Gimnazija „Fran Galović“ Koprivnica

Bugy

Tehnička dokumentacija

Autorica: **Leticija Crnković**

Mentori: Kristina Ledinski, Bernard Crnković

Sadržaj

1. Uvod - što je *Bugy*?

Bugy je aplikacija namijenjena entomolozima, biolozima svim ljubiteljima kukaca. Služi za bilježenje klasifikacije, podataka i zapisa o pojedinom kukcu. Aplikacija je vrlo intuitivna; korisnici na efikasan način mogu mijenjati i čitati podatke o svim kukcima koji postoje u bazi od ukupno 11,230 jedinki te će se taj broj povećati na oko 800,000 dok dodam sve poznate vrste kukaca. Izmjene u podacima o nekom kukcu vidljive su svim korisnicima. Na taj se način lako može doći do najnovijih informacija i biti u toku s promjenama.

Aplikacija i server u potpunosti su napisani u *Javi*, s time da su neke početne skripte koje su bile potrebne pisane u *Pythonu*. Spomenute su u kasnijim poglavljima.

1.1. Inovativnost

Ako pretražujemo postojeća rješenja možemo uočiti da na *Google Play Storeu* postoje aplikacije koje omogućavaju korisnicima pregled slike i podataka o nekom kukcu, no ne nude korisniku mogućnost uređivanja kukca, stvaranja svoje osobne kolekcije i tome slično. *Bugy* nudi mnogo korisnih stvari koje su od velike važnosti istraživačima koji se bave kukcima te im može znatno podići produktivnost rada. Štoviše, korisnicima je omogućeno uređivanje već postojećeg kukca, dodavanje istoga u njihovu osobnu kolekciju i vođenje osobnih zapisa o nekom kukcu.

1.2. O autorici

Zovem se Leticija. Pohađam 4. razred matematičke gimnazije u Koprivnici. U ljeti 2019. počela sam učiti Javu, *scraping* podataka, rad sa serverom, *SQL*, *JSON* i druge tehnologije te mi je ovo prva *Android* aplikacija. Važno je spomenuti da mi je najdraža hrana čokoladni bananko. Ovu sam aplikaciju programirala radi vježbe i zato što su mi kukci zanimljivi. Izgledaju kao mali robotići. Osim programiranja, volim svirati violinu i klavir te gledat japanske animirane crtiće.

2. BAZA NA SERVERU

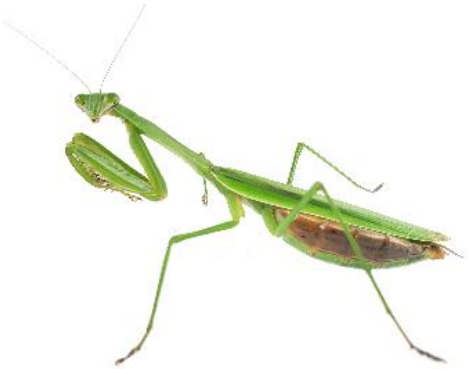
Bugy se sastoji od dvije komponente; aplikacije i serverske strane koja je neophodna za funkcioniranje aplikacije. Na serveru su spremljeni podaci o kukcima, njihove slike, podaci o korisnicima te brojni drugi resursi koje server koristi za posluživanje korisnika. Ovo je poglavlje namijenjeno opisu prikupljanja podataka te modela baze koja se nalazi na serveru, a u kojoj se nalaze podaci o kukcima.

2.1. *Scraping* informacija

Podatak koji mi je trebao, a specifičan je za svakog kukca jest njegova klasifikacija. To je skupina taksonomskih kategorija poredanih po hijerarhijskoj ljestvici.

Primjer klasifikacije Europske bogomoljke:

DOMENA: *Eukarya*
CARSTVO: *Animalia*
KOLJENO: *Arthropoda*
POTKOLJENO: *Hexapoda*
RAZRED: *Insecta*
RED: *Mantodea*
PORODICA: *Mantidae*
POTPORODICA: *Mantinae*
ROD: *Mantis*
VRSTA: *religiosa*



Slika 2.1. Europska bogomoljka

Za početak sam trebala klasifikacijske podatke o kukcima kako bih napunila bazu.

Odlučila sam se za automatizirano uzimanje podataka s jedne *web* stranice, tj. *web scraping*.

To je najefikasniji način dohvaćanja velike količine podataka s interneta. Web scraping radi na principu pronalaženja uzorka u *HTML*-u stranice (pomoću alata kao što su *Scrappy* ili *Beautiful Soup* knjižnice za Python) te ponavljanje postupka ekstrakcije zanimljivih podataka za velik broj stranica istog *layouta*. To može olakšati posao za, u mojem slučaju, približno 13,288 sati. Dakle, izvor podataka koje imam u bazi bila je stranica: https://bugguide.net

Ta domena ima Creative Commons licencu što znači da se podaci smiju koristiti, no uz njih se mora navesti izvor. *Bugy* poštuje intelektualno vlasništvo te je kod pregleda svakog kukca moguće vidjeti poveznice na izvorne stranice. Isto vrijedi i za slike.

Napomena: Scraping je bila jedina opcija jer stranica nema programski API koji bi mogla pozivati u svojoj aplikaciji.

Moja skripta napisana je u *Pythonu* i koristi knjižnicu *BeautifulSoup.* Slanje upita nekom *web serveru* tipično je vrlo skupo (vremenski) za velik broj poziva te sam zbog toga skinula *HTML* stranice na kojoj su bile klasifikacije svih kukci koje je ta domena posjedovala i onda parsala s nje. Na taj način sam uštedjela dosta vremena.

Nadalje, iz skinutog *HTML*-a iterativnom metodom ekstrahirani su podaci za svakog kukca i spremani u *JSON* format u novu datoteku na serveru. Nakon toga sam podacima iz *JSON* datoteke (također skriptom) punila *SQL* bazu. Zašto nisam odmah punila bazu, nego prvo u *JSON* datoteku pa tek onda u SQL bazu? Zato što je postojala mogućnost pojavljivanja grešaka pri izvršavanju skripte za *scrapeanje*. Greške se na kraju nisu događale te je baza uspješno popunjena.

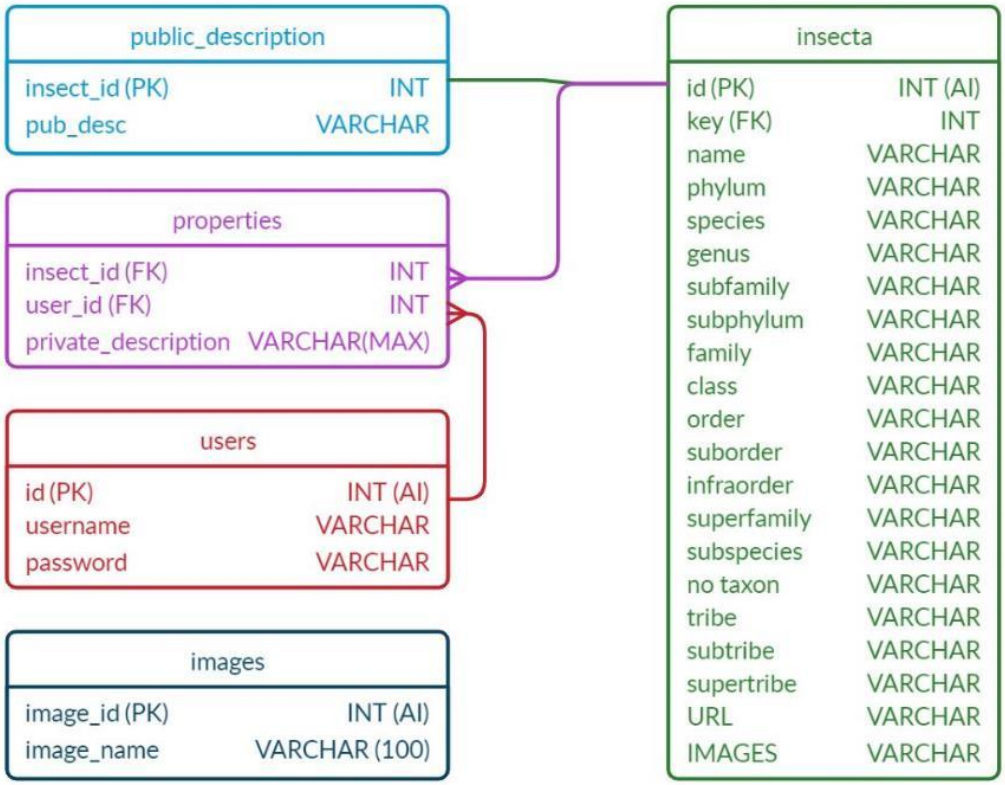
2.2. *Scrapeanje* slika kukci

Kako bi aplikacija bila još bogatija, potrebne su slike kukaca. Njih sam uzimala s istog web mjesta, dakle: https://bugguide.net i spremala u mapu na serveru. Naravno, potrebno je na neki način povezati svakog kukca u *SQL* bazi s njegovim pripadajućim slikama u mapi. U suprotnom slike nemaju smisla. Svaki kukac u bazi imao je generirana imena svih svojih slika (*UUID*) te su ta imena spremljena u *SQL* tablici na serveru. Tim istim imenima nazivane su i slike koje sam skriptom preuzimala s interneta i spremala ih u mapu. Na taj sam način povezala slike s kukcima. U sljedećem ću poglavlju detaljnije opisati bazu na serveru koju sam malo prije spominjala.

2.3. ERA model baze

Za bazu podataka koristim *h2 database engine*. To je jedna jednostavna open source *Java* implementacija *SQL* baze. Omogućava efikasan pristup informacijama i lak upis podataka. Imala sam nekoliko problema s korupcijom baze te sam odlučila u budućnosti prijeći na *SQLite* alternativu koja je brža i stabilnija (*JDBC API* je jednak za obje opcije tako da tranzicija nije problematična). Linkovi za download drivera za ovu bazu nalaze se na stranici: https://www.h2database.com/html/main.html

Bazu sam podijelila u 5 tablica: *insecta, users, properties, public_description i images*. Tablica *insecta* sadrži klasifikaciju svih kukaca te potrebne podatke koje koriste serveru. Tablica *users* služi za čuvanje podataka o registriranim korisnicima. U tablici *properties* nalaze se vanjski ključevi koji povezuju *ID* korisnika sa svakim kukcem kojeg je dodao u svoju kolekciju te opis tog kukca (ako je išta napisao) kojeg će u aplikaciji vidjeti samo taj korisnik. Tablica *public_description* sadrži *ID* kukca te njegov opis. Kada će svi kukci imati svoje opise, ova tablica će imati isti broj redaka kao i tablica *insecta* u kojoj su svi kukci. Zadnja tablica *images* sadrži imena slika kukaca te je potrebna serveru.



