba sadašnje verzije aplikacije na način da uz pojedino pitanje slučajnim odabirom izabere jednu od tih animacija. Vjerujem da bi na taj način aplikacija postala zanimljivija.
3. Dodavanje funkcionalnosti Speech Recognition na sve stranice aplikacije.
4. Dodati mogućnost da svaki korisnik izabere za sebe avatara – sliku po njegovom izboru koja bi ga predstavljala u aplikaciji.
5. Dodati pamćenje što korisnik 'zna' pa prilagoditi algoritam da češće daje hintove koje nisu korisnika asocirali na rješenje, a rijetko one temeljem kojih odmah uspješno pogodi.
6. Omogućiti unos novih kategorija kandidata i pitanja za korisnike s posebnom rolom.
7. Omogućiti da aplikacija radi na više jezika.



## 6 Literatura

- *Ionic 2/Ionic 3 – Build iOS & Android Apps with Angular*, video tečaj kojeg drži Maximilian Schwarzmüller dostupan na <https://www.udemy.com/ionic-2-the-practical-guide-to-building-ios-android-apps/>
- *Angular 5 (formerly Angular 2) - The Complete Guide*, video tečaj kojeg drži Maximilian Schwarzmüller dostupan na <https://www.udemy.com/the-complete-guide-to-angular-2/>
- *Atlas svijeta*, sedmo izdanje, Leksikografski zavod Miroslav Krleža
- Slike položaja i zastave država Europe, <https://en.wikipedia.org/>
- *The world factbook*, Central Intelligence Agency, <https://www.cia.gov/library/publications/the-world-factbook/>
- *Creating Animated Characters in After Effects*, video tečaj kojeg drži George Maestri dostupan na <https://www.lynda.com/After-Effects-CS5-tutorials/Creating-Animated-Characters-in-After-Effects>
- *World Population Review*, <http://worldpopulationreview.com/cities-in-europe/>
- *Enciklopedija online*, Leksikografski zavod Miroslav Krleža, <http://www.enciklopedija.hr>

## 7 Zahvala

Želim zahvaliti:

- profesorici geografije Ivani Varga koja je pomogla u izradi pitanja za kategoriju Europske države
- mojem bratu Jozi Tadić koji je posudio svoj glas zmaju Dragi
- profesoru informatike i mentoru Mladenu Ćuriću koji me je uz puno strpljenja usmjeravao tijekom izrade aplikacije i pomogao da dođem do današnjeg rješenja