**Kviz o biljnim vrstama**

**Begić, Šarar, Čančarević, Pajović, Turk**

# OPIS APLIKACIJE

*Web*-aplikacija „Kviz o biljnim vrstama“ služi studentima stručnog studija Agroturizma za učenje o biljnim vrstama, formirajući kviz s različitim tipovima pitanja koji služi za provjeru njihovog znanja. Studentima se nasumično generiraju pitanja (4 vrste pitanja koje izvlači iz baze podataka) za koja se može odabrati jedan mogući odgovor, te im se odmah daje povratna informacija o točnim/netočnim odgovorima. Ukoliko je odgovor netočan, studentu se prikaže i točan odgovor. Na kraju se prikazuju i rezultati omjera točnih i netočnih odgovora.

Primjer korištenja aplikacije: studenti rješavaju kviz vezan uz biljne vrste. Unutar kviza postoje četiri vrste pitanja, od toga je jedno povezivanje fotografije s nazivom (što se nalazi na slici?) . Studentu se prikazuje fotografija koju on mora potom povezati s botaničkom porodicom s hrvatskim nazivom.

Mogući scenariji:

* Klikom na gumb "Započni kviz" korisnik započinje kviz, korisnik započinje kviz, te se istom otvara pitanje sa slikom.
* Pitanja u kvizu su nasumično generirana, te u kvizu ne postoji mogućnost da se među deset pitanja ponovi isto pitanje.
* Klikom na gumb "Prihvati odgovor" dobija se povratna informacija u obliku skočnog prozora u kojem piše informacija da li je odgovor točan, ili ako je netočan piše točan odgovor.
* Klikom na gumb "OK" na skočnom prozoru generira se novo pitanje.
* Aplikacija generira novo pitanje s podacima iz baze podataka, također to pitanje može biti random.
* Kada korisnik dođe do zadnjeg pitanja pritišće gumb „Završi i predaj“ završava s rješavanjem kviza.
* Nakon klika na gumb „Završi i predaj“ korisniku se prikazuje skočni prozor s ukupnim točnim i netočnim odgovorima.

**Terminološki sustav**

Glavni pojam na kojeg se vežu svi podatci jest biljna vrsta.

* Biljna vrsta - organizam koja uz čovjeka i životinje tvori živi svijet (hrani se, diše i giba).
* Botanička porodica – kategorija koja obuhvaća jedan ili više srodnih biljnih rodova
* Bioaktivna tvar – kemijska tvar koja može iskoristiti (sirovina)
* Rod – ime roda predstavlja skupinu biljnih vrsta (prezime biljnih vrsta)
* Uporabni dio – dio biljke koji sadrži bioaktivnu tvar

**Zahtjevi**

1. Za generiranje pitanja i postavljanje odgovora aplikacija koristi podatke koji se već nalaze u bazi podataka.
2. U aplikaciji su postavljena 4 tipa pitanja koja se nasumično pojavljuju, te se iz baze izvlače nasumični mogući odgovori, od kojih je jedan točan.
3. Broj točnih/netočnih odgovora se zbraja tijekom rješavanja kviza i prikazuje na kraju
4. S klijentske strane kreirani su potrebni prozori za prikaz pitanja i interakciju s korisnikom, te su formirani upiti prema serverskoj strani. Klijentski dio aplikacije je napravljen u Quasar.js razvojnom okviru
5. Sa serverske strane kreiran je API koji na temelju zahtjeva od strane klijenta šalje pitanja i odgovore. Serverski dio aplikacije (engl. backend) je napravljen u Node.js i Express razvojnom okviru

## Scenariji i definiranje korisničkih priča

**Autori: Šarar, Turk, Begić, Čančarević,  Pajović**

Korisnička priča:

Kao student želim provjeriti znanje o biljnim vrstama kako bi se što bolje pripremio za nadolazeći ispit i riješio ga sa što boljim rezultatom.

**Raščlanjenje priče:**

- Kao student želim da mi se prikaže početni zaslon aplikacije gdje mogu klikom na gumb započeti ispit.

  Kriterij prihvatljivosti: početna strana sa gumbima počni ispit.

- Kao student Prije nego započnem kviz, želim imati informacije o samom kvizu (npr. koliko pitanja, način bodovanja, vremensko ograničenje itd.)

  Kriteriji prihvatljivosti: informativna poruka sa navedenim podacima u zaglavlju ili podnožju aplikacije.

- Kao student želim da se generira prvi tip pitanja o raspoznavanju porodice za određenu biljnu vrstu kako bi bilo u mogućnosti provjeriti znanje.

  Kriterij prihvatljivosti: prikaz slike sa 4 ponuđena odgovora random poredani.

- Kao student želim da se prilikom prikaza prvog tipa pitanja pokaže više odgovora od kojih je jedan točan kako bi se taj točan mogao prepoznati i odabrati kao odgovor.

  Kriterij prihvatljivosti: Odgovora mora biti 4 i moraju biti random izmješani.

- Kao student želim da se nakon odabira odgovora prikaže povratna informacija točno/netočno kako bih znao jesam li odabrao ispravan odgovor.

  Kriteriji prihvatljivosti –Prikaz pitanja: Kojoj botaničkoj porodici pripada određena biljna vrsta, ponuđeno je više od jednog odgovora te su odgovori nasumično izmješani. Odabrani odgovor se provjerava i ispisuje se povratna info o tome da li je odgovor točan ili ne. Ukoliko nije, prikazati ispravan odgovor.)

- Kao student želim da se nakon svakog klikanja na odgovor o biljkama prikazuje odgovor da li je točan ili netočan.

Kriterij prihvatljivosti: U bazi podataka biti će pohranjeni točni odgovori koji će se prikazati studentu kad odgovori na njih.

- Kao student želim da se generira drugi tip pitanja kako bi se provjerilo znanje o latinskom nazivu biljne vrste.

- Kao student želim da se prilikom prikaza drugog tipa pitanja pokaže više odgovora od kojih je više odgovora točno kako bi se taj točan mogao prepoznati i odabrati kao odgovor. Naravno, odgovori moraju biti izmiješani.