Slika 2.2. ERA model baze podataka

3. SERVER

3.1. O serveru

Prije opisa same *Bugy* aplikacije, želim predstaviti server koji je zapravo u njenoj pozadini. Server je poveznica između baze i korisnika aplikacije. Njegova je primarna zadaća upravljati upitima korisnika. Trenutno je *Bugy* server na mom laptopu (no postoji i live verzija na mom serveru). U budućnosti ga planiram preseliti negdje drugdje gdje neće biti *downtimea*. Za stvaranje servera koristim implementaciju *HTTP* servera (paket *com.sun.net.httpserver*). *HttpServer* vezan je za *IP* adresu te sluša dolazne *TCP* konekcije od klijenata na zadanom portu. Server može imati mnogo konteksta (*REST* endpoint). *HttpContext* je poveznica između *URI*-ja servera (*IP* adresa:port) i *HttpHandlera* koji obrađuje korisnikov zahtjev. Dakle korisnik šalje zahtjev jednome od mogućih konteksta pa kontekst poziva *HttpHandler* s kojim je povezan i koji se dalje brine za *response*, tj. odgovor koji je dužan poslati korisniku.

Knjižnice koje koristim na serveru:

https://commons.apache.org/proper/commons-codec/download_codec.cgi
https://mvnrepository.com/artifact/org.json/json/20180813
https://mvnrepository.com/artifact/com.h2database/h2/1.4.197
https://www.oracle.com/technetwork/java/javamail-1-4-1-141959.html

3.2. Upiti

Requestovi ili upiti su korisnikovi upiti serveru na koje očekuje određeni *response*. Oni se, baš kao i *responseovi* sastoje od 2 dijela: *header* i *body*. U *headeru* se obično šalje veličina *bodyja* i podaci vezani uz konekciju, no mogu se slati i neke dodatne manje informacije. *Body* je veći od *headera* i sadrži više podataka kojima primatelj dalje rukovodi.

3.3. Serverski konteksti

Bugy korisniku daje mogućnost da stvori svoj račun, dodaje kukce u kolekciju, pretražuje bazu kukaca, stvara zapise o kukcima, dodaje one koji još ne postoje u bazi, *upload* slika i ostale bogate mogućnosti. *Bugy* ima prilično opsežan *API* pa ću, radi bolje preglednosti, njih, njihove pripadajuće *REST endpointove* i opise prikazati tablično.

Tablica 1. Specifikacija Bugyjevog REST-API-ja

KONTEKST	HANDLER	PRIMA	ZADATAK	VRAĆA
/register	*SignInHandler()*	*username, password*	Provjeriti postoji li korisnik u bazi. Stvoriti objekt *User* i dodijeliti mu *sessionId*.	Ako je *logIn* uspješan, vraća *sessionId* koji mu je dodijelio. U suprotnom vraća *false*.
/login	*SignUpHandler()*	*username, password, e-mail*	Provjeriti postoji li korisnik u bazi. Stvoriti objekt *User*, dodijeliti mu *sessionId*.	Ako ne postoji korisnik s tim *usernameom* i *passwordom*, vraća *true*. U suprornom *false*.
/logOut	*LogOutHandler()*	*sessionId, allDevices*	Briše korisnikov *sessionId* s liste aktivnih korisnika. Ako je *allDevices==true*, briše sve *sessionIdjeve* koji su vezani na korisnika koji to traži.	Ako je *sessionId* valjan i sve je izvršeno, vraća *true*.
/home/searchBugs	*SearchInsectsHandler()*	*sessionId, insectID*	Prvo provjerava je li korisnikov *sessionId* valjan. Nakon toga izvodi *select* naredbu nad SQL bazom te pretvara dobiveni *resultSet* u JSON format.	Ako *sessionId* nije valjan, vraća *false*. U suprotnom vraća JSON objekt sa svim rezultatima pretvoren u *String*.
/home/getPublicDescription	*PublicDescriptionHandler( )*	*sessionId, insectID*	Provjerava je li *sessionId* valjan. Nadalje, poziva funkciju koja vraća *public_description* nekog kukca prema njegovom ID-ju u JSON formatu.	Ako *sessionId* nije valjan, vraća: *false*. U suprotnom vraća JSON objekt s *public_descriptionom* pretvorenim u *String*.

/home/checksessionId	*sessionIdChecker()*	*sessionId*	Provjerava je li *sessionId* valjan.	Ako je *sessionId* valjan, vraća *true*. U suprotnom *false*.
/home/addToCollection	*InsectAdditionHandler()*	*sessionId, insectID*	Provjerava *sessionId*. U tablicu slabih entiteta *properties* dodaje korisnikov ID koji uzme iz *User* objekta i *insectID* od kukca kojeg korisnik želi dodati u svoju kolekciju.	Ako je *sessionId* valjan te je uspješno izvršeno dodavanje kukca u kolekciju, vraća: *true*. U suprotnom vraća *false*.
/home/removeFromCollection	*InsectRemovalHandler()*	*sessionId, insectID*	Provjerava točnost *sessionIdja*. Iz tablice *properties* briše redak koji sadrži korisnikov ID i *insectID* kojeg korisnik želi izbrisati iz svoje kolekcije.	Ako je *sessionId* valjan te je uspješno izvršeno brisanje kukca iz kolekcije, vraća *true*. U suprotnom vraća *false*.
/home/getProperties	*UserPropertiesHandler()*	*sessionId*	Provjerava točnost *sessionIdja,* poziva funkciju koja uzima korisnikov ID iz *User* objekta i sve kukce koji pripadaju tom korisniku zajedno s njihovim osobnim opisima stavlja u JSON.	Ako je *sessionId* valjan te je JSON pravilno formiran, vraća JSON objekt u obliku *Stringa*. U suprotnom vraća *false*.
/home/getBugData	*BugDataHandler()*	*sessionId, insectID*	Provjerava valjanost *sessionIdja*. Funkcija (koja je za to zaslužna) uzima podatke (*data*) o traženom kukcu (s *insectIDjem)* iz *insecta* tablice i pretvara to u JSON objekt.	Ako je *sessionId* valjan te je JSON pravilno formiran, vraća JSON objekt u obliku *Stringa*. U suprotnom vraća *false*.

/home/uploadTextualChanges	*TextUploadHandler()*	*sessionId, insectID, body-length, body*	Provjerava točnost *sessionIdja.* Uzima JSON iz *bodyja,* parsa ga i zapisuje nove podatke u tablice: *insecta, public_description* i, ako je kukac u korisnikovoj kolekciji, tablicu *properties.*	Ako je *sessionId* valjan te su podaci ispravno *updateani,* vraća *true.* U suprotnom vraća *false.*
/home/getSubGroups	*ClassificationHandler()*	*sessionId, insectID*	Provjerava valjanost *sessionIda.* Iz tablice *insecta* uzima sva imena klasifikacijskih grupa kojima je nadređena ona tražena u *requestu.* Stvara JSON objekt svih rezultata.	Ako je *sessionId* valjan, vraća JSON s klasifikacijskim grupama u obliku *Stringa.* U suprotnom vraća *false.*
/home/uploadNewInsect	*NewInsectToBaseHandler()*	*SessionId, body-length, body*	Provjerava valjanost *sessionIdja.* Provjerava postoji li takav kukac već u bazi. Ako ne postoji, dodaje novog kukca u tablicu *insecta* i puni je parsanim podacima. Tako i s tablicom *public_description.*	Ako je *sessionId* aktivan i svi podaci su upisani u tablicu, vraća ID koji je dodijelio novom kukcu. U suprotnom vraća *false.*
/home/uploadPhotos	*PhotoHandler()*	*sessionId, body-length, body*	Provjerava *sessionId.* Generira ime slici te sliku koju je korisnik uploadao uzima iz *bodyja* i sprema u folder sa slikama.	Ako je *sessionId* ispravan i slika zapisana, vraća *true.* U suprotnom vraća *false.*
/home/loadImage	*LoadImage()*	*sessionId, imageName*	Provjerava *sessionId.* Učitava traženu sliku iz svog foldera u *File* objekt.	Ako je *sessionId* ispravan, zapisuje sliku

				u *outputStream*. U suprotnom vraća *false*.
/home/updateCredentials	*CredentialsUpdater()*	*sessionId, newUsername, newPassword*	Provjerava *sessionId*. Ako ne postoji korisnik koji već ima traženi *username*, mijenjaju se ti korsnikovi podaci u tablici *users*.	Ako je *sessionId* ispravan, upisuje podatke u tablicu *users* i vraća *true*. U suprotnom vraća *false*.
/home/changeNotificationsSettings	*NotificationsHandler()*	*sessionId, notifications*	Provjerava *sessionId*. U korisnikove postavke u tablici *users* stavlja traženo.	Ako je *sessionId* ispravan, napravi sve traženo i vraća *true*. U suprotnom vraća *false*.
/home/getSettings	*SettingsHandler()*	*sessionId*	Provjerava *sessionId*. Uzima korisnikove postavke iz tablice *users* (one su zapisane u JSON formatu)	Ako je *sessionId* ispravan, vraća korisnikove postavke u JSON formatu. U suprotnom *false*.
/home/deleteAccount	*DeleteAccountHandler()*	*sessionId*	Provjerava *sessionId*. Zatim prvo iz tablice *properties* briše sve retke u kojima je korisnikov *ID*, a nakon toga izbriše i korisnika iz tablice *users*.	Ako je *sessionId* ispravan, i sve je ispravno izvršeno vraća *true*. U suprotnom *false*.
/home/getUserCollection	*CollectionInsectsHandler()*	*sessionId*	Provjerava ispravnost *sessionIdja* i us pomoć *join SQL* naredbe povezuje *id* kukca koji se nalazi u korisnikovoj kolekciji s informacijama o tom kukcu u drugoj tablici.	Ako je *sessionId* ispravan, vraća kukce i njihove podatke. U suprotnom vraća *false*.

3.4. SQL naredbe

Za upisivanje i čitanje podataka iz *SQL* baze koriste se *SQL statementi.* Postoje obični *SQL statementi* i oni malo drukčiji *prepared statementi.* U većini slučajeva ja koristim ove druge zato što su efikasniji i puno sigurniji za bazu. Neke od njihovih prednosti su:

1) Omogućavaju ubacivanje teksta na prije određena mjesta unutar *statementa.*
2) Unaprijed su sastavljeni (jednom), te su brži za iterativno izvršavanje dinamičkog SQL-a gdje se parametri mijenjaju.
3) 'Keširanje' *statementa* baze podataka povećava brzinu izvedbe.
4) Štite bazu od *SQL injection* napada, tj. ubacivanja *SQL* naredba od strane korisnika koji nadalje može jako naštetiti bazi.

Dakle, *Bugy* ima siguran unos podataka u bazu te tako korisnici nemaju ovlasti išta mijenjati u njezinoj strukturi i podacima. Osim kroz aplikaciju i to na kontroliran način.

4. *BUGY* ANDROID APLIKACIJA

4.1. O aplikaciji

Aplikacija je, kao i server, programirana u *Javi*. Radi efikasnijeg razvoja aplikacije koristila sam *Googleov* open source projekt *Android Studio*. To je IDE (*integrated development environment*) za razvoj *Android* aplikacija. Aplikacija implementira sve potrebne funkcije za komunikaciju između servera i korisnika te se brine da korisnik steče pozitivno iskustvo prilikom njenog korištenja. Sadrži neke osnovne korisne stvari koje bi koristile sakupljačima kukaca. U sljedećim ću poglavljima opisati i ukratko objasniti kako to programski funkcionira. Priložit ću slike koje pokazuju kako to konkretno izgleda iz korisničkog sučelja.

