

VELEUČILIŠTE U RIJECI
POSLOVNI ODJEL RIJEKA

Alen Abdulahović i Roberto Salamon

KOKOSHINJAC

PROJEKTNNA DOKUMENTACIJA

Rijeka, 2021.

VELEUČILIŠTE U RIJECI
POSLOVNI ODJEL RIJEKA

KOKOSHINJAC

PROJEKTNA DOKUMENTACIJA

Kolegij: Izgradnja objektno orijentiranih aplikacija

Mentor: Vlatka Davidović, viši predavač

Mentor: Ivan Šimac, asistent

Studenti: Alen Abdulahović (MBS: 2422000113/20) i Roberto Salamon (MBS: 2422000104/20)

Rijeka, svibanj 2021.

Sadržaj

1. Opis sustava (naziv poslovnog sustava)	1
2. Specifikacija zahtjeva	2
<i>Plan intervjua.....</i>	<i>2</i>
<i>Detalji o korisniku sustava.....</i>	<i>2</i>
<i>Zadaci korisnika.....</i>	<i>2</i>
<i>Identifikacija problema</i>	<i>3</i>
<i>Sumarizacija problema.....</i>	<i>4</i>
<i>Identificiranje nefunkcionalnih zahtjeva</i>	<i>4</i>
3. Analiza sustava.....	9
3.1. <i>Analiza zahtjeva i korištenje sustava</i>	<i>9</i>
3.2. <i>Odabir tehnologija.....</i>	<i>10</i>
4. Dizajn sustava	11
4.1. <i>Dizajn korisničkih sučelja.....</i>	<i>11</i>
4.2. <i>Dijagram klasa.....</i>	<i>13</i>
4.3. <i>Dijagram objekta</i>	<i>14</i>
4.4. <i>Model podataka</i>	<i>14</i>
4.5. <i>Sekvencijalni dijagram.....</i>	<i>14</i>
5. Implementacija sustava	16
5.1. <i>Postavljanje radnog okruženja</i>	<i>16</i>
5.2. <i>Verzije aplikacije.....</i>	<i>16</i>
5.3. <i>Prikaz dijelova programskog koda</i>	<i>17</i>
6. Problemi u kreiranju aplikacije.....	19
7. Isporuka i korištenje aplikacije	20
7.1. <i>Pakiranje i isporuka aplikacije</i>	<i>20</i>
7.2. <i>Korisničke upute za korištenje aplikacije.....</i>	<i>20</i>
Zaključak.....	21
Literatura i izvori	22
Popis slika	23

1. Opis sustava (naziv poslovnog sustava)

Projekt nastaje kao potreba za evidencijom, praćenjem i potpora u odlučivanju pri izvršavanju aktivnosti vezanih za sakupljanje jaja, hranjenjem kokoši, nabavom hrane te praćenjem podataka vezanih za ove aktivnosti na obiteljskom gospodarstvu. Sakupljanje jaja je aktivnost koja se provodi na dnevnoj razini, može se dogoditi više puta i unosi se odmah nakon sakupljanja. Nakon obavljenog sakupljanja u aplikaciju se unosi količina jaja koja je sakupljena. (Unos ne mora nužno biti isti dan, moguće je naknadno dodati jaja s naglaskom da nije moguće unaprijed unijeti vrijednosti). Osim unosa jaja planira se i bilježenje hranjenja, gdje je hranjenje periodička aktivnost u kojoj se hrana stavlja u hranilice. Hrana može biti u obliku mljevenih žitarica, suhog kruh s mekinjama ili kuhana kora od krumpira. Prilikom obavljanja aktivnosti hranjenja potrebno je evidentirati kolika se količina hrane i koja vrsta hrane daje (kg) zbog kasnije optimizacije. Cijena žitarica na tržištu je varijabilna te je potrebno omogućiti izračun cijene s promjenjivom vrijednosti što unosi korisnik sustava.

Do sada su ovi podatci, koji će biti prikupljeni i obrađeni, bili čuvani u papirnato obliku što je dovodilo do čestog gubitka informacija i vrlo otežanog načina vršenja analize. Ovim programskom rješenjem olakšalo bi se korištenje podataka te tako povećalo učinkovitost sustava.

Praćenje podataka odnosilo bi se na statističke podatke koje je moguće naknadno čitati.

Na primjer: Prosječni broj jaja u mjesecu, min i max potrošnje hrane u kg (za određeni period), ukupni troškovi, prihodi (ukupno jaja puta 2 kn)

1. Evidencija količine sakupljenih jaja
2. Evidencija potrošene količine hrane za kokice
3. Evidencija troškova
4. Pregled podataka

2. Specifikacija zahtjeva

Plan intervjua

Sustav:	Web aplikacija
Projekt:	Kokoshinjac
Učesnik(ci):	Djelatnici obiteljskog gospodarstva
Datum:	12.3.2021
Vrijeme:	19:54
Mjesto:	Bakar
Trajanje:	1h 12 min
Namjena:	Odrediti funkcionalnosti i probleme koji se rješavaju
Dokumenti:	

Detalji o korisniku sustava

Ime i prezime:	Ružica Nomalas
Firma/odjel:	OPG
Uloga u sustavu:	Voditelj OPG-a

Zadaci korisnika

Na koje načine akter koristi sustav?
Voditelj se brine o nabavi hrane, količinom sakupljenih jaja kao i o hranjenju kokica.
Koje zadatke izvršava/za koje je odgovoran?
Ostali djelatnici imaju zadatke da sakupljaju jaja i hrane kokoši. Najčešće do ovih aktivnosti dolazi oko 2 puta dnevno.
Kome je odgovoran za izvršavanje zadataka?
Svi djelatnici odgovorni su voditelju OPG-a dok je voditelj odgovoran prema sebi i kokošima.
Na koji način se izvršavaju zadaci? Opis!

Zadatci se izvršavaju 2 puta dnevno kao što je već spomenuto. U zadatke spada sakupljanje jaja i hranjenje kokoši. Sakupljanje se vrši ručno te je potrebno voditelju prijaviti broj sakupljenih jaja zbog evidencije, također prilikom sakupljanja jaja potrebno je odraditi hranjenje kokoši. Hranjenje se odvija pomoću unaprijed određene posude za davanje hrane te se ovisno o potrebi daje 1 do 2 posude odnosno mjerice. Što se tiče nabave hrane ista se naručuje za nekoliko mjeseci i u to spadaju suhi kruh(kupuje se u pekari), žitarice – kukuruz, pšenica, posije, koncentrat(direktno od proizvođača).