- Kao student želim da se nakon odabira odgovora za drugi tip pitanja prikaže povratna informacija točno/netočno kako bih znao jesam li odabrao ispravan odgovor ili odgovore.

  Kriteriji prihvatljivosti –Prikaz pitanja: Koji je latinski naziv biljne vrste, ponuđeno je više od jednog odgovora te su odgovori nasumično izmiješani. Odabrani odgovor se provjerava i ispisuje se povratna info o tome da li je odgovor točan ili ne. Ukoliko nije, prikazati ispravan odgovor/odgovore.

- Kao student želim da mi se pokaže postotak na kraju pisanja testa.

  Kriterij prihvatljivosti: Skočni prozor s prikazom postotka uspješnosti.

- Kao student želim da se pitanja pokazuju nasumično, a ne da su svaki put istim redoslijedom.

  Kriterij prihvatljivosti: Postojala bi opcija koja bi omogućila studentu da klikom na gumb omogući pitanja koja nisu poslagana istim redoslijedom.

- Kao student želim da na svakoj slici biljke klikom na biljku dobijemo informaciju o toj biljci koja će voditi na novi tab,

  Kriterij prihvatljivosti: Svaka biljka imala bi svoj link koji vodi na informaciju o toj biljci ili Nakon odgovorenog pitanja, želim imati informaciju ukoliko je moj odgovor točan/netočan te "fun fact" o samoj biljci. Npr. ukoliko je pitanje o biljci Pachira aquatica, sa strane želim da mi se prikaže tekst o samoj biljci : Pachira Aquatica (narodni naziv: Drvo novca) je tropski član porodice Malvaceae i porijeklom iz Srednje Amerike.Pripada obitelji Bombacaceae, istoj kojoj pripada i baobab. Oprašuju je šišmiši, sfingidni moljci i pčele medarice.

- Kao student želim da aplikacija bude responzivna (mobile friendly) što znači da na određenom widthu se smanjuje slika (flexbox layout) npr. Kreira se hamburger menu na mobilnim uređajima.

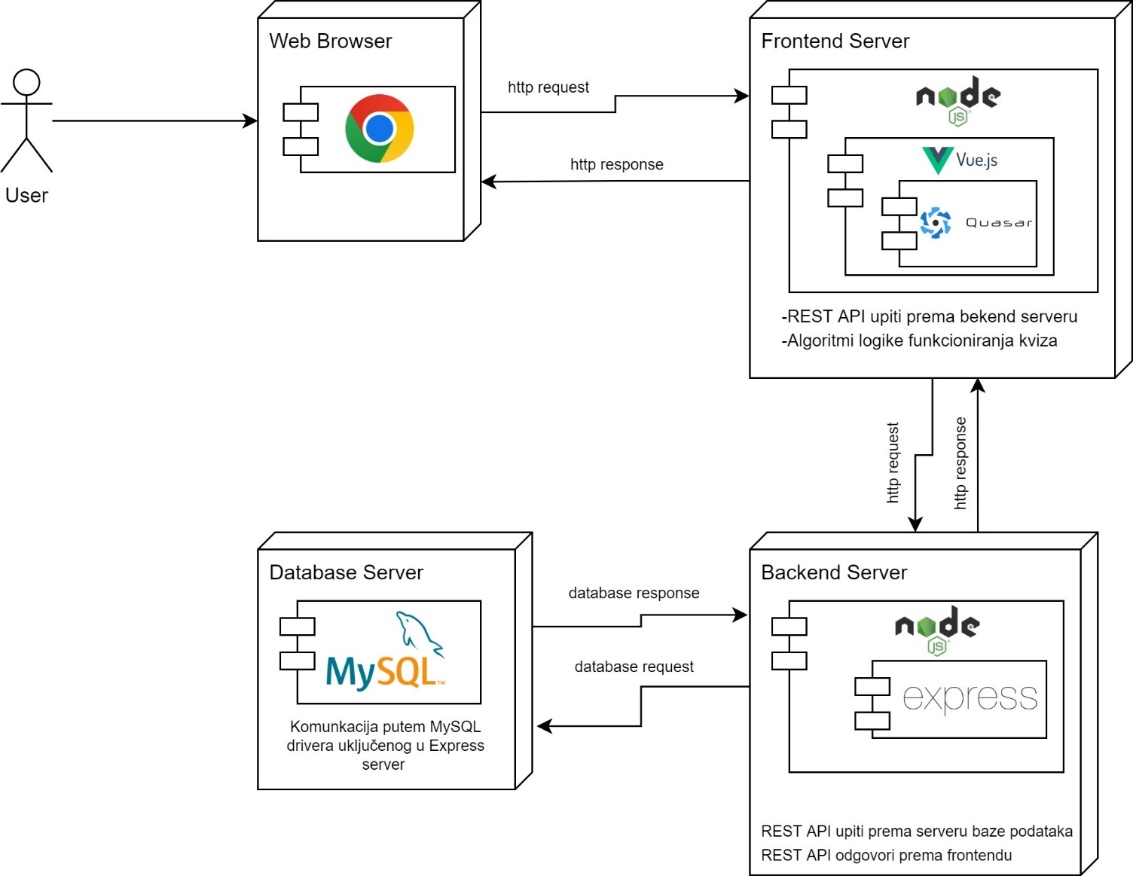
  Kriterij prihvatljivosti: ekran se prilagođava uređaju i prikaz je sukladan prikazu na tom uređaju. Naodređenim zaslonima prikazivao bi se „malo prilagođeniji oblik“ aplikacije.

- Kao student želim opciju „Pomoći“ unutar aplikacije ukoliko nisam siguran koji je odgovor točan.

  Kriterij prihvatljivosti: Kreirati opciju pomoći koji će koristiti bodove za prikaz točnog odgovora. Npr. Prikaz točnog odgovora iznosi 1 bod a prikupljanje bodova za ovu opciju izvršavala bi se na pricipu točnosti odgovaranja.