Funkcionalnosti:

1. Stvaranje vlastitog računa
2. Pretraga baze kukaca
3. Pregled podataka o kukcu
4. Pregled slika kukca (galerija)
5. Dodavanje kukca u vlastitu kolekciju
6. Uređivanje klasifikacije kukca i opisa
7. Dodavanje vlastitih zapisa o kukcu
8. Pregled vaše kolekcije, brisanje, edit
9. Dodavanje kukca koji još ne postoji u bazu (novootkrivena jedinka)
10. Postavke aplikacije (jezik, obavijesti, promjena korisničkog imena…)
11. Pomoć - korisnička podrška

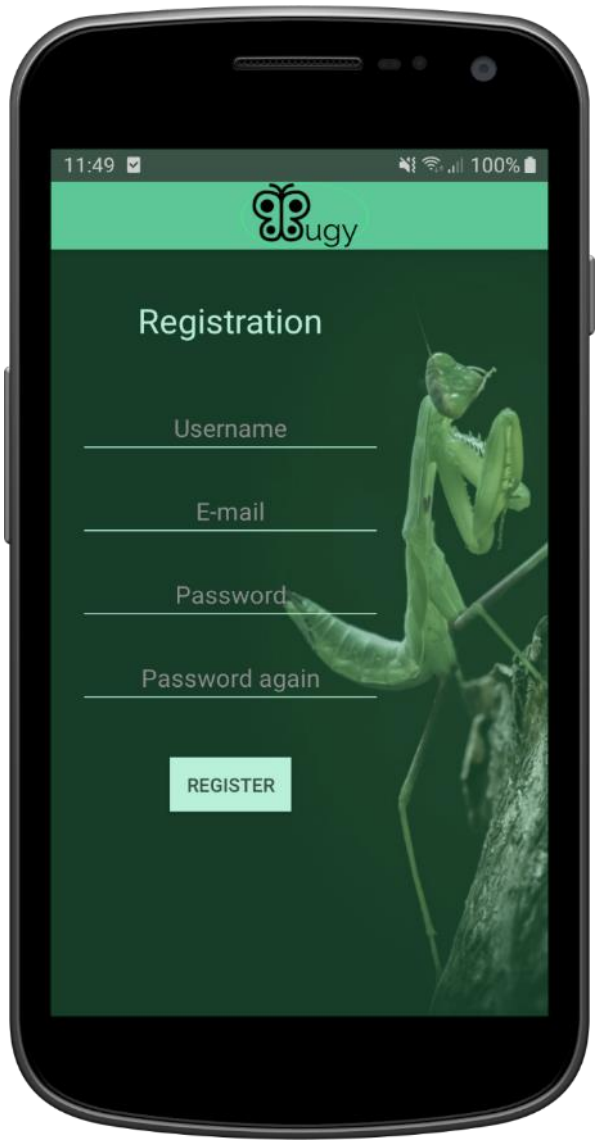
4.2. Zašto je aplikacija korisna?

Kada sakupljate kukce, želite znati neke podatke, imati njihove slike te voditi bilješke o svakome od njih. Ako se sve te stvari pišu na papir, zauzimaju mnogo prostora, znaju se izgubiti ili sakupljaju prašinu. Kako bismo izbjegli takve komplikacije, osmislila sam alternativni sustav upisivanja tih podataka o kukcima u računalo ili mobitel. Međutim, nailazite na problem u snalaženju među svim tim dokumentima koje ste napisali o kukcima. Bugy može riješti sve Vaše probleme! U sljedećim ću poglavljima ukratko nabrojiti i opisati sve korisne mogućnosti koje Bugy nudi. Objasnit ću na apstraktnoj, *high-level* UI razini

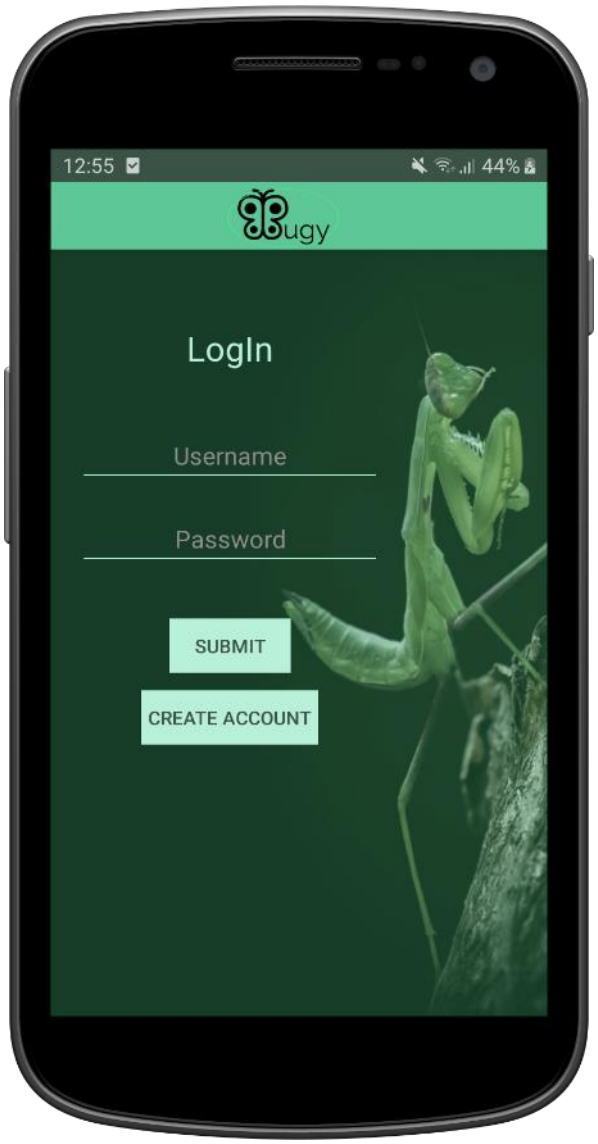
objasniti kako to programski funkcionira te ću priložiti i slike koje pokazuju kako to konkretno izgleda iz korisničkog sučelja.

4.3. Registracija i *login*

Svaka aplikacija koja omogućava stvaranje vlastitog korisničkog računa ima registraciju i *login*. *Bugy* također korisnicima omogućuje stvaranje vlastitog računa i druge stvari koje se kasnije nadovezuju na to. Na serveru se za registraciju i *login* brinu ranije navedeni konteksti u tablici.



Slika 4.1. Registracija



Slika 4.2. LogIn

Izbornik

Kada se uspješno registrirate ili logirate, pojavi se izbornik koji sadržava: *Help, Info, Settings, My Collection* i *Add Insects*. Svaki od njih vodi na sljedeći activity (korisničko sučelje s odgovarajućom klasom koja implementira *Androidov* AppCompatActivity) koji zatim radi nešto drugo.

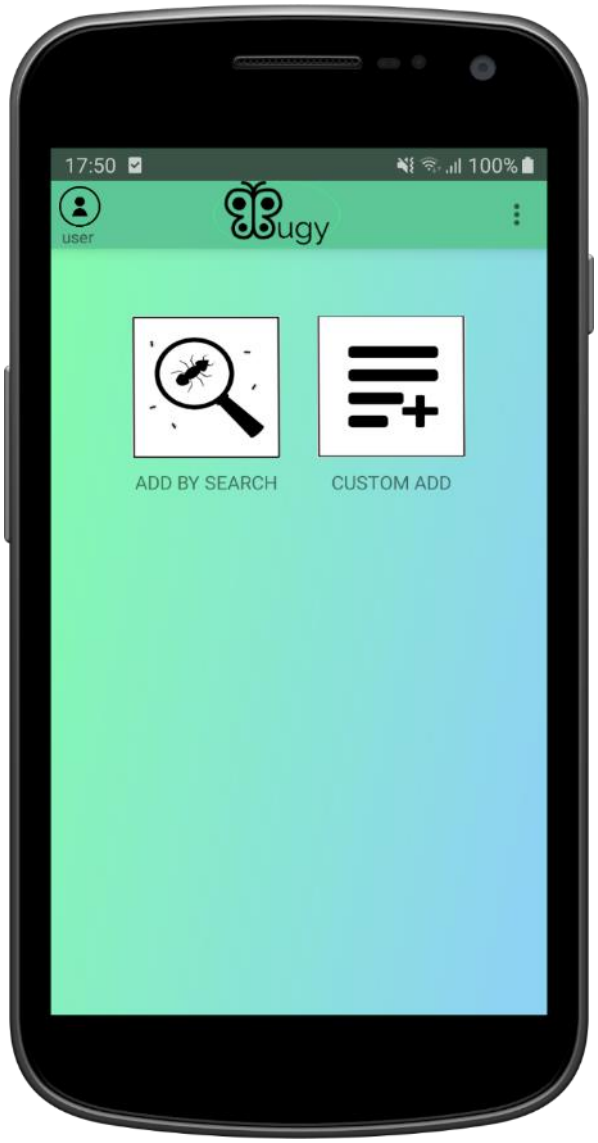
Slika 4.3. Glavni izbornik

Slika 4.4. Gornji izbornik

4.4. Pretraživanje baze kukaca

Korisnik ponekad želi dodati novog kukca u svoju kolekciju. U slučaju da kukac već postoji u bazi, omogućena mu je pretraga kojom lako može naći određenog kukca. Dakle, pošalje se upit serveru za pretragu kukca te on vraća sve rezultate koji se u jednom dijelu poklapaju s traženim kukcem. Kada aplikacija primi *response*, rezultati se izlistavaju u

*scrollView*u koji omogućuje korisniku *scrolling* po rezultatima. Kada korisnik dođe do dna *scrolla,* ponovno se šalje upit serveru te se učitava još rezultata. Ovako to izgleda u aplikaciji:



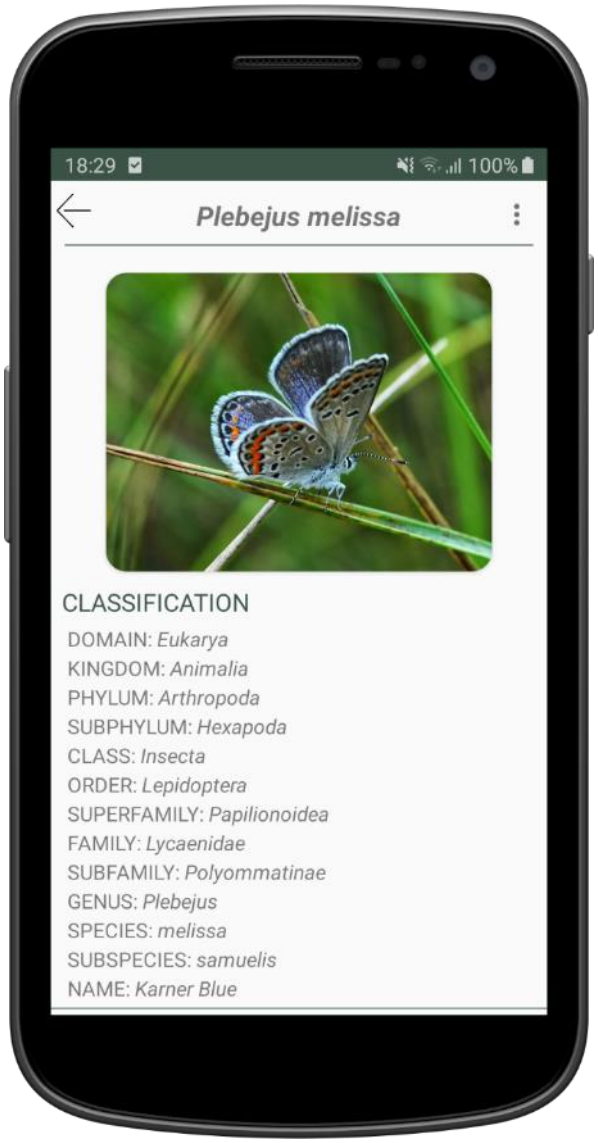
Slika 4.5. Izbornik pretrage baze

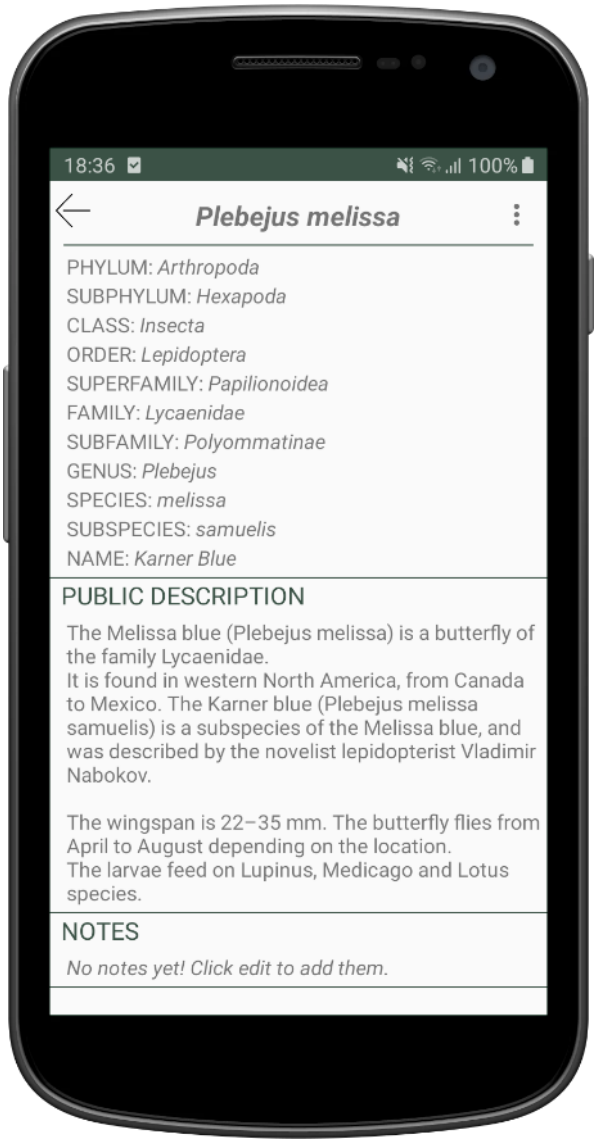
Slika 4.6. Primjer rezultata pretrage

Kod izlistavanja kukaca morala sam paziti da se prikažu slike koje su dobre za reprezentaciju. Na primjer, nisam mogla uzeti sliku kojoj je visina veća od širine jer radi estetike treba paziti da su sve kartice izlistanih kukaca približno iste veličine. Tako da je aplikacija uzimala slike koje odgovaraju uvjetima. Iz tog razloga učitavanje nekad potraje; više puta se mora *requestati* nova slika.

4.5. Pregled podataka o kukcu

Nakon što se izlistaju rezultati pretraživanja, korisnik može kliknuti na karticu kukca. Klik na kukca vodi na informacije o kukcu. Pri tome se u pozadini šalje upit serveru za podatke o tom kukcu. Nakon odgovora servera, izlistavaju se podaci. Dakle, na početku se pojavi slika kukca koja na klik vodi korisnika u galeriju tog kukca. Ispod slike nalazi se hijerarhijski ispisana klasifikacija. Nakon toga, ispisana je klasifikacija kukca pa opis kukca koji vide svi korisnici. Zato mu je dan naziv *public description*, a ispod njega su korisnikovi osobni zapisi ako ih je pisao, a omogućeno mu ih je pisati samo ako se kukac nalazi u njegovoj kolekciji (ako ga je prethodno dodao u kolekciju). Ovako izgleda opisan pregled kukca u aplikaciji:



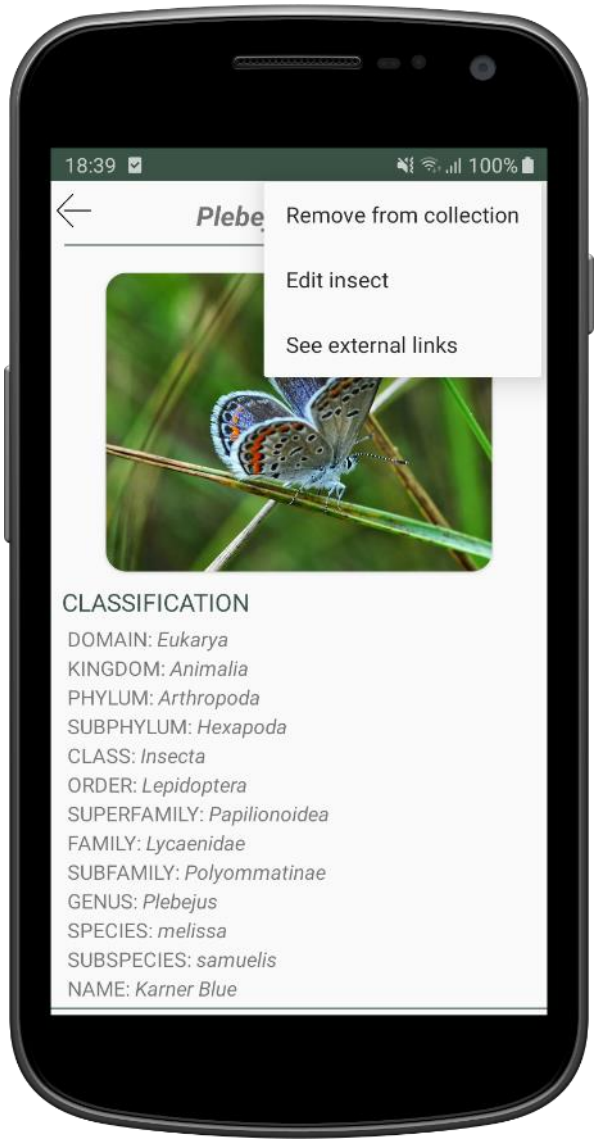


Slika 4.7. Primjer podataka o kukcu (BugInfoActivity) *Slika 4.8. Primjer podataka o kukcu*

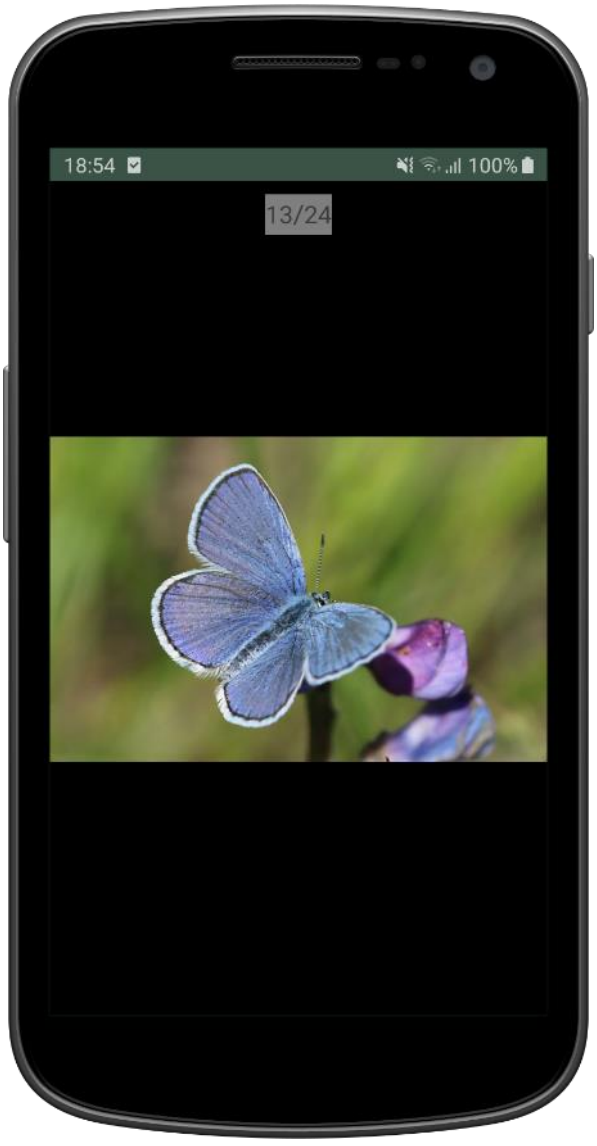
Naravno, ovaj *Activity* je moguće i *refreshati*. U slučaju da korisnik povuče *scrollView* skroz do vrha, zatražit će se osvježavanje stranice te će učinjene promijene postati vidljive.

4.6. Uređivanje podataka o kukcu

Kod informacija o kukcu u gornjem desnom kutu nalaze se tri točkice. Pri kliku na njih pojavljuje se padajući izbornik koji nudi *Edit insect, View external links* te dinamički element s nazivom *Add to collection.* On se, u slučaju da korisnik posjeduje kukca, mijenja u *remove from collection.* Ukratko, to se mijenja na način da se u pozadini zahtjeva (*requesta)* server, provjerava posjeduje li korisnik kukca te se tom gumbu mijenja naziv i prema tome mu se daje određena funkcija.