Postoje li problemi i koji su u izvršavanju zadataka?

Jedan od problema koji se učestalo pojavljuje je pitanje koliko je bilo jaja odnosno koliko je pojedini djelatnik sakupio jaja. Uz ovaj problem o količini jaja pojavljuje se i problem je li hrana dana kokošima te da li je napunjena voda. Što se tiče nabave hrane cijena je varijabilna i teško je u papirima pratiti cijenu koja je plaćena po kilogramu hrane te voditelj nema uvid u recimo godišnju potrošnju sredstava.

Identifikacija problema

Pitanja ponoviti više puta!

Na koje probleme korisnik nailazi tijekom posla?

Korisnik se najčešće susreće s problemom o količini sakupljenih jaja kao i s pitanjem da li je hrana dana kokošima budući da nema potrebe prekomjerno ostavljati hranu ako još nije potrebno dodavati. Također za vodu vrijedi kao i za hranu. Vođenje OPG-a zahtjeva sredstva koja nisu neograničena te je potrebno pažljivo njima upravljati kako bi se maksimizirao profit na što direktno utječe cijena po kojoj se nabavlja hrana koja također diktira prodajnu cijenu jaja.

Postoje li standardni načini rješavanja tih problema?

Standardni način za rješavanje problema količine je spremanje jaja u karton no nije moguće efikasno svakodnevno pratiti kolika je količina sakupljena. Što se hrane i vode tiče hrana se daje svaki dan bez kontrole i ikakve optimizacije dok se voda puni po potrebi ovisno o potrošnji i vremenskim uvjetima. Prilikom nabave hrane cijene se zapisuju na papiriće i teško je voditi evidenciju o troškovima koji se događaju jer je trenutni sustav građen ad hoc.

Postoji li bolji način za riješiti problem?

Aplikacijom bi se omogućilo da svaki djelatnik ima mogućnost unosa količine sakupljenih jaja kao i evidencija da li je dana hrana i voda. Na ovaj bi se način riješili i evidentirali svi problemi koji trenutno postoje u sustavu te se dobila osnova za stvaranje analize što bi voditelju dalo bolji uvid u stanje sustava kao i potrošnju materijala, troškove i potencijalni profit od prodaje sakupljenih jaja.

Sumarizacija problema

- Grupirati probleme, opisati ih vlastitim riječima
- Pročitati to korisniku!
- Slaže li se ovo što je napisano s korisnikovim očekivanjima?
- Navesti 3-4 najbitnije stavke za implementaciju

Problemi koji se pojavljuju su kontrola djelatnika i voditelja o izvršenim poslovima. Naravno da je od izrazite važnosti da djelatnici poštuju načela i pravila da će unositi u aplikaciju obavljene poslove u smislu količine jaja koja je sakupljena prilikom obilaska, unos količine (broj mjerica – posuda) hrane i markacije da je data voda kokošima. U početku je moguće naići na poteškoće s obzirom na to da korisnici do sada nisu obavljali ovakve vrste aktivnosti te će im ovo stvoriti dodatni vid poslovnih aktivnosti ali sve u svrhu napredovanja i stvaranja boljih radnih uvjeta.

Voditelju će se svakako olakšati vođenje sustava s obzirom na to da će imati podatke o potrošnji sredstava na hranu kao i dodane izračune kao ostvareni dohodak i sve rashode koji se događaju u sustavu.

Evidencija količine jaja

Evidencija količine hrane

Evidencija o vodi

Izračun troškova nabave hrane

Identificiranje nefunkcionalnih zahtjeva

Koju razinu edukacije ima korisnik?

Korisnik sustava ima nisku do srednju razinu edukacije što znači da se služi računalom i mobitelom ali ne više od jedan sat dnevno.

Koju razinu vještina rada na računalu ima korisnik kojem je sustav namijenjen?

Korisnik sustava ima nižu do srednju razinu vještine rada na računalu.

Koji drugi informatički sustavi se koriste u firmi i na kojim platformama?

U ovom OPG-u se ne koriste drugi informacijski sustavi.

Kako se novi sustav može povezati s postojećim IT sustavima? Postoji li potreba za tim?

S obzirom na to da ne postoje drugi sustavi nema potrebe za povezivanjem ni migracijom podataka.

Postoje li planovi za nadogradnju postojećih sustava ili platformi?

Trenutno se ne planira nadogradnja. Ali buduću nadogradnju je uvijek moguće naknadno dogovoriti.

Koja su očekivanja korisnika od novog sustava?

Očekivanja su da će sustav olakšati vođenje evidencije o hranjenju, sakupljanju kao i vođenje podataka o nastalim troškovima.

Kakvu vrstu dokumentacije korisnik očekuje na kraju?

Jednostavno čitljivo.

U kojoj mjeri bi sustav trebao biti dostupan?

Sustav bi trebao biti dostupan u svakom trenutku s time da je veći naglasak na dostupnost nego na brzinu sustava.

Koja su očekivanja korisnika vezana uz performanse sustava?

Da sustav bude intuitivan i radi fluidno po mogućnosti bez gubitka podataka i rušenja sustava.

Tko će održavati i konfigurirati sustav?

Kreatori sustava su dužni i konfigurirati i održavati sustav po potrebi.

Kako bi se sustav trebao instalirati i konfigurirati?

Instalacija i konfiguracija sustava nije nužno potrebna pošto je ideja sustava da bude dostupna na uređajima koji imaju internetsku vezu kroz web preglednik.

Koji su planovi za backup podataka?

Sustav ima planiranje backupe kroz *firestore* bazu podataka.

Koji su sigurnosni zahtjevi?

Sigurnost će se održavati sustavom autentifikacije. Pošto je ciljana i jednostavnost sustava koristit će se *cache* kako bi spremili podaci korisnika čime će se maknuti potreba za prijavom nakon inicijalne prijave.

Kako će se sustav distribuirati?

Web aplikacija se planira distribuirati koristeći *DigitalOcean* server i *nginx* kao web server.

Postoje li još neke specifičnosti ili zahtjevi o kojima bi trebalo voditi računa?