# ARHITEKTURA SUSTAVA

Slika 1 UML Dijagram isporuke



Arhitektura web aplikacije podijeljena je na tri nezavisne cjeline; frontend, bekend i baza podataka.

Frontend:

* Quasar okvir utemeljen na Vue.js okviru za agilnu izradu web aplikacija
  + Definira izgled stranica paletom besplatnih komponenti poput gumbova, raznih pomičnika, dinamičkih listi, radio gumbova...
  + Renderiranje sadržaja na ekran
  + Putem axios modula šalje REST api upite prema bekend dijelu aplikacije
  + Poziva metode sa algoritmima za logiku izvršavanja kviza
* Express.js utemeljen na Node.js okviru za izradu bekend funkcionalnosti web aplikacija
  + Jednostavan asinkroni web server utemeljen na http tehnologiji
  + Zadužen za uspostavu komunikacije i izvršavanje CRUDE komandi prema MySql bazi podataka
  + Zadužen za komunikaciju sa frontend dijelom aplikacije putem brzih asinkronih REST API metoda
* MySql baza podataka
  + Skladište podataka modelirano prema EV modelu podataka (Slika 2)
  + Poslužuje bekend server sa odgovrima na CRUD zahtjeve
  + Komunikacija sa express serverom ustanovljena je pomoću MySQL drivera instaliranog na poslužitelju

Moduli ugrađeni na frontend strani sustava:

"@quasar/extras": "^1.0.0",

"axios": "^0.27.2",

"core-js": "^3.6.5",

"quasar": "^2.6.0",

"vue": "^3.0.0",

"vue-axios": "^3.4.1",

"vue-router": "^4.0.0"

Moduli ugrađeni na bekend strani sustava:

* Express
* Body-parser
* Mysql

Tehnologije koje se koriste prilikom izrade kviza o biljnim vrstama su:

**BACKEND**

**Node.js**

Programiranje na strani poslužitelja. Omogućuje pisanje poslužiteljskog koda u Javascriptu. Kod Node.js-a posebno je važno napomenuti npm. Npm je kratica za node package manager, upravljač paketima za Node.js. Osnovne postavke, instalacije modula i Express razvojnog okruženja dobivaju se preko npm Node.js upravitelja paketima (Node.js Package Manager). Aplikacija npm preuzima i instalira zahtijevane pakete te automatski instalira i o njima ovisne pakete. Paketi se instaliraju lokalno, unutar foldera odakle se radi instalacija. Početna inicijalizacija projekta u trenutačnom folderu radi se naredbom **npm init**

Uloga npm modula je da se olakša upravljanje eksternim JavaScript bibliotekama koje programeru mogu zatrebati,u smislu da se ne mora razmišljati na tradicionalni način korištenja JavaScripta, a to je da se pazi gdje se i koje biblioteke uključuju, u koje HTML datoteke, na koje točno mjesto i kojim redoslijedom .

**Express**

Express je najvažniji modul backenda, okvir za razvoj web aplikacija na poslužitelju

Express sadrži osnovne funkcionalnosti:

* upravljanje rutama (routing), npr. /home, /registration,...
* kreiranje srednjeg sloja aplikacije (middleware) - middleware su funkcije koje imaju potpun pristup zahtjevima i odgovorima, te može raditi kontrolu i upravljanje poslovnom logikom.
* pogledi (view) - korištenje predložaka (template engine) i statičkih datoteka (static file serving) - template engine omogućuje renderiranje HTML datoteke s ubačenim varijablama sa serverske strane.

Express omogućuje izradu CRUD operacije koje će se izvršavati koristeći drivere za različite baze podataka.

**Neke od prednosti Express-a:**

* Lako postavljanje i korištenje
* Jednostavnije rukovanje zahtjevima uvođenjem tvz. middleware-a, „funkcija“ koje se pokreću tijekom zahtjeva
* Daje mogućnost lakog serviranja statičnih dokumenata, poput slika
* Lako se integrira sa poznatim bazama podataka kao što su MongoDb ili MySQL
* Instalacija Express razvojnog okruženja kroz upravitelj paketa npm:

**npm install express**

**Body Parser**

Modul body-parser ima funkcije za parsiranje tijela (body) dolaznih HTTP zahtjeva na srednjem sloju (engl.middleware) prije nego odu u daljnju obradu. Modul podržava parsiranje formata: JSON, raw, text, URLencoded form.

**MySQL**

Open source sustav za upravljanje relacijskim bazama. Kviz Biljne vrste koristi MySQL kao bazu za spremanje podataka.

**Web servisi**

Web servisi se općenito mogu definirati kao kolekcija standardiziranih protokola i aplikacijskih

programskih sučelja (engl. API, Application Programming Interface) koji dopušta pojedincima i

aplikacijama da ih lociraju i koriste. U konkretnim okvirima definiraju se kao dio aplikacije koji

izvodi specifičan zadatak ili skup zadataka koji se izvršavaju na zahtjev te kojem se pristupa preko

web sučelja.

Web servisi koriste jezično i platformski neovisne tehnologije: standardizirani web protokol (HTTP ili HTTPS) za komunikaciju i razmjenu podatka i poruka, te XML (engl. EXtensible Markup Language) ili JSON (engl. JavaScript Object Notation) tekstualne formate za te podatke.

Karakteristike web servisa su:

* neovisnost o platformi i programskom jeziku - web servis može biti napisan u bilo kojem
* programskom jeziku i izvršavati se na bilo kojoj platformi,
* dostupnost preko interneta ili intraneta,
* korištenje XML/JSON standarda za komunikaciju i opis servisa,
* jednostavno otkrivanje lokacije.

**FRONTEND**

**Quasar**

Quasar je razvojni okvir za razvoj prednjeg dijela (engl. frontend) web aplikacija, baziran na razvojnom okviru Vue.js. U pozadini se koriste osnovne web tehnologije: HTML, CSS te programski jezik JavaScript. Također koriste i razvojne okvire Bootstrap i JQuery, koji se isto temelje na CSS-u i JavaScriptu..

Quasar aplikacija građena je na „\*.vue“ datotekama. Te datoteke u Quasaru koriste Single

File Components (SFC) format iz Vue.js razvojnog okvira za definiranje njihove strukture.