Slika 4.9. Opcije za svakog kukca

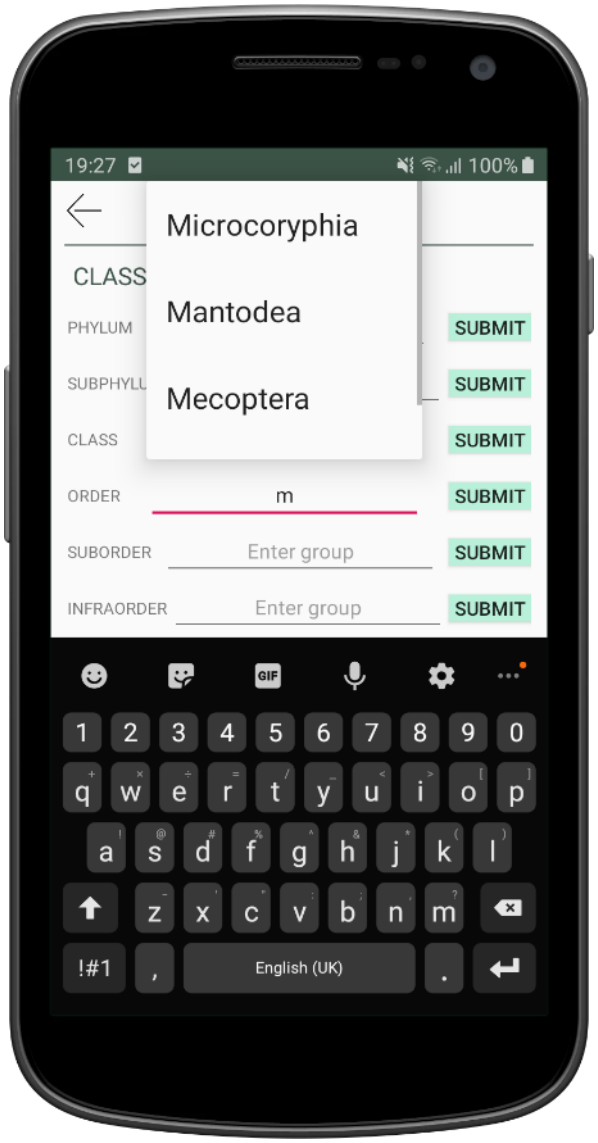


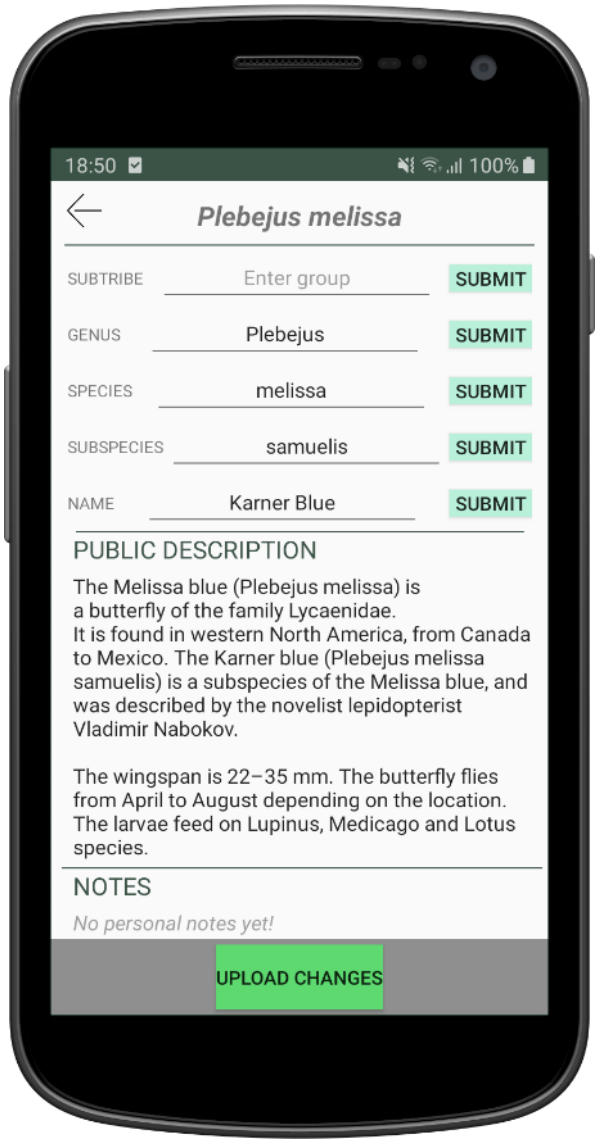
Slika 4.10. Galerija slika

Klikom na *edit insect* kod ovog *dropdown menua* na ljevoj slici pojavljuje se novi *Activity* u kojem je korisniku omogućeno mijenjati sve informacije koje su bile prikazane u prošlom *Activityju.*

1. Prvo što može dodati jest slika kukca. Sliku može odabrati iz galerije, *Google Diska*, a može i ovaj tren poslikati. Naravno, nakon odabrane slike omogućen je *crop* pa se tako može urediti slika.
2. Također možete mijenjati klasifikaciju. Kao što sam ranije spomenula, neke klasifikacijske grupe kod nekih kukaca nemaju naziv, a kod drugih te iste grupe imaju naziv. To spominjem zato što je kod uređivanja kukca omogućeno uređivanje baš svih klasifikacijskih grupa; i onih za koje kukac ima naziv, i onih za koje još nema. Kod klasifikacije sam omogućila još jednu vrlo korisnu stvar koja služi i korisnicima, a i bazi. Naime, pri ručnom unosu podataka (pogotovo nesvakidašnjih imena klasifikacijskih grupa) dolazi do grešaka. Zato je korisnicima ovdje omogućen *autocomplete*. Kod svake klasifikacijske oznake s desne strane postoji gumbić *submit.* On šalje upisanu klasifikacijsku grupu serveru nakon čega se, kad server odgovori, u *ArrayAdapter* učitavaju sva imena klasifikacijskih grupa koje su podređene onoj prethodnoj. Primjer: 'razredu' je podređen 'red'. 'Razred' *Insecta* ima više 'redova' koji su mu podređeni, to su, na primjer: *Mantodea, Microcoryphia, Zygentoma, Odonata, Zoraptera, Lepidoptera, Thysanoptera...* Tada će se svi ti 'redovi' ponuditi pri upisivanju te klasifikacijske grupe u *editu.* Naravno, prednost pri ponuđivanju rezultata imaju klasifikacijske grupe koje su slovima najsličnije upisanome. Korisnik nakon toga može kliknuti na ime klasifikacijske grupe koja mu odgovara i to polje se automatski ispuni. Na taj način može biti siguran da nije napravio pogrešku u pisanju.
3. Opis koji je vidljiv svim korisnicima također se može mijenjati. Možete dodavati korisne informacije te podijeliti znanje o nekom kukcu. To će se sve pri kliku na *save changes* spremiti na server te će nadalje biti vidljivo ostalim korisnicima koji će moći više naučiti o kukcu.
4. Mijenjanje ili dodavanje vlastitog opisa kukca omogućeno je samo ako ste ga prethodno dodali u svoju kolekciju. Ako niste, lako ga možete klikom na *add to collection* kod onog prošlog *Activityja* s podacima o kukcu (klik na tri točkice *add to collection*). Nakon toga kod uređivanja će se omogućiti dodavanje i vlastitog opisa kukca koji se također sprema u bazu na serveru.

Ovako to izgleda u aplikaciji:

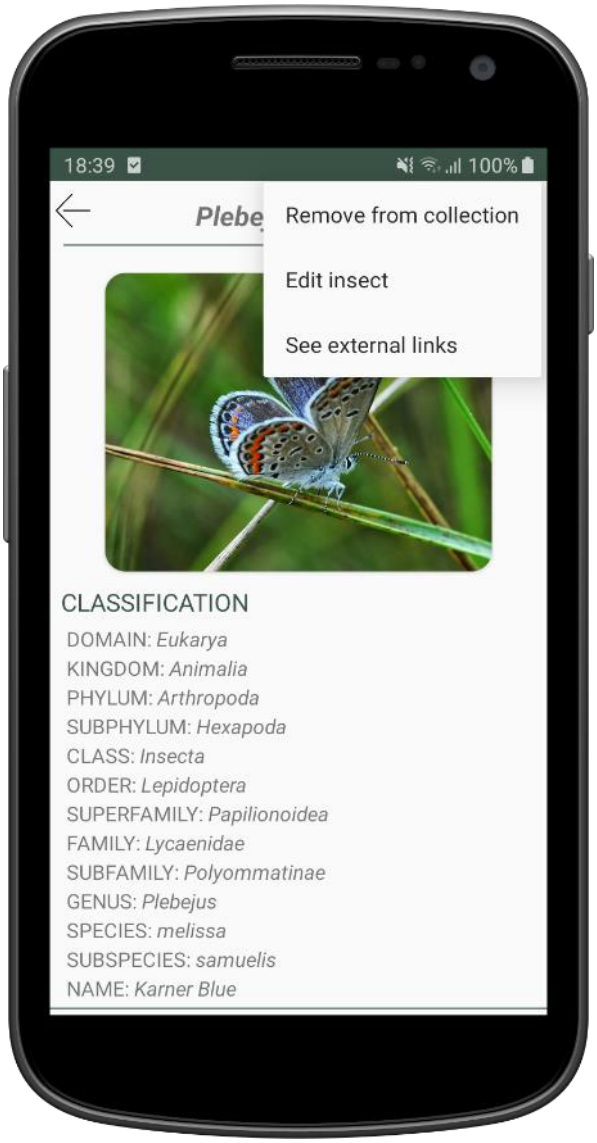




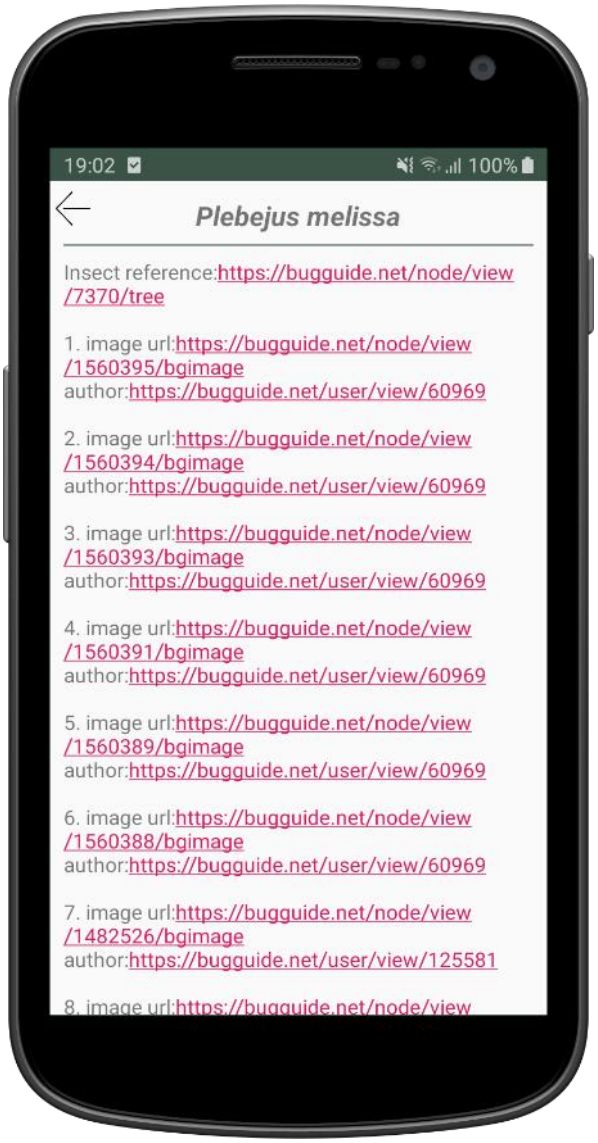
Slika 4.11. Autocomplete

Slika 4.12. Javni opis kukca i vlastite bilješke (ovdje prazne)

Klikom na *See external links* kod onog padajućeg izbornika pojavi se *Activity* s linkovima na slike i informacije o kukcu na koje možete kliknuti te vas to odvede na *web* stranicu.