Ideja dizajna sustava se temelji na *KISS (keep it simple and stupid)* principu.

Intervju korisnika za Kokoshinjac

Obavljen je intervju s korisnikom te su razgovorom specificirani zahtjevi odnosno funkcionalnosti koje su korisniku potrebne u aplikaciji.

Evidencija količine sakupljenih jaja

Korisnici na obiteljskom gospodarstvu svakog dana sakupljaju jaja te imaju problem bilježenja količine sakupljenih jaja odnosno potrebno je svakodnevno komunicirati među djelatnicima kako bi se znao točan broj sakupljenih jaja u tom danu. Aplikacijom će se riješiti taj problem tako da će se

omogućiti svim korisnicima obiteljskog gospodarstva unos količine pokupljenih jaja u tom danu. Moguće je više unosa u jednom danu.

Evidencija hrane

Hranjenje je još jedna od aktivnosti koja se obavlja svakodnevno na obiteljskom gospodarstvu. Hrana se kupuje u većim količinama i za nekoliko mjeseci te se to evidentira u obliku troška. Hrana se koristi svakodnevno i daje se pomoću mjerica zbog lakšeg snalaženja u količinama. Za potrebe unosa u aplikaciju potrebno je izmjeriti (izvagati) mjericu kojom se daje hrana zbog povećanja točnosti evidencije. Korisnik će unositi broj mjerica koje su dane (primjer 1,2,3, 2.5 i sl).

Troškovnik

Na obiteljskom gospodarstvu koristi se više oblika žitarica, suhi kruh i krumpir kao hrana za kokice. Cijene za navedeno variraju te je potrebno omogućiti izmjenu cijene prilikom svake nabave. Neke od žitarica koje se koriste su pšenica – u zrnju i posije, kukuruz – šrot. Cijena kukuruza i pšenice je oko 2 kn po kilogramu.

Pregled podataka

Korisniku je potrebno omogućiti pregled sakupljenih podataka kako bi se mogli šepuriti po selu koliko je jaja proizvedeno kao i pregledati koliko je jaja proizvedeno u kojem dijelu godine. Prikazati minimalnu i maksimalnu količinu jaja proizvedenih po mjesecu kao i potrošnju hrane za pojedini mjesec. Ako je moguće prikazati grafički. Na ovaj će način korisnik dobiti uvid u odnos uložene hrane i dobivenih jaja te tako moći intervenirati u određenim situacijama i tako povećati prinos.

1.1 Specifikacija softverskih zahtjeva za Kokoshinjac

Datum: 6.4.2021

Kratak opis sustava

Projekt nastaje kao potreba za evidencijom, praćenjem i kao potpora u odlučivanju pri izvršavanju aktivnosti vezanih za sakupljanje jaja, hranjenjem kokoši, nabavom hrane te praćenjem podataka vezanih za ove aktivnosti na obiteljskom gospodarstvu.

Za koga (što) je aplikacija napravljena: Aplikacija je namijenjena za korištenje na obiteljskom gospodarstvu.

Akteri

- Voditelj (1)*
- zaposlenik (2)*

Popis funkcionalnosti prema pojedinom akteru

Za aktera 1:

- Pregled podataka(unosa)
- Izračun troškova

Za aktera 2:

- evidencija sakupljenih jaja
- evidencija potrošnje hrane

Scenariji

Akter 1 - voditelj

Funkcionalnost 1: Izračun troškova

Tijek događaja (koraci):

Voditelj nadzire OPG i ima uvid u troškove koji radom OPG-a nastaju. Na temelju unesenog broja jaja kao i količine hrane koja je potrošena, voditelj ima uvid u stanje jaja, stanje hrane te profit koji je OPG ostavario. Voditelj se brine o nabavi hrane.

Funkcionalnost 2: Evidencija sakupljenih jaja(kom)

Tijek događaja (koraci):

Zaposlenik odlazi u kokošinjac i sakuplja jaja. Pri spremanju jaja na za to predviđeno mjesto unosi u aplikaciju broj sakupljenih jaja(kom).

Funkcionalnost 3: Evidencija količine potrošene hrane(kg)

Tijek događaja (koraci):

Zaposlenik odlazi u prostoriju s hranom te uzima određeni broj mjerica hrane, koja je prethodno izvagana i određena, te daje kolicama hranu. Nakon obavljenog hranjenja unosi se u aplikaciju podatak o količini potrošene hrane u kilogramima.

Funkcionalnost 4: Pregled podataka app

Tijek događaja (koraci):

Voditelj ima mogućnost pregleda unesenih podataka za tekući mjesec, prikazani su podatci o sakupljenim jajima i količini hrane koja je taj dan potrošena.

Scenariji za aktera 1:

Rbr.	Naziv funkcionalnosti	Opis	Tijek događaja (koraci)
1	Funkcionalnost 1	Izračun troškova	- vođenje evidencije jaja - vođenje evidencije o količini hrane
2	Funkcionalnost 2	Evidencija sakupljenih jaja	- sakupljanje jaja - unos količine sakupljenih jaja(kom)
3	Funkcionalnost 3	Evidencija količine potrošene hrane	- priprema/hranjenje - unos količine potrošene hrane(kg)
4	Funkcionalnost 4	Pregled podataka app	- ukupno potrošena hrana - ukupno sakupljena jaja -izračun mjesečnog profita na temelju sakupljenih jaja(+) i količine potrošene hrane(-)

Scenariji za aktera 2:

Rbr.	Naziv funkcionalnosti	Opis	Tijek događaja (koraci)
1	Funkcionalnost 1	Evidencija sakupljenih jaja	- sakupljanje jaja - unos količine sakupljenih jaja(kom)
2	Funkcionalnost 2	Evidencija količine potrošene hrane	- priprema/hranjenje - unos količine potrošene hrane(kg)