Datoteke se sastoje od tri dijela:

* template
* script
* style

**Postupak instalacije Quasar frameworka**

1.Instalacija se vrši unutar frontend foldera naredbom „cd frontend“

2. Naredbom npm i -g @quasar/cli pokrećemo instalaciju Quasar frameworka

3. Inicijalizacija (početno postavljanje) Quasar frameworka

npm init quasar

Prilikom inicijalizacije ostaviti sve predloženo ,osim u jednom koraku kad se odabire ESLint, Axios i Vue-i18n.

4.Za pokretanje Quasar frameworka upisujemo naredbe

cd quasar-project/

quasar dev

Nakon pokretanja otvara se web preglednik <http://localhost:8080/#/> Sve izmjene koje se rade u programskom kodu vidljive su odmah nakon spremanja datoteka.

# Model podataka za Kviz

Diagram

Description automatically generatedModel koji je korišten za trenutna pitanja u aplikaciji je tablica Biljna vrsta koja je preko tablice Rod povezana sa tablicom Botanička porodica.

Model koji se može korisiti za buduće tipove pitanja u aplikaciji je tablica Biljna vrsta koja je preko tablice Biljni dio povezana sa tablicom Uporabni dio.

Diagram

Description automatically generated

**Baza podataka**

HOST: "student.veleri.hr",

USER: "iooa",

PASSWORD: "11",

DB: "iooa\_biljnedev22"

**IMPLEMENTACIJA SUSTAVA**

**BACKEND API**

API je programsko sučelje koje pomaže programerima u interakciji s pozadinskim uslugama, na primjer poslužitelj. Često se API smatra pozadinskom "komponentom"

API može koristiti bilo koje načine komunikacije za pokretanje interakcije između aplikacija.

**Web servis (API) za dohvat svih biljnih vrsta**

API za dohvat svih biljnih vrsta ne uzima niti jedan parametar već iz baze dohvaća sve dostupne biljne vrste iz tablice *plant\_species*. Iz *plant\_species* tablice se uzimaju svi (\*) dostupni podaci te se nakon toga prosljeđuju na klijentski dio aplikacije. Ovaj API služi samo za dohvat i čitanje podataka pa se poziva i koristi GET HTTP metodom.

SQL upit:

*SELECT \* FROM plant\_species*

API endpoint:

*/plant\_species*

**Web servis (API) za dohvat svih biljnih porodica**

Slično kao i API za dohvat svih biljnih vrsta, niti ovaj API ne zahtjeva posebne parametre za izvršavanje već samo poziv GET metodom, nakon čega se iz baze podataka dohvaćaju sve dostupne botaničke porodice. Ovime se iz tablice *botanical\_family* dohvaćaju svi podaci te šalju prema frontend-u.

SQL upit:

*SELECT \* FROM botanical\_family*

API endpoint:

*/botanical\_family*

**Web servis (API) za dohvat jedne biljne vrste**

Ovaj API, za razliku od prethodnih, zahtjeva parametar, a taj parametar je *id* biljne vrste (*plant\_species*) koja se želi dohvatiti. Na temelju ovog parametra se iz baze podataka dohvaćaju (GET metoda) svi podaci o biljnoj vrsti sa tim *id*-jem. Tablica nad kojom se izvršava ovaj upit je *plant\_species*. Nakon dohvata podataka iz baze se isti prosljeđuju na klijentsku stranu aplikacije za izradu kviza.

SQL upit:

*SELECT \* FROM plant\_species where id=?*

API endpoint:

*/plant\_species/:id*

**Web servis (API) za dohvat biljne porodice za jednu biljnu vrstu**

Ovaj API služi za dohvat botaničke porodice za biljnu vrstu. Botanička porodica se dohvaća iz baze podataka na temelju id-ja biljne vrste koji se prosljeđuje sa strane klijenta i na serverskoj strani se čita preko req.params.id. Ukoliko se sa klijentske strane pošalje prazan id parametar ili se pošalje id biljne vrste koji ne postoji u bazi podataka, klijentu se šalje poruka: „*Please provide plant\_species\_id*“. Ukoliko je id parametar ispravan i postoji u bazi, na temelju dohvaćenog id-ja se radi SQL upit (query) na bazi podataka kojim se dohvaćaju podaci kao što su naziv botaničke porodice na hrvatskom i latinskom jeziku. Nakon dohvaćanja ovih podataka se daje odgovor klijentu (res.send), odnosno, ovi podaci se samo prosljeđuju na frontend dio kako bi se koristili za sastavljanje pitanja kviza i prikaz odgovora.

API endpoint:

*/botanical\_family\_plant\_species/:id*

SQL upit za ovaj API:

*SELECT botanical\_family.id, botanical\_family.croatian\_name, botanical\_family.latin\_name FROM botanical\_family LEFT JOIN genus ON botanical\_family.id=genus.botanical\_family\_id LEFT JOIN plant\_species ON genus.id=plant\_species.genus\_id WHERE plant\_species.id=?*

* // Dohvat biljne porodice za jednu biljnu vrstu #21
* app.get("/botanical\_family\_plant\_species/:id", function (request, response) {
* let plant\_species\_id = request.params.id;
* if (!plant\_species\_id) {
* return response
* .status(400)
* .send({ error: true, message: "Please provide plant\_species\_id" });
* }
* dbConn.query(
* "SELECT botanical\_family.id, botanical\_family.croatian\_name, botanical\_family.latin\_name FROM botanical\_family LEFT JOIN genus ON botanical\_family.id=genus.botanical\_family\_id LEFT JOIN plant\_species ON genus.id=plant\_species.genus\_id WHERE plant\_species.id=?",
* plant\_species\_id,
* function (error, results, fields) {
* if (error) throw error;
* return response.send({
* error: false,
* data: results[0],
* message: "botanical\_family\_plant\_species",
* });
* }
* );
* });

**Web servis (API) za dohvat jedne biljne porodice**

Prilikom pozivanja ovog API-ja sa klijentske strane, potrebno je da klijent pošalje valjan id botaničke porodice. Nakon toga, API radi SQL upit kojim se dohvaćaju podaci o botaničkoj porodici sa priloženim id-jem. Iz baze podataka se ovom prilikom dohvaćaju zapravo svi dostupni podaci o toj botaničkoj porodici. Ovi podaci se dalje prosljeđuju na frontend te se koriste kako bi se sastavila pitanja i odgovori za kviz.