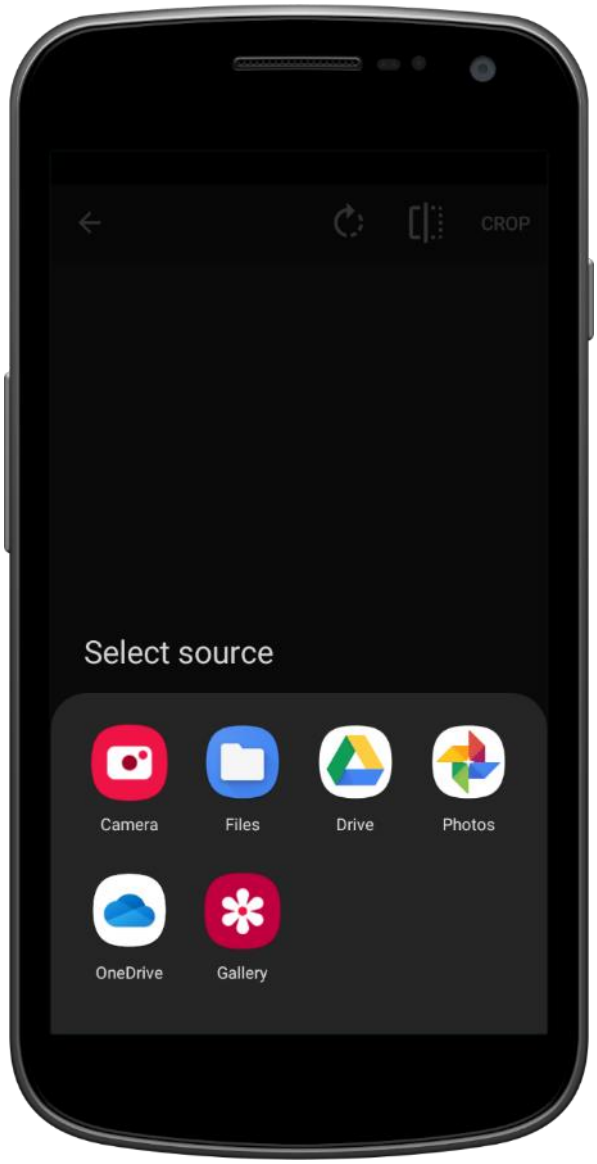


Slika 4.13. Opcija See external links

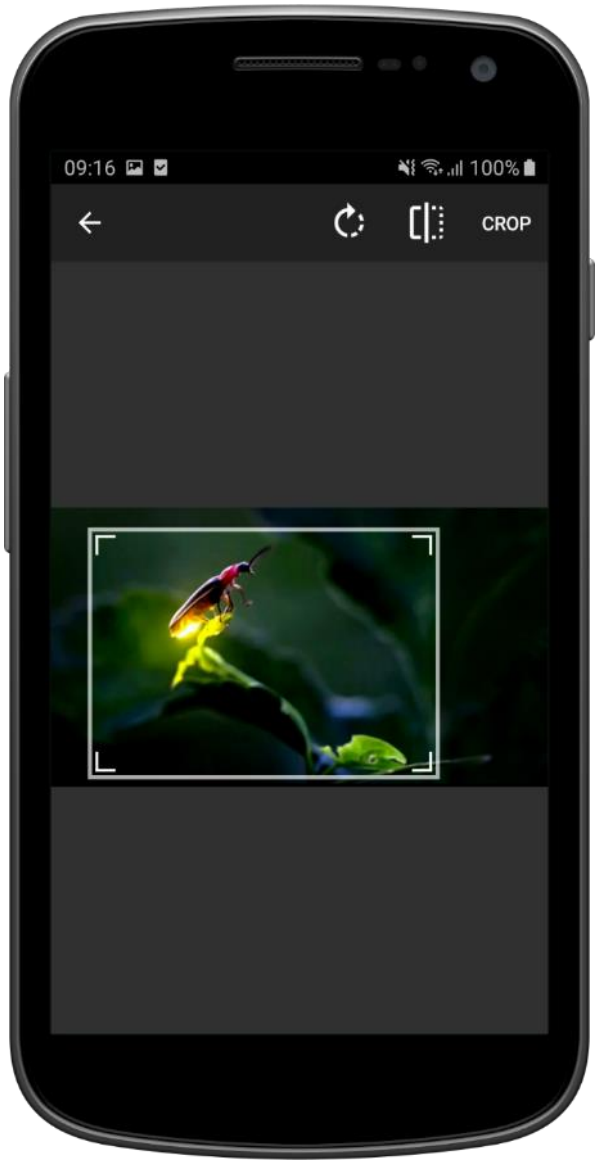


Slika 4.14. Primjer popisa svih izvora

Kod dodavanja slika korisniku je omogućeno odabiranje slike koju želi *uploadati*. Nakon toga pojavi se *crop Activity* u kojem po volji može rotirati i izrezati sliku. Kada kod *Edit insect Activityja* klikne *Upload changes*, ta će se slika zajedno sa svim tekstom uploadati na server te će pri sljedećem pregledu galerije tog kukca biti vidljiva svim korisnicima. Također, kod pregleda vanjskih linkova moguće je vidjeti da se kao autor te slike navodi upravo taj korisnik koji je uploadao sliku; njegov mail.



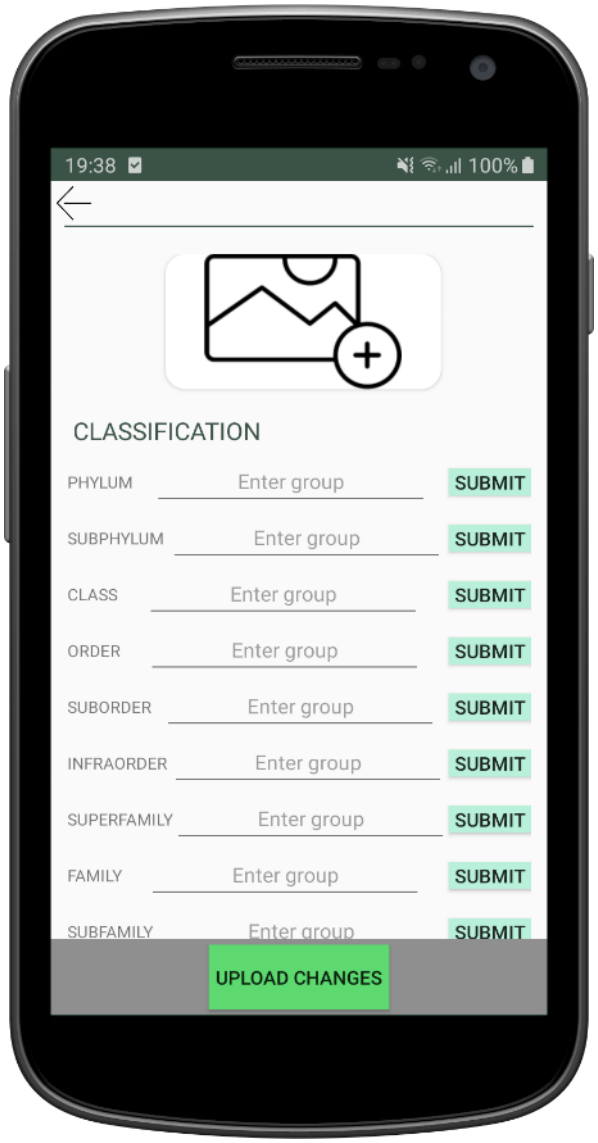
4.15. Dodavanje footografija



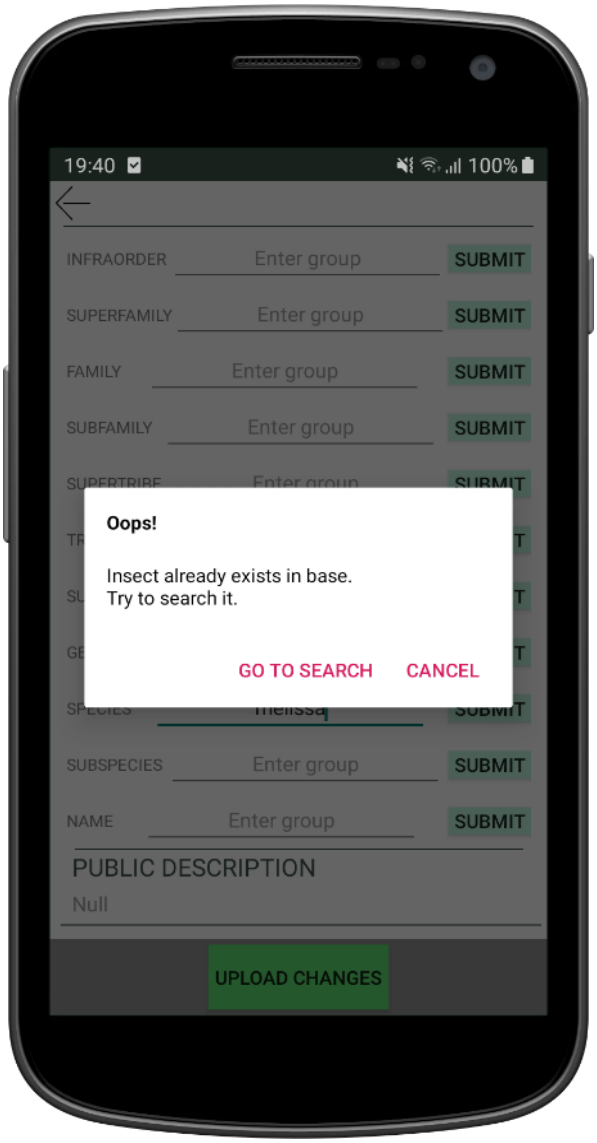
4.16. Crop fotografije

4.7. Custom add

Ako neki kukac još ne postoji u bazi, a korisnik ga želi u svojoj kolekciji, može ga dodati u bazu uz pomoć ovog *Activityja*. Sličan je izgled kao i kod uređivanja kukca. Ako korisnik ništa ne upiše i želi to dodati, naravno da ga aplikacija upozori. Mora dodati barem rod (*genus*) i vrstu (*specie*). Tada se taj kukac podrazumijeva kao nova vrsta. Nije nužno dodati sliku. To se može i kasnije. U ovom je *Activityju* korisniku također omogućeno instantno dodavanje tog novog kukca kojeg je upravo stvorio u bazi u njegovu vlastitu kolekciju. Ako pak korisnik nije prvo pretražio bazu i vidio da već takva jedinka postoji, baza će se pobrinuti za to te će ga aplikacija upozoriti. Za takvu interakciju s korisnikom brinu se dijalozi. Primjeri su ispod:



Slika 4.17. Dodavanje novog kukca duplikata



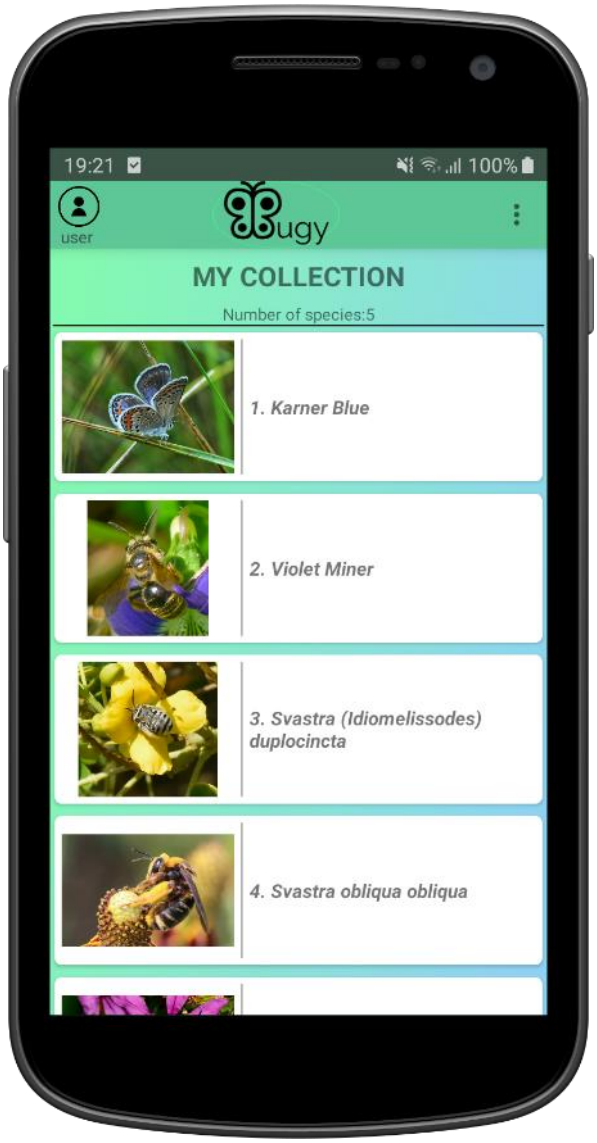
Slika 4.18. Obavijest u slučaju

4.8. Pregled kolekcije

Korisniku je omogućen pregled njegove kolekcije kukaca. Sve kukce koje posjeduje u svojoj kolekciji ispisat će se u ovom *Activityju.* Na početku se šalje upit serveru za prvih 5 kukaca, a kasnije kod svakog *scrolla* do dna ponovno se šalje upit te se učitava još više rezultata. Sličan princip koristim i kod pretraživanja baze u prethodno opisanom *Activityju.* Dakle, izlistaju se kukci i klikom na nekog od njih možete vidjeti informacije o kukcu, slike, opise… zapravo sve što se vidi i kod već prethodno opisanog *Activityja.* Klasu za taj *Activity* sam zapravo skroz ponovno ovdje iskoristila. Samo ju je trebalo jednom napisati i dalje je upotrebljiva više puta. Ako dobro planirate projekt, manje je ponavljanja u kodu. To pokušavam postići i ovdje sam to uspjela. Jedino što navedena klasa treba za učitavanje svih ostalih podataka je JSON objekt koji čuva podatke o određenom kukcu, a taj objekt joj se mora predati u *Intentu* (objekt koji služi za prelaženje između *Activityja*). Naime, kod prelaženja u novi *Activity* doslovno možete pohraniti *String* koji želite koristiti u sljedećem, s metodom *Intent.putExtra().* Na taj sam način iz *Activityja* s izlistanim kukcima na klik prenijela potrebne podatke u novi *Activity.* To koristim kod mnogo prijelaza između *Activityja,* no nije ih potrebno stalno spominjati.

Ovako izgleda izlistavanje kukaca iz korisnikove kolekcije (kod ovog sam korisnika dodala nepar kukci):

Slika 4.19. Glavni izbornik

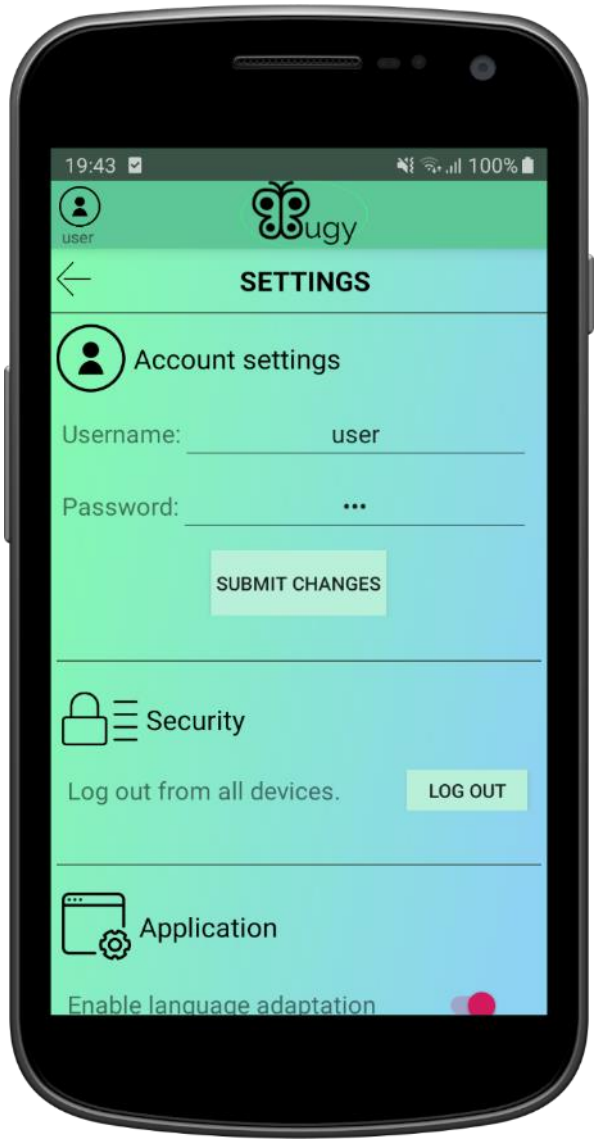


Slika 4.20. Primjer korisnikove kolekcije

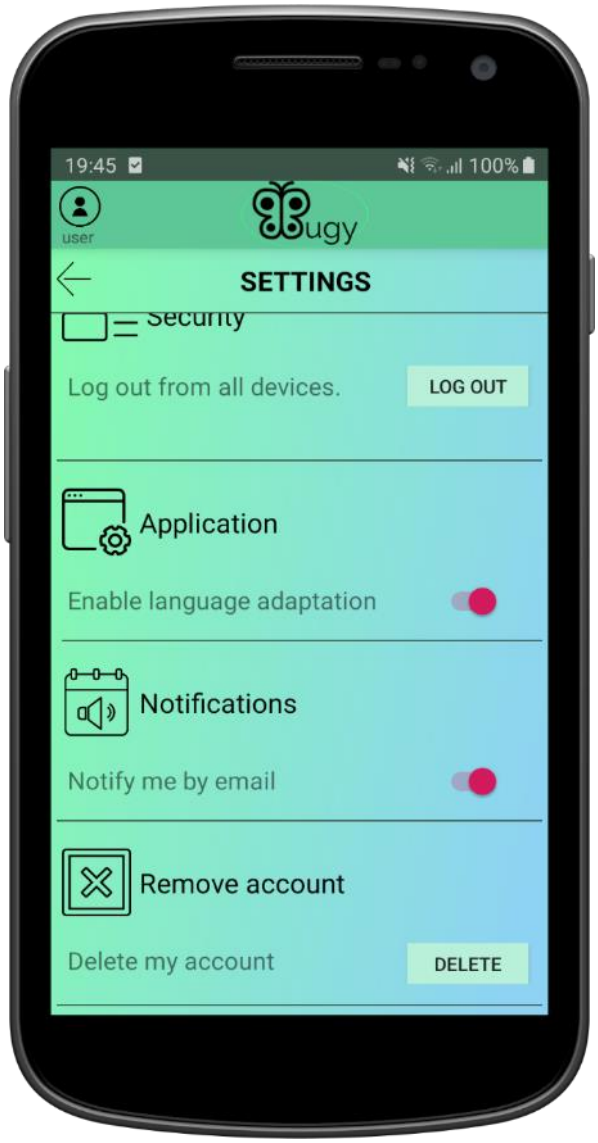
4.9. Korisničke postavke

Aplikacija također posjeduje neke postavke koje sam za početak smatrala neophodnima. Nabrojat ću što je korisniku omogućeno u postavkama:

- Mijenjanje korisničkog imena i lozinke.
- Odjavljivanje sa svih uređaja (osim ovog na kojem su trenutno *ulogirani*).
- Omogućavanje prijevoda u aplikaciji
- Omogućavanje slanja *mail* obavijesti od strane *Bugy* aplikacije.
- Brisanje korisničkog računa.



Slika 4.21. Korisničke postavke (1. dio)



Slika 4.22. Korisničke postavke (2. dio)

4.9.1. Mijenjanje korisničkog imena/lozinke

Kada korisnik želi promijeniti svoje podatke, upiše nove te se šalje zahtjev serveru za promijenu korisnikovih podataka. Naravno, uz to dolaze sve provjere validacije podataka (jesu li polja prazna, postoji li korisnik s takvim imenom već u bazi…) te se na temelju toga ispisuje poruka koja navodi korisnika na moguće pogreške. Podrazumijeva se da aplikacija korisnika *dialogom* ponovno pita je li siguran u svoju namjeru. Nakon promjene navedenih podataka korisnik se mora ponovno ulogirati (odveden je do logina) kako bi se dodatno potvrdile promjene.

4.9.2. Odjavljivanje sa svih uređaja

Odjavljivanje sa svih uređaja na serveru se provodi na sličan način kao i odjavljivanje s trenutnog uređaja. Server na temelju *sessionIdja* može zaključiti o kojem se korisniku radi te odjaviti sve *sessionIdjeve* koji pripadaju tom korisniku.

4.9.3. Prijevod u aplikaciji

Bugy podržava više jezika. Ako je korisniku u postavkama mobitela naznačeno da je primarni jezik hrvatski, u aplikaciji će svi gumbovi, *menu* elementi sav tekst koji se generira u aplikaciji biti na hrvatskom jeziku. Svi jezici za koje *Bugy* ima prijevod na taj će se način prikazivati u aplikaciji. Da, to ne uključuje informacije o kukcu, klasifikaciju i ostale stvari koje korisnici unose, a vidljive su svima. Takav tekst mora biti svima razumljiv te se piše na engleskom s latinicom. Za sada *Bugy* podržava 4 jezika: hrvatski, engleski, njemački i japanski.

Slika 4.23. Primjer prijevoda na japanski jezik

4.9.4. Slanje obavijesti

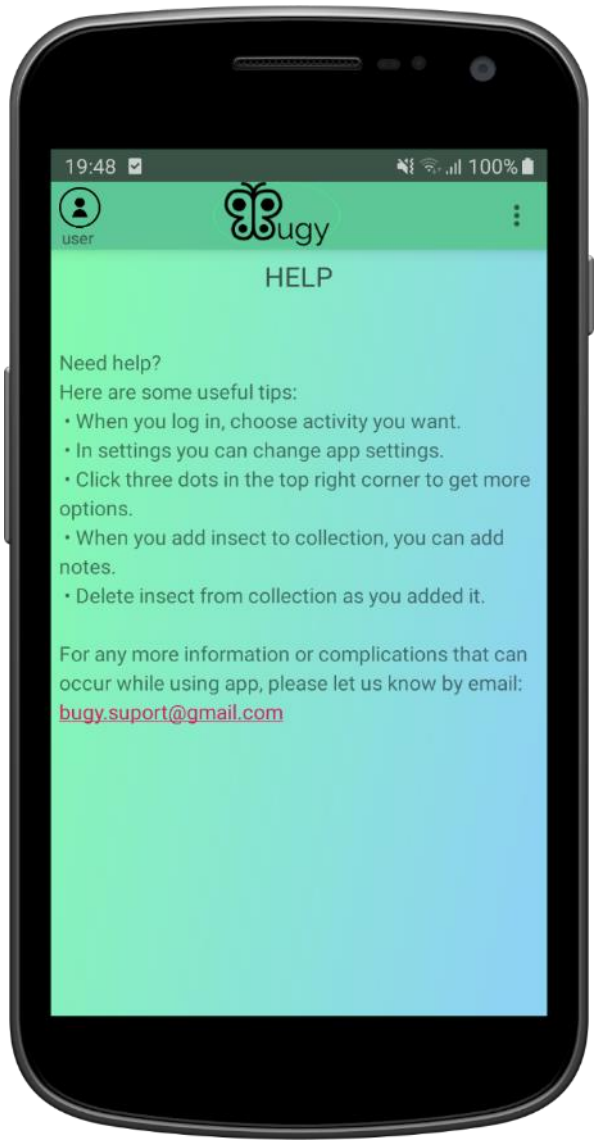
Po *defaultu* je u aplikaciji složeno da pri registraciji i loginu server pošalje obavijest korisniku na mail. To se radi iz sigurnosnih razloga, no korisniku je omogućeno uključivanje i isključivanje ove postavke. Postavke za svakog korisnika zapisuju se u bazu na serveru te se učitavaju kod odlaska u postavke korisnika.

4.9.5. Brisanje korisničkog računa

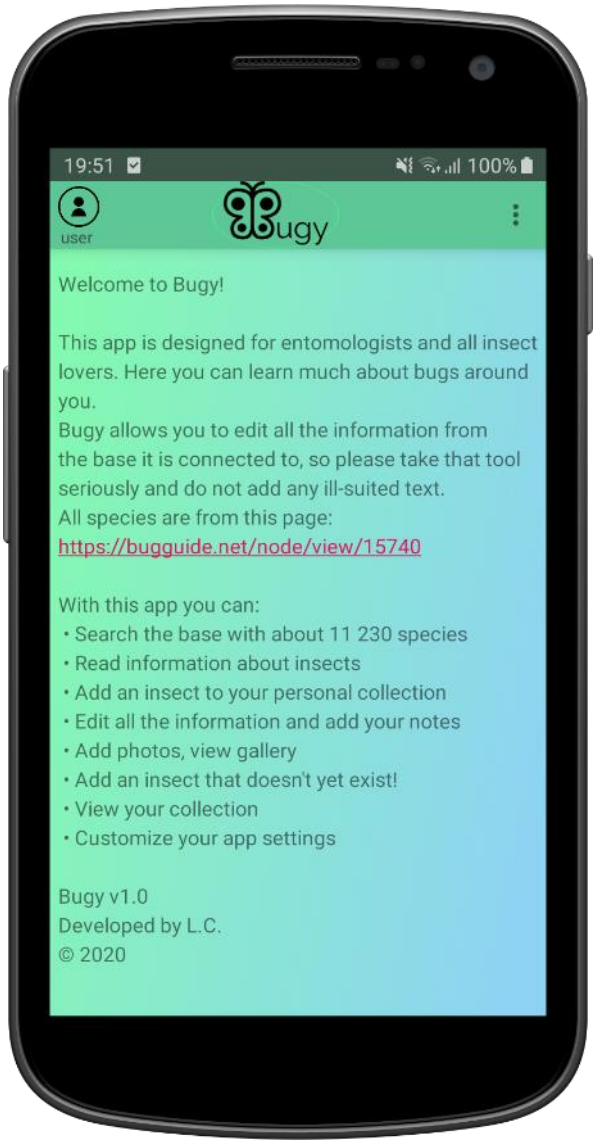
Za sada zadnja stvar koja je omogućena u postavkama aplikacije je brisanje korisničkog računa. Naravno, kod klika na taj gumb korisnika se ponovno pita je li siguran u svoju namjeru.

4.10. Help

U *Activityju* nakon logina pojavljuju se gumbi koje sam prethodno nabrojala. Jedan od njih je i *help*. Tu se nalaze početnička uputstva korisniku za korištenje aplikacije s primjerima i *screenshotovima.*



Slika 4.24. Pomoć

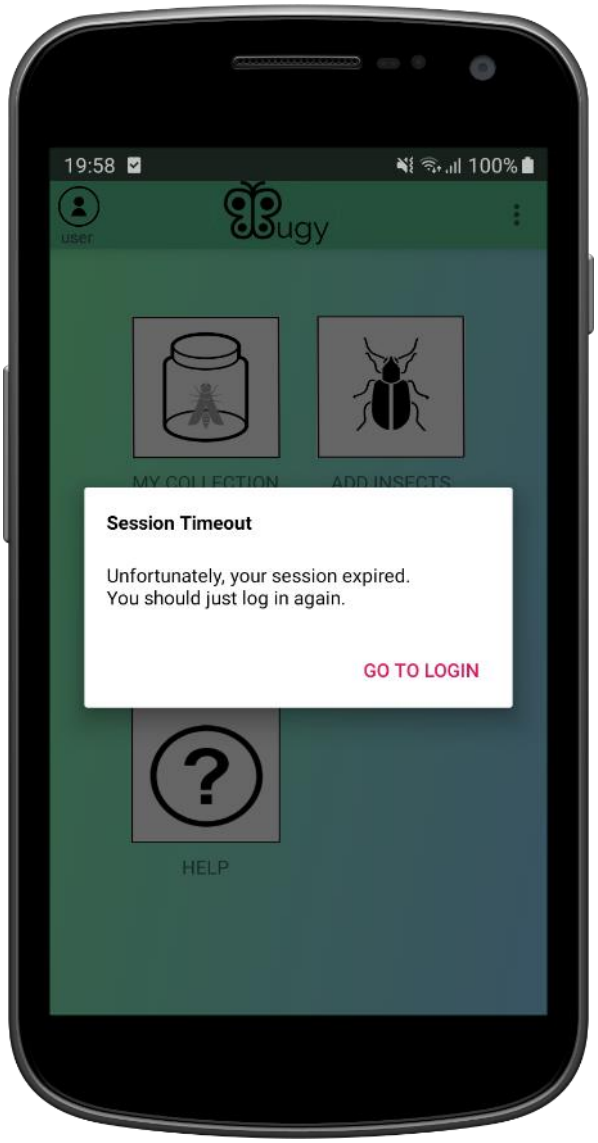


Slika 4.25. O aplikaciji

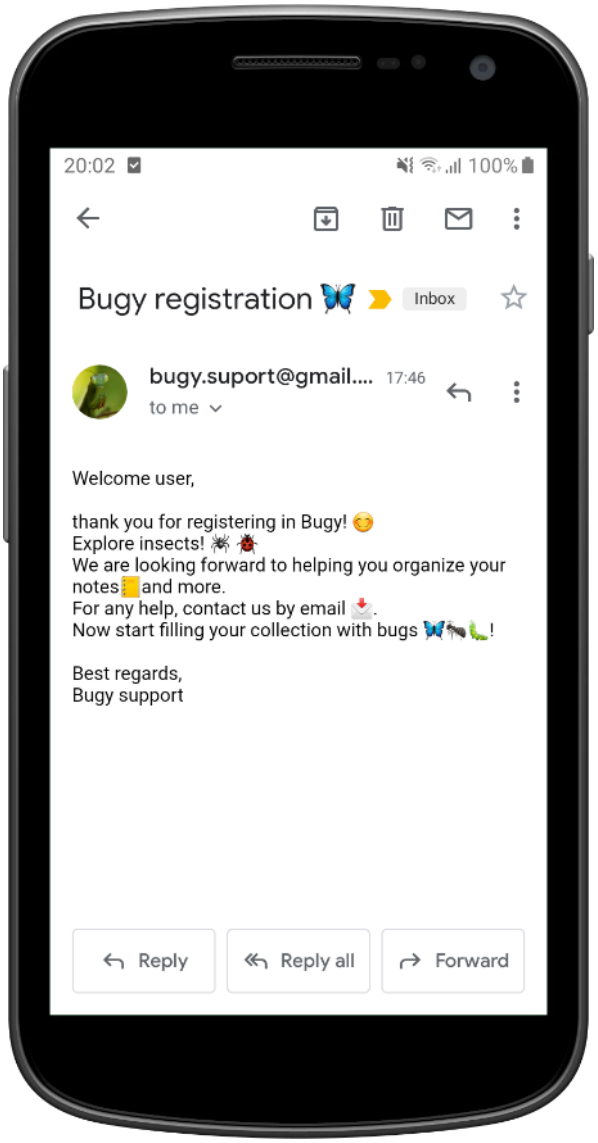
4.11. Info

U aplikaciju sam također dodala jedan *info Activity* (slika 4.25. na prethodnoj stranici). Ovdje korisnici mogu dobiti informacije o samoj aplikaciji kao što su: autor, pravila korištenja, linkovi, kontakt...

Kroz cijelu se aplikaciju svim *requestovima* provjerava i valjanost *sessionIdja.* To je zbog sigurnosnih mjera. Ako se u jednom trenutku utvrdi da je korisniku istekao *sessionId,* pojavit će se *dialog* kao na slici 4.26. Važno je dodati da pri registraciji i loginu *Bugy* šalje automatski generiran mail iz sigurnosnih razloga.



4.26. Obavijest o isteku sesije



Slika 4.27. Primjer automatskog e-maila

5. Planovi za budućnost

1. Stavljanje aplikacije na *Play Store*
2. Napraviti *iOS* i *Desktop* (*browser*) verzije aplikacije
3. Premještanje *Bugy* servera u *Docker* kontejner na neki *host* server radi nezavisnosti
4. Omogućiti korisnicima da stave profilnu sliku
5. Omogućiti sklapanje prijateljstva između korisnika te rad na zajedničkoj kolekciji.
6. Daljnji plan je analogno ovoj aplikaciji o kukcima napraviti aplikaciju za životinje ili bilo koju skupinu stvati koja ima kategorije
7. Dodati administrirajuće korisnike koji provjeravaju i potvrđuju ispravnost unešenih podataka prije spremanja u bazu
8. Prijeći sa *h2* na *SQLite*