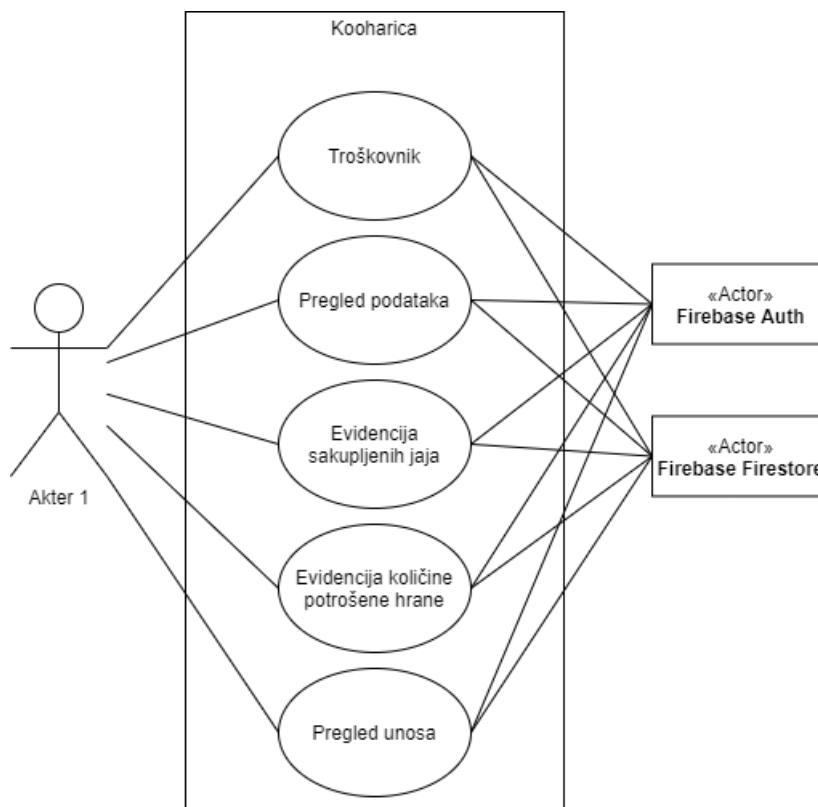
3. Analiza sustava

Kroz analizu sustava obradit će se analiza zahtjeva i korištenja sustava, ali i odabir tehnologija koje su korištene u projektu.

3.1. Analiza zahtjeva i korištenje sustava

Na slici 1 vidljiv je use case dijagram za Aktera1 – Voditelja obiteljskog gospodarstva te na istom se nalazi koje su funkcionalnosti namijenjene tom tipu korisnika.

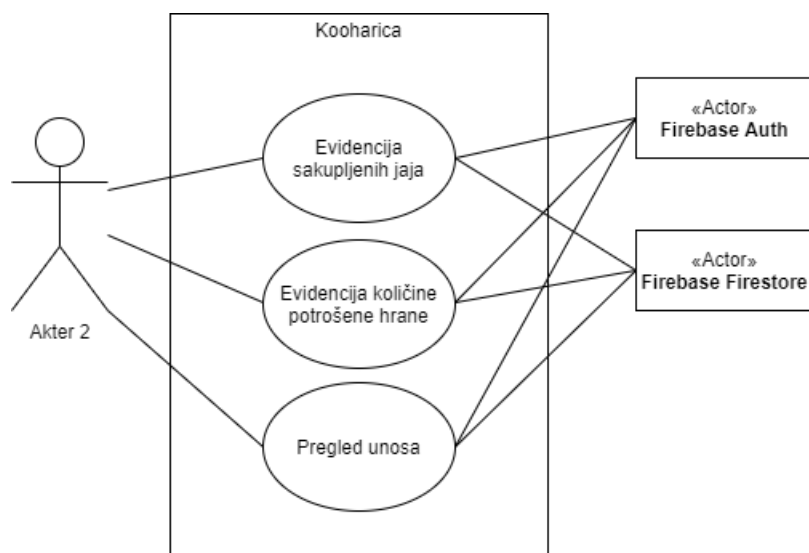
Slika 1: Use case akter1



Izvor: autor

Na slici 2 je vidljiv use case dijagram za Aktera2 – Zaposlenika obiteljskog gospodarstva te na istom se nalazi koje su funkcionalnosti namijenjene tom tipu korisnika.

Slika 2: Use case akter2



Izvor: autor

3.2. Odabir tehnologija

Programsko rješenje izrađeno je za web infrastrukturu. Podatci se pohranjuju u ne relacijsku bazu podataka – firebase. Korisničko front-end sučelje izrađeno je pomoću Vue frameworka te se koristi responzivni dizajn kako bi aplikaciju bilo moguće koristiti na računalu i mobilnim uređajima. Backend za komunikaciju front-enda i baze podataka je node.js. Programsko rješenje ne zahtijeva instalaciju a baza podataka nalazi se u cloudu.

Verzije:

@vue/cli 4.5.11

Node v14.15.2

4. Dizajn sustava

Dizajn sustava je podijeljen na dizajn korisničkih sučelja, dijagram klasa, dijagram objekta, model podataka i sekvencijalni dijagram. Kroz daljnji rad detaljno će se razraditi isti.

4.1. Dizajn korisničkih sučelja

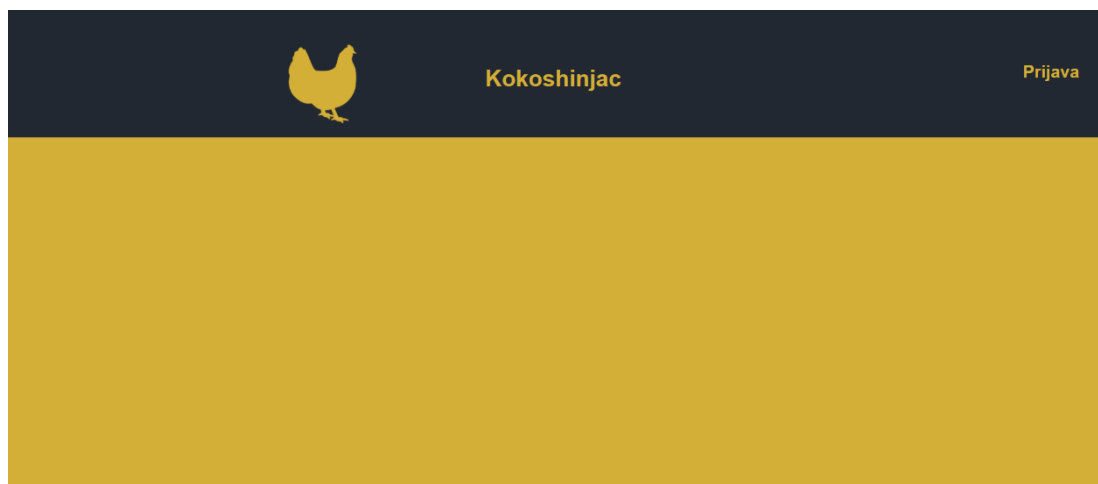
Na sljedećim slikama se nalaze prozori koji predstavljaju dizajn korisničkog sučelja. Slike su prikazane redoslijedom korištenja. Na slici 3 je moguće pristupiti web aplikaciji, ali nije moguće funkcionalnostima aplikacije. Slika 4 sadrži prozor za prijavu i ispravnom prijavom se dobivaju funkcionalnosti.