API endpoint:

*/botanical\_family/:id*

SQL upit za ovaj API:

*SELECT \* FROM botanical\_family where id=?*

* // Retrieve botanical\_family with id
* app.get("/botanical\_family/:id", function (request, response) {
* let botanical\_family\_id = request.params.id;
* if (!botanical\_family\_id) {
* return response
* .status(400)
* .send({ error: true, message: "Please provide botanical\_family\_id" });
* }
* dbConn.query(
* "SELECT \* FROM botanical\_family where id=?",
* botanical\_family\_id,
* function (error, results, fields) {
* if (error) throw error;
* return response.send({
* error: false,
* data: results[0],
* message: "botanical\_family list.",
* });
* }
* );
* });

**Web servis (API) za dohvat svih biljnih vrsta za jednu biljnu porodicu**

Ovaj API također koristi id botaničke porodice koji se šalje sa klijentske strane te se na temelju njega pretražuje baza podataka i kao rezultat daje sve biljne vrste koje pripadaju botaničkoj porodici sa id-jem koji je poslan. Glavni podaci koji se uzimaju su nazivi na hrvatskom i latinskom jezku.

API endpoint:

*/plant\_species\_by\_bf/:id*

SQL upit za ovaj API:

*SELECT ps.id, ps.croatian\_name, ps.latin\_name FROM plant\_species ps left OUTER join genus g ON ps.genus\_id=g.id left OUTER join botanical\_family bf on g.botanical\_family\_id=bf.id where bf.id=?*

//dohvat svih biljnih vrsta za jednu botanicku porodicu #23

* app.get('/plant\_species\_by\_bf/:id', function (request, response) {
* let botanical\_family\_id = request.params.id;
* if (!botanical\_family\_id) {
* return response.status(400).send({ error: true, message: 'Please provide botanical\_family\_id' });
* }
* dbConn.query('SELECT ps.id, ps.croatian\_name, ps.latin\_name FROM plant\_species ps left OUTER join genus g ON ps.genus\_id=g.id left OUTER join botanical\_family bf on g.botanical\_family\_id=bf.id where bf.id=?', botanical\_family\_id, function
* (error, results, fields) {
* if (error) throw error;
* return response.send({
* error: false, data: results, message:
* 'plant\_species list.'
* });
* });
* });

**Dohvat svih uporabnih dijelova za određenu biljnu vrstu ili uporabnog dijela prema id uporabnog dijela, pod pretpostavkom da je id pitanja za prvi slučaj 5, a za drugi 6. (Marina i Hrvoje)**

app.get("/useful\_part/:id/:questionid", function (request, response) {

  let plant\_species\_id = request.params.id;

  let question\_id = request.params.questionid;

   ;

  if (!plant\_species\_id) {

    return response

      .status(400)

      .send({ error: true, message: "Please provide plant\_species\_id" });

  }

  if(question\_id==5){

 dbConn.query(

    "SELECT ps.id, ps.croatian\_name, up.croatian\_name, up.latin\_name FROM useful\_part up LEFT OUTER JOIN plant\_part pp ON up.id=pp.useful\_part\_id LEFT OUTER JOIN plant\_species ps ON pp.plant\_species\_id=ps.id WHERE ps.id=?",

    plant\_species\_id,

    function (error, results, fields) {

      if (error) throw error;

      return response.send({

        error: false,

        data: results,

        message: "plant\_species\_usful\_parts.",

      });

    }

  );

  }else if (question\_id==6){

     dbConn.query(

    "SELECT \* FROM useful\_part where id=?",

    plant\_species\_id,

    function (error, results, fields) {

      if (error) throw error;

      return response.send({

        error: false,

        data: results[0],

        message: "useful\_part list.",

      });

    }

  );

  }

});

**Dohvat fotografije za određenu biljnu vrstu (Emina i Ivan)**

app.get("/image/:id", function (request, response) {

  let plant\_id = request.params.id;

  if (!plant\_id) {

    return response

      .status(400)

      .send({ error: true, message: "Please provide plant\_id" });

  }

  dbConn.query(

    "SELECT  i.image\_url FROM image i LEFT JOIN plant\_species\_image psi ON i.id=psi.image\_id LEFT JOIN plant\_species ps ON psi.plant\_species\_id=ps.id WHERE ps.id=? LIMIT 1",

    plant\_id,

    function (error, results, fields) {

      if (error) throw error;

      return response.send({

        error: false,

        data: results[0],

        message: "plant\_species\_image.",

      });

    }

  );

});

app.get("/image/:id", function (request, response) {

  let plant\_id = request.params.id;

  if (!plant\_id) {

    return response

      .status(400)

      .send({ error: true, message: "Please provide plant\_id" });

  }

  dbConn.query(

    "SELECT  i.image\_url FROM image i LEFT JOIN plant\_species\_image psi ON i.id=psi.image\_id LEFT JOIN plant\_species ps ON psi.plant\_species\_id=ps.id WHERE ps.id=? LIMIT 1",

    plant\_id,

    function (error, results, fields) {

      if (error) throw error;

      return response.send({

        error: false,

        data: results[0],

        message: "plant\_species\_image.",

      });

    }

  );

});

app.get("/plant\_species\_by\_bf/:id", function (request, response) -> **promijenjen SQL upit za dohvat biljnih vrsta u ID (Marina i Hrvoje)**