Slika 5 prikazuje prozor za pregled gdje su u kalendarskom stilu prikazani podaci vezani za prikupljena jaja i potrošenu hranu.

Slika 6 sadrži prozor za unos te služi za unos u bazu podataka.

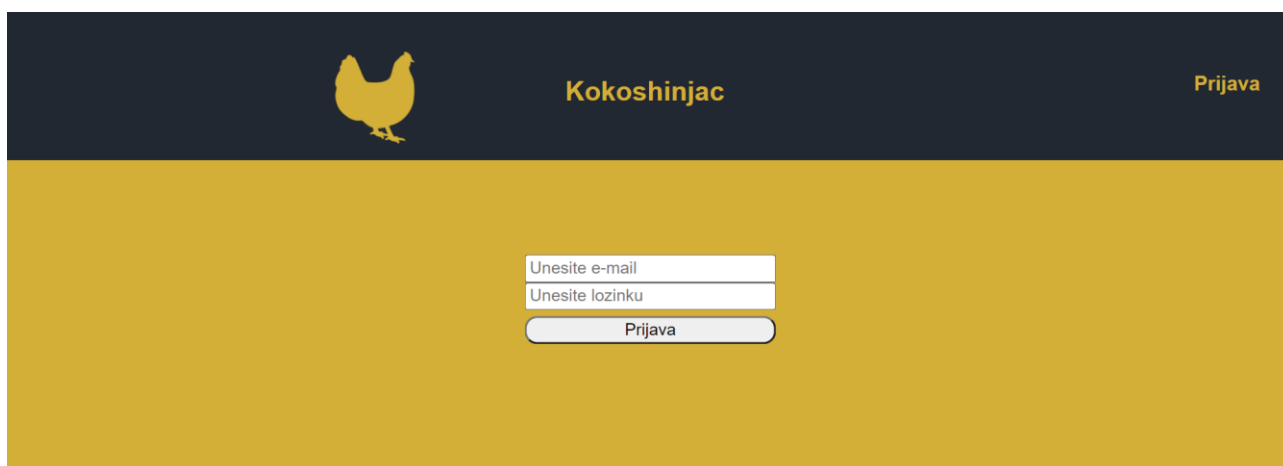
Slika 7 je pregled trenutnih unosa u bazu podataka dok je slika 8 pregled troškovnika gdje je moguće dobiti trenutno stanje profita ovisno o sakupljenim jajima i potrošenoj hrani.

Slika 3: Prozor bez prijave



Izvor: autor

Slika 4: Prozor za prijavu




Izvor: autor

Slika 5: Prozor za pregled

<div> Pregled Unos Pregled unosa Troškovnik </div> <div>  <div>Kokoshinjac</div> <div>voditelj</div> <div>Odjava</div> </div>						
1. Lipanj	2. Lipanj	3. Lipanj	4. Lipanj	5. Lipanj	6. Lipanj	7. Lipanj
Količina jaja: 11	Količina jaja: 10	Količina jaja: 12				Količina jaja: 1
Potrošena hrana (kg): 0.8	Potrošena hrana (kg): 4					
8. Lipanj	9. Lipanj	10. Lipanj	11. Lipanj	12. Lipanj	13. Lipanj	14. Lipanj
Količina jaja: 6	Količina jaja: 9000000026					
Potrošena hrana (kg): 2	Potrošena hrana (kg): 1.3					
15. Lipanj	16. Lipanj	17. Lipanj	18. Lipanj	19. Lipanj	20. Lipanj	21. Lipanj


Izvor: autor

Slika 6: Prozor za unos

<div> Pregled Unos Pregled unosa Troškovnik </div> <div>  <div>Kokoshinjac</div> <div>voditelj</div> <div>Odjava</div> </div>	
<div>Unos</div> <div> <div>Odabir ▼</div> <div>Unesite količinu</div> <div>Spremi</div> </div>	

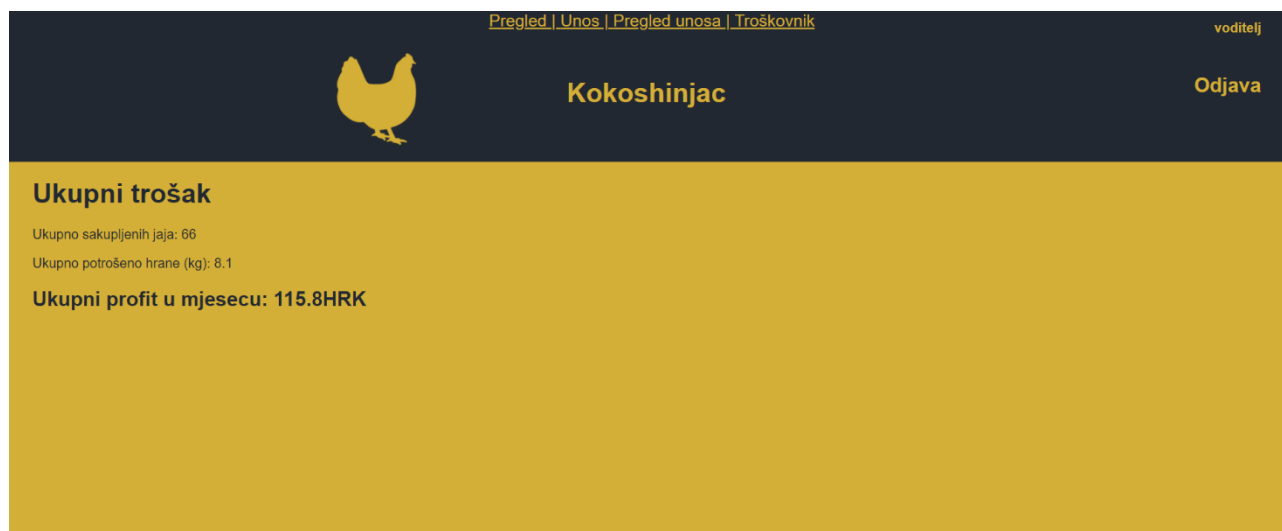
Izvor: autor

Slika 7: Prozor za pregled unosa

<div> Pregled Unos Pregled unosa Troškovnik </div> <div>  <div>Kokoshinjac</div> <div>voditelj</div> <div>Odjava</div> </div>	
<div>Odabir kategorije pregleda</div> <div> <div>Odabir ▼</div> <div> <div>Odabir</div> <div>Jaja</div> <div>Hrana</div> </div> </div>	