**FRONTEND**

***Korisnička priča ili funkcionalnosti + opis***

Web aplikacija „Kviz o biljnim vrstama“ je namijenjena studentima stručnog studija Agroturizma za provjeru znanja i za učenje o biljnim vrstama kroz kviz s različitim tipovima pitanja. Studentima se nasumično generira 10 pitanja za koja se može odabrati jedan točan odgovor, a pitanja se prikazuju redom jedan po jedan te je za prijelaz na iduće pitanje potrebno potvrditi odgovor na gumb *„PRIHVATI ODGOVOR“*, tada se pojavljuje pop-up prozor s informacijom o točnosti odgovora kada je odgovor točan, odnosno pop-up prozor će obavijestiti korisnika kada je odgovor i netočan. Pop-up prozor se zatvara pritiskom na *„OK“* te se automatski generira iduće pitanje i na taj način korisnik prolazi kroz svih 10 pitanja. Nakon prikaza desetog pitanja, ujedno i zadnjeg pitanja, odgovor se potvrđuje na gumb *„PRIHVATI I ZAVRŠI“* te se u pop-up prozoru ispisuje konačni rezultat s brojem točnih i netočnih odgovora. Odabirom *„OK“* je završen kviz te je moguće ponovno ga pokrenuti odabirom gumba *„PONOVNO POKRENI KVIZ“*. Tijekom cijelog kviza se prikazuje jedna slika između pitanja i ponuđenih odgovora.

*Napomena:*

*Gore navedena korisnička priča predstavlja idealan opis web aplikacije, no postoje i greške u web aplikaciji. Greška je ta da se zapravo generira 11 pitanja na način: odabire se odgovor na 10. pitanju s gumbom „PRIHVATI ODGOVOR“, pojavljuje se pop-up prozor s informacijom o točnosti tog odgovora, te nakon odabira „OK“ broj pitanja ostaje na 10. pitanju ali se mijenja sadržaj pitanja te nije moguće završiti kviz prije nego li se opet odabere gumb „PRIHVATI ODGOVOR“ nakon koje se ne pojavljuje**pop-up prozor (s informacijom o točnosti odgovora) već gumb „PRIHVATI I ZAVRŠI“ postaje aktivan te je tada moguće završiti kviz.*

***ISPRAVAK***

Navedene greške u napomeni su ispravljene, te se sad ispisuje do desetog pitanja, a nakon toga klikom na gumb „ZAVRŠI I PREDAJ“ uračuna i odgovor na zadnje pitanje i zbroji ga u točne ili netočne odgovore. Problem je riješen na način da smo dodali i na gumb „ZAVRŠI I PREDAJ“ opciju zbrajanja:

<q-btn

id="PrihvatiIZavrsi"

color="white"

text-color="black"

label="Završi i predaj"

@click="

state.odabraniOdgovor === state.tocanOdgovor.id

? (state.brojTocnih = state.brojTocnih + 1)

: (state.brojNetocnih = state.brojNetocnih + 1);

state.zavrsniPopup = true;

"

disabled

/>

Funkcionalnosti:

* Nasumično generiranje pitanja
* Prikaz ponuđenih odgovora i odabir jednog točnog odgovora
* Prikaz slike
* Gumb *„PRIHVATI ODGOVOR“* za prihvaćanje odabranog odgovora
* Pop-up prozor s povratnom informacijom o točnom ili netočnom odgovoru
* Gumb *„ZAVRŠI I PREDAH“* za prihvaćanje zadnjeg odgovora i kraj kviza
* Pop-up prozor s ispisom konačnog rezultata (broj točnih i netočnih odgovora)
* Gumb *„PONOVNO POKRENI KVIZ“* za ponovno pokretanje kviza

**Prvo je dodana image u state. (Marina)**

const state = reactive({

      tip\_pitanja: 1,

      plant: {},

      pitanje: "",

      odgovori: [],

      odabraniOdgovor: {},

      tocanOdgovor: {},

      brojTocnih: 0,

      brojNetocnih: 0,

      image: "",

      alert: false,

      zavrsniPopup: false,

    });

**Zatim je dodana funkcija za dohvat slike (Marina i Emina i Hrvoje)**

    async function getImage() {

      const json = await axios.get(

        `<http://localhost:3000/image/>${state.plant.id}`

      );

      const data =  json.data.data;

       if(data){

      if(Object.getOwnPropertyNames(json.data).length===0|| json.data.data===undefined){

       state.image="";

      }else{

           const image=data;

       state.image=image.image\_url;

       }}

       else{

        state.image="";

       }

    }

**U funkciji se poziva slika i sprema se u state image. State image se poziva u kod. (Marina)**

      <q-img width="700px" height="350px" :src="state.image" :ratio="16 / 9" />

**Napralvjen je state u kojem se nalazi lista pitanja u kojemu se nalazi 4 tipa pitanja i napravljena je funkcija da se ispisuju random pitanja (Emina)**

            state.pitanje = [

        "Koji je latinski naziv za " + state.plant.croatian\_name,

        "Koji je hrvatski naziv za " + state.plant.latin\_name,

        "Kojoj botaničkoj porodici pripada " + state.plant.croatian\_name,

        "Koja biljna vrsta se nalazi na slici ",

      ];

      const randomQuestionIndex = Math.floor(

        Math.random() \* state.pitanje.length

      );

      state.tip\_pitanja = randomQuestionIndex;

      state.pitanje = state.pitanje[randomQuestionIndex];

    }

**Prepravljen je upit za poziv odgovora odnosno dodana je metoda za raspoznavanje vrste pitanja i sukladno tome ispisuje odgovore u bazi. (Emina i Hrvoje)**

<q-radio

        v-for="odgovor in state.odgovori"

        v-bind:key="odgovor.id"

        v-model="state.odabraniOdgovor"

        :val="odgovor.id"

        :label="getLabelOdgovor(odgovor)"

        color="teal"

      />

    </div>

getLabelOdgovor(odgovor) {

      const pitanje = this.state.pitanje

      if (pitanje.includes("latinski naziv")) {

           return odgovor.latin\_name;

         } else if (pitanje.includes("hrvatski naziv")) {

          return odgovor.croatian\_name;

         } else if (pitanje.includes("botaničkoj")) {

          return odgovor.latin\_name;

         } else if (pitanje.includes("nalazi na slici")) {

          return odgovor.croatian\_name;

         }

    },

**Ograničenja da se ne ispisuju dva ista odgovora (Hrvoje)**

///// gledamo ako je odogovor već u listi, ako je je true-> break

        var found=false;

for(var i=0;i<botanicList.length;i++){

if(botanicList[i].id==botanicObject?.id){

found=true;

break;

}

}

///// ukoliko nije odgovor u listi, dodaj ga

if(!found){

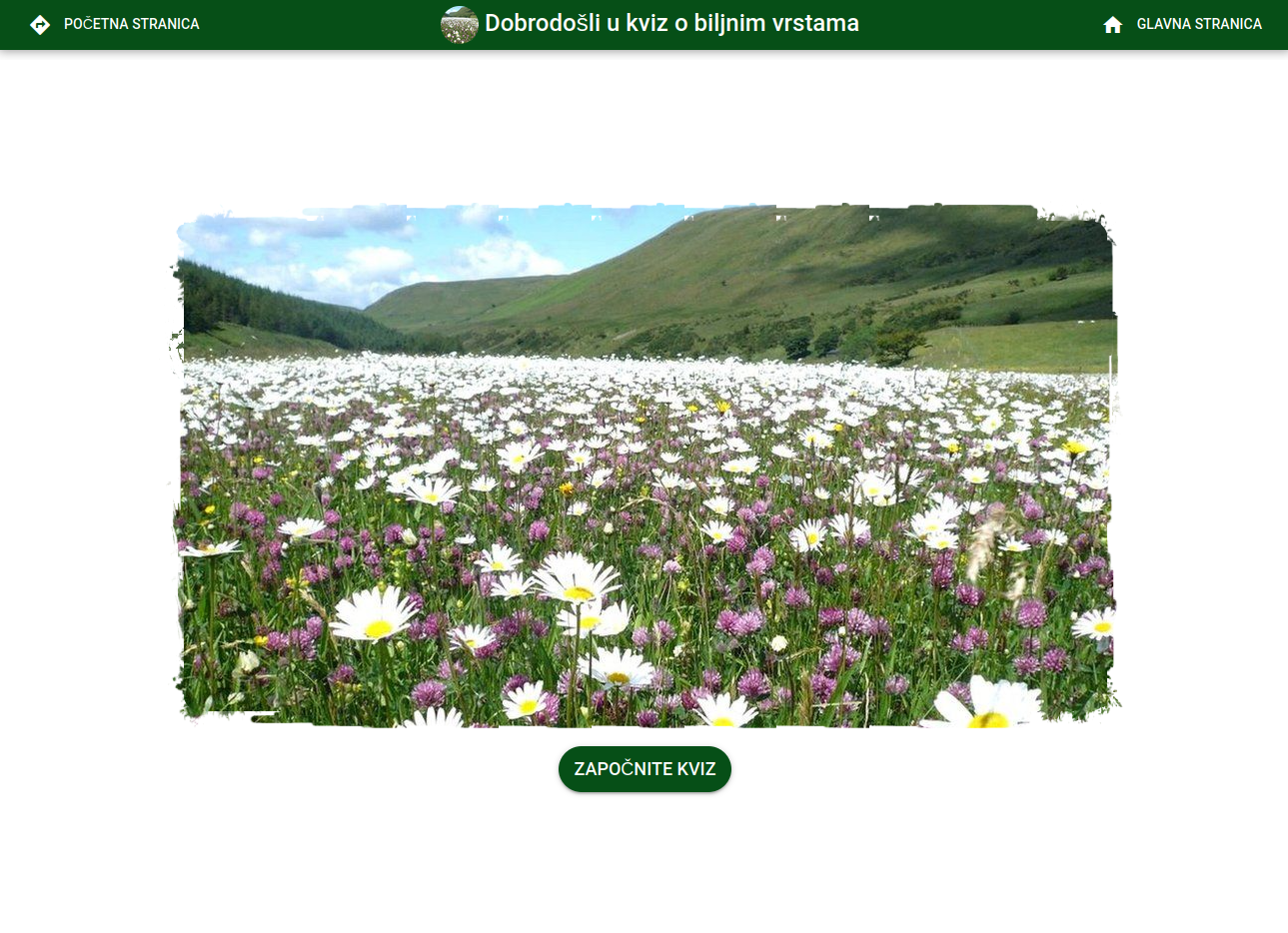
botanicList.push(botanicObject);

}

***Grafičko rješenje (screenshots) za kviz***

Na slici 2 je prikaz početnog zaslona kviza gdje je postavljeno gumb „ZAPOČNITE KVIZ“ Nakon odabira gumba započinjemo s rješavanjem kviza.