Izvor: autor

Slika 8: Prozor za troškovnik

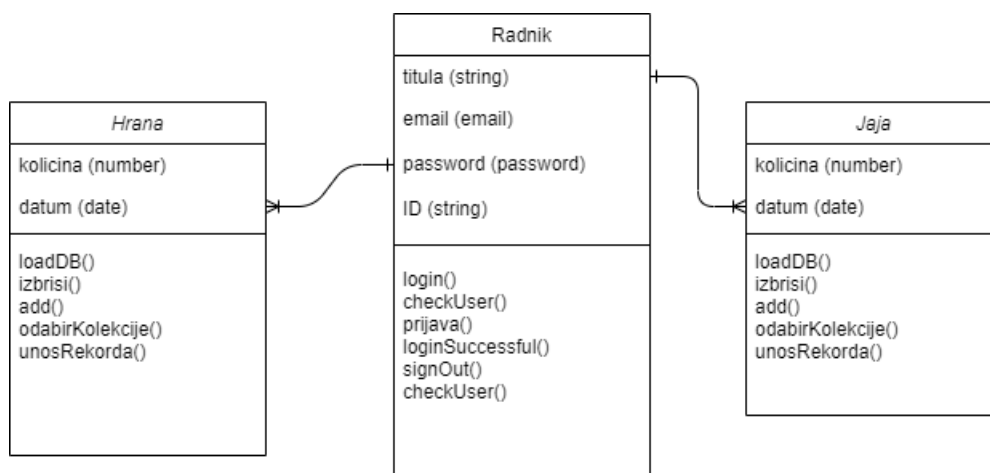


Izvor: autor

4.2. Dijagram klasa

Model podataka koncipiran je tako da radnik ima mogućnost stvoriti unos hrane i jaja u odnosu da jedan ili više unosa pripada samo jednom radniku.

Slika 9: Dijagram klasa

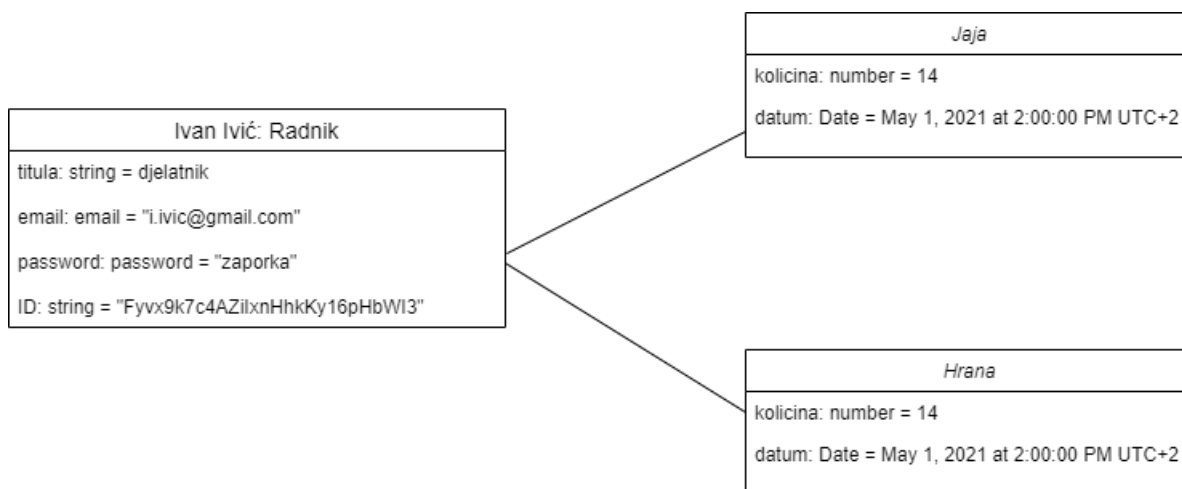


Izvor: autor

4.3. Dijagram objekta

Prikazan je dijagram objekta i primjer podataka koji će se nalaziti u bazi podataka.

Slika 10: Dijagram objekta

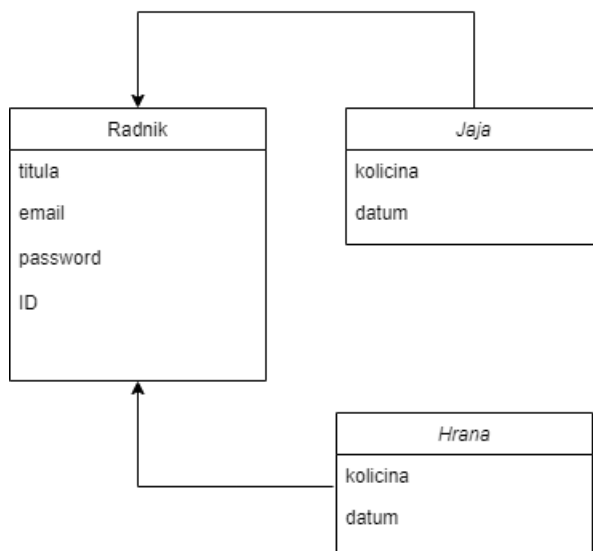


Izvor: autor

4.4. Model podataka

Model podataka na slici 11 je nerelacijski model podataka.

Slika 11: Model podataka

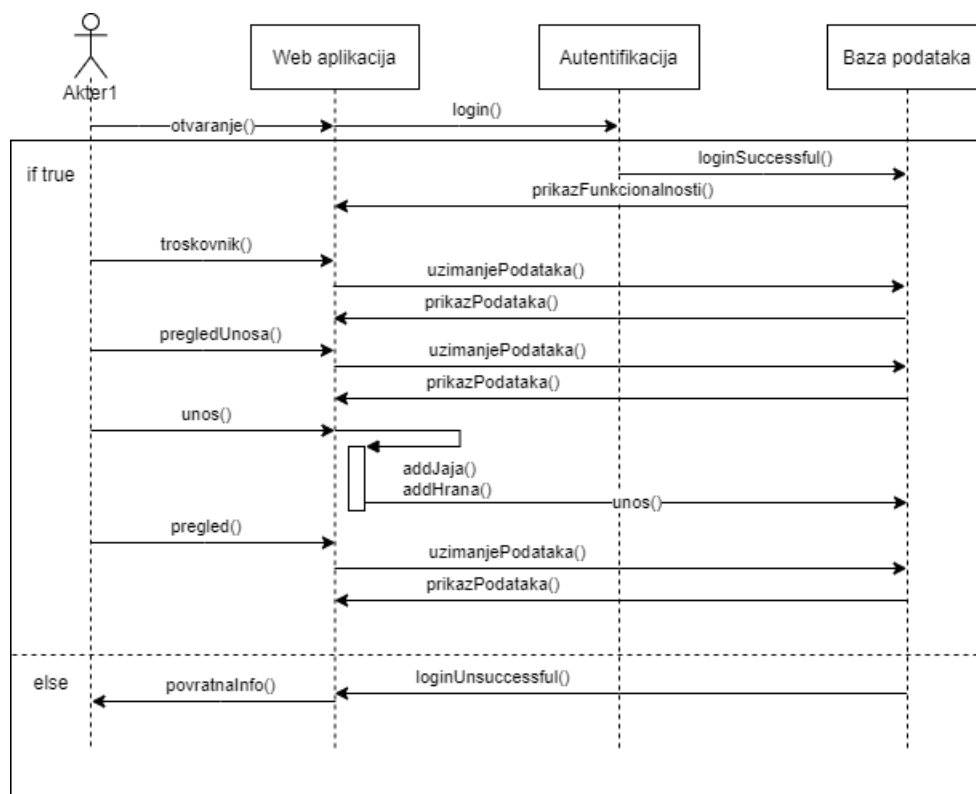


Izvor: autor

4.5. Sekvencijalni dijagram

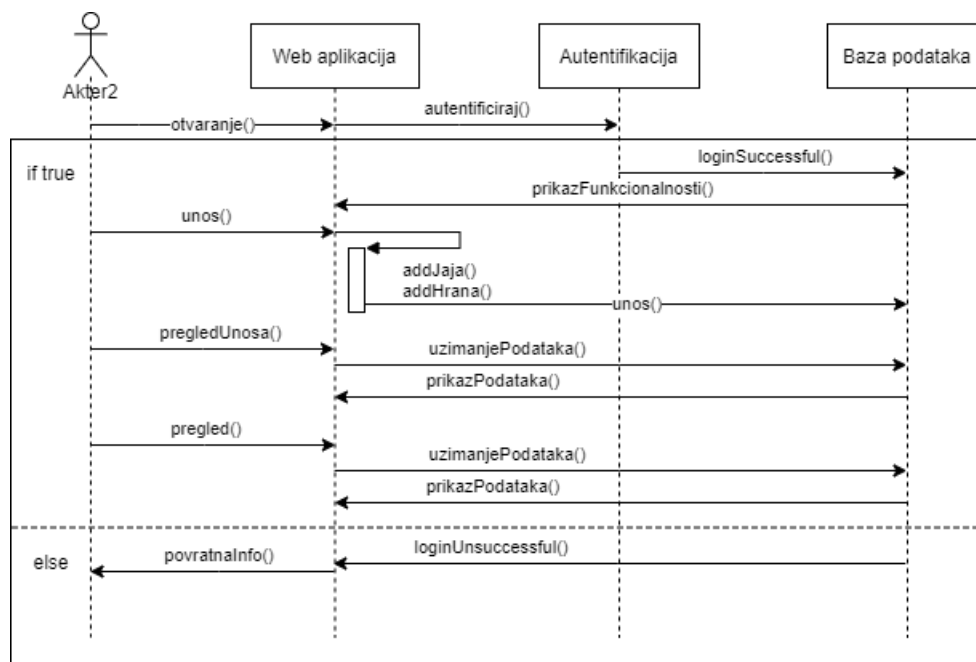
Na slikama 12 i 13 se nalaze sekvencijalni dijagrami i prikazuju vremensku interakciju glavnih objekata s akterima.

Slika 12: Sekvencijalni dijagram akter 1



Izvor: autor

Slika 13: Sekvencijalni dijagram akter2



5. Implementacija sustava

Implementacija sustava se bavi postavljanjem radnog okruženja, opisu verzije aplikacije sa opisom učestalosti rada te prikazom dijelova programskog koda s obrazloženjem važnijih funkcionalnosti.

5.1. Postavljanje radnog okruženja

Za izradu ovog aplikativnog rješenja korišteni su *IDE*: Visual Studio Code (1.56.2) za programiranje, dok je Google Chrome web preglednik korišten za pristupiti web aplikaciji.

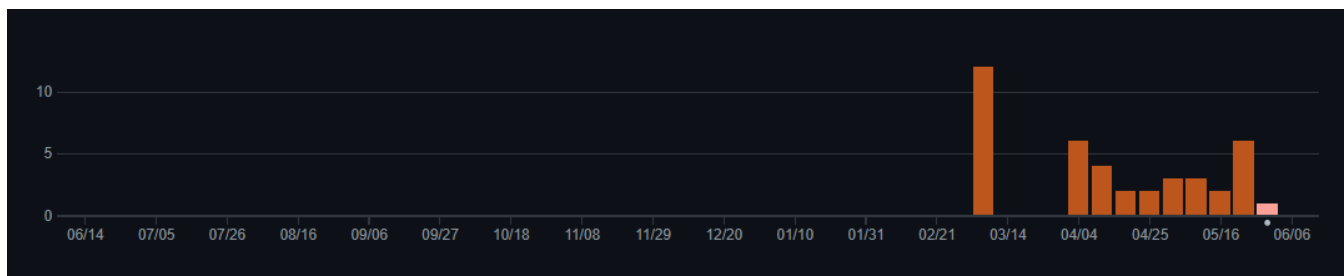
5.2. Verzije aplikacije

<https://github.com/fox1806/Kokoshinjac>

Prvi commit izveden je 9.3.2021 godine.

Iz grafa na slici 14 je moguće uočiti kako je rad kroz dva mjeseca bio konstantan i kako je najviše *commit*-ova napravljeno na početku i na kraju projekta, dok je period između bio obilježen konstantnim radom.

Slika 14: grafički prikaz učestalosti rada



Izvor: autor

5.3. Prikaz dijelova programskog koda

Unos u bazu podataka je rađen na način da se prvo odabere koji tip se želi ubaciti i potom se odradi unos u kolekciju. Primjer je prikazan na slici 15.

Slika 15: Unos u bazu

```
unosRekorda(){
  if (this.odabrano === "jaja" && this.kolicina > 0){
    jaja.add({
      kolicinaJaja: this.kolicina,
      datum: new Date(),
      korisnik: auth.currentUser.uid,
    }).then(()=>{
      alert("Podatak unesen");
      this.odabrano = '';
      this.kolicina = '';
    });
  } else if (this.odabrano === "hrana" && this.kolicina > 0){
    hrana.add({
      kolicinaHrane: this.kolicina,
      datum: new Date(),
      korisnik: auth.currentUser.uid,
    }).then(()=>{
      alert("Podatak unesen");
      this.odabrano = '';
      this.kolicina = '';
    });
  } else {
    alert("Upisana neispravna vrijednost");
  }
},
```

Izvor: autor

Na slici 16 je prikazano čitanje podataka za prikaz u „Pregled“ funkcionalnosti. Prikazano je čitanje kolekcije hrana, ali važno je napomenuti kako se učitava i kolekcija jaja. Nije prikazano na slici pošto je princip čitanja isti.

Crvenim pravokutnikom je označen zanimljiv dio koji koristi *reduce* funkciju kako bi spremilo svu količinu hrane koja je spremljena za jedan datum. Odnosno sprema i zbraja vrijednosti po datumu.

Slika 16: Čitanje podataka

```
hrana.onSnapshot({
  // Listen for document metadata changes
  includeMetadataChanges: true
}, (doc) => {
  this.hranaDb.length = 0;
  // ...
  doc.forEach(data=>{
    this.hranaDb.push(data.data());
  })
  result = Object.values(this.hranaDb.reduce((a, {datum, kolicinaHrane}) => {
    let date = new Date(datum.seconds*1000);
    datum = [date.getDate(), date.getMonth()];
    a[datum] = (a[datum] || {datum, kolicinaHrane: 0});
    a[datum].kolicinaHrane = String(Number(a[datum].kolicinaHrane) + Number(kolicinaHrane));
    return a;
  }, {}));
  this.hranaDb.length = 0;
  this.hranaDb.push.apply(this.hranaDb, result);
});
```

Izvor: autor

Slika 17: Čitanje podataka - pregled unosa

```
startLoadDb(){
  this.dalje=true;
  let query;
  if(this.odabrano=="Jaja"){
    query = jaja.orderBy("datum").limit(this.limit);
  }else if(this.odabrano=="Hrana"){
    query = hrana.orderBy("datum").limit(this.limit);
  }else alert("Odaberite kategoriju")

  this.loadDb(query)
  this.stranica=1;
},
loadDb(query){
  let user = auth.currentUser.uid;
  let grupa;
  users.onSnapshot((doc) => {
    doc.forEach((data)->{
      if(user===data.data().UID) grupa = data.data().grupa;
    })
    // ako je admin ima pristup svim podacima
    if(grupa=="voditelj") {
      query.onSnapshot({
        // listen for document metadata changes
        includeMetadataChanges: true
      }, (doc) => {
        this.loadedDb.length = 0;
        // ...
        this.last = doc.docs[doc.docs.length-1]
        this.first = doc.docs[0]

        let counter = 0;

        doc.forEach((data)->{
          counter++;
          this.loadedDb.push([
            data.id,
            data.data(),
          ])
        })
        if(counter!=this.limit) this.dalje=false;
        // console.log(counter);
        this.formatDate()
      });
    }
    // inace je korisnik i ima pristup samo svojim podacima
  }else {
    query.where("korisnik","==",user).onSnapshot({
      // listen for document metadata changes
      includeMetadataChanges: true
    }, (doc) => {
      this.loadedDb.length = 0;
      // ...
      this.last = doc.docs[doc.docs.length-1]
      this.first = doc.docs[0]

      let counter = 0;

      doc.forEach((data)->{
        counter++;
        this.loadedDb.push([
          data.id,
          data.data(),
        ])
      })
      //
      if(counter!=this.limit) this.dalje=false;

      this.formatDate()
    });
  }
});
},
You, 13 days ago • Dodan pregled unosa
```

Izvor: autor

Prilikom čitanja podataka iz baze prvo se poziva funkcija *startLoadDb()* koja ovisno o odabranom izboru definira varijablu koja će vršiti upit nad kolekcijom. Pošto se u jednom vremenu može učítavati samo jedna kolekcija, odlučeno je da će se definirati *query* varijabla uz koju će se moći lakše i univerzalnije raditi upiti na bazi podataka.

Nakon kreiranja *query* varijable poziva se funkcija *loadDb()* koja učítava bazu. Ako je korisnik grupe „voditelj“ učítat će se svi podaci iz baze, a ako se radi o nekoj drugoj grupi korisnika na *query* će se dodati još funkcija koja definira pravilo za čitanje podataka. U ovom slučaju podaci će se čitati samo ako je korisnik (UID korisnika) jednak *user* (UID trenutno prijavljenog korisnika).

Nakon toga se podaci iz baze podataka spremaju lokalno za prikaz u web aplikaciji.

Programski kod opisanih funkcija je dostupan na slici 17.

6. Problemi u kreiranju aplikacije

Jedan od najvećih poteškoća u radu na projektu se pojavila kod implementiranja prijave u projekt. Zamislili smo projekt u kojem prilikom prijave se pošalje Boolean podatak s potvrdom da se obavila prijava kako bi se moglo definirati koja vrsta korisnika je prijavljena u sustav.

Zbog komponentalnog razvoja u *Vue*-u nije se mogla napraviti lokalna