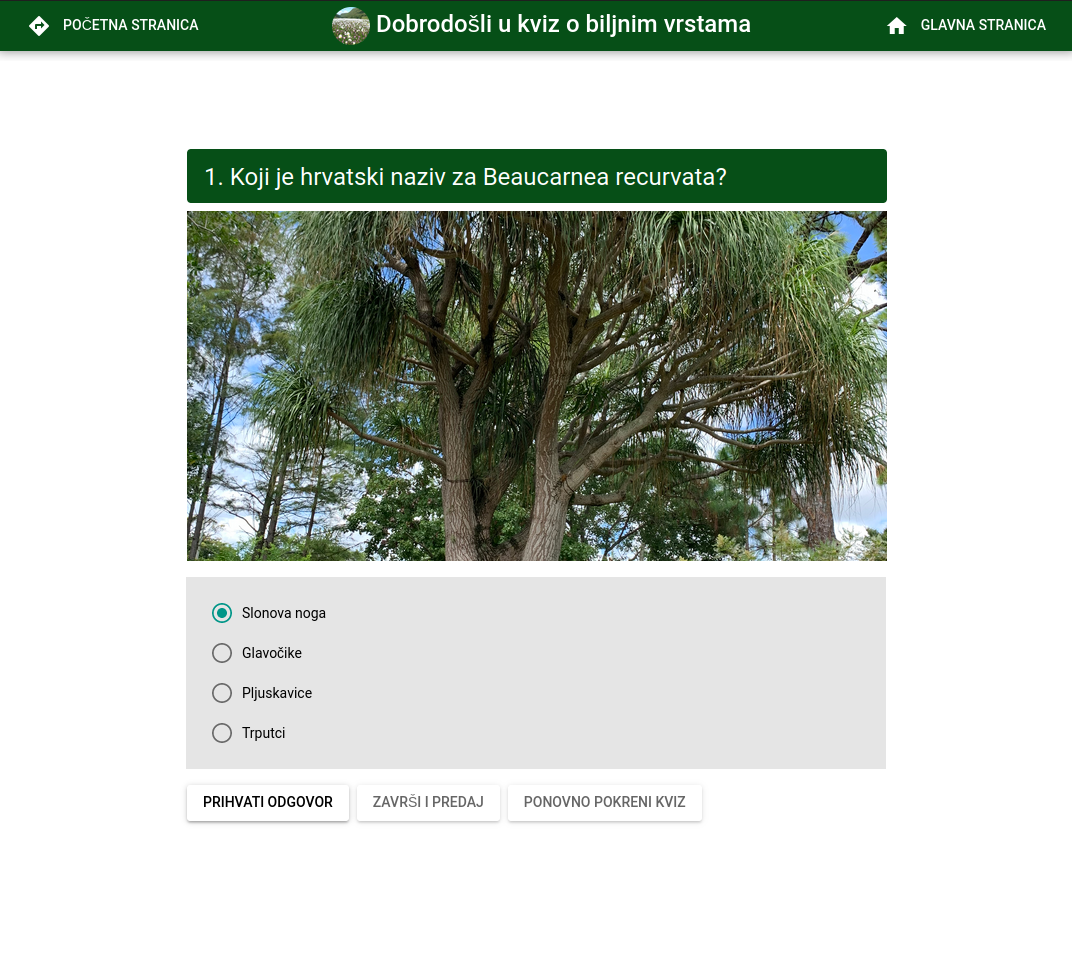
Slika 2 Početna stranica



Izvor: obrada autora

Na slici 3 je prikaz započetog kviza gdje je postavljeno prvo pitanje, ponuđena su četiri odgovora od kojih je jedan odgovor točan. Nakon odabira odgovora, treba ga potvrditi gumbom *„PRIHVATI ODGOVOR“*.

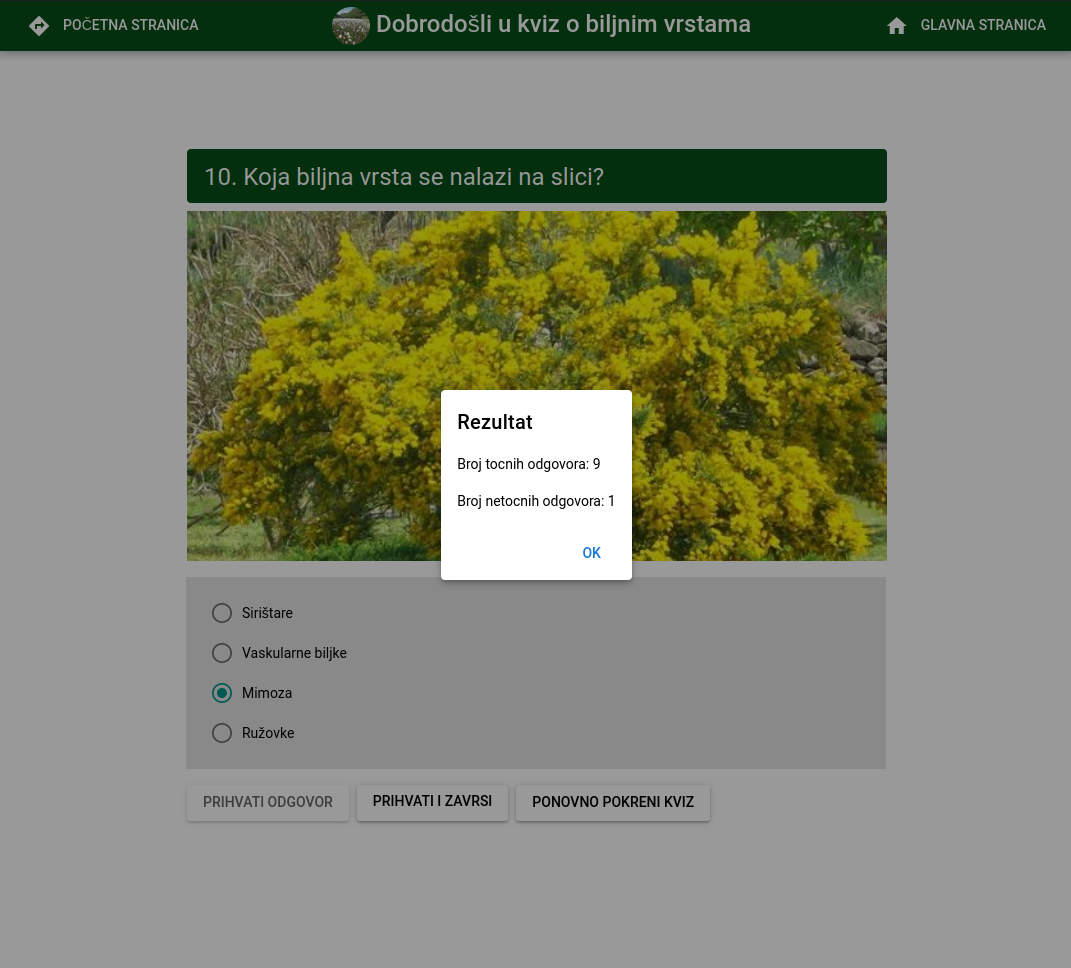
Slika 3 Kviz pitanja



Izvor: Obrada autora

Na slici 4 je prikaz zadnjeg pitanja u kvizu gdje je postavljeno pitanje, prikaz pop-up prozora s informacijom o konačnom rezultatu, koja služi da se korisnik informira o broju točnih i netočnih odgovora u kvizu.

Slika 4 Završno pitanje i pop up prozor



Izvor: Obrada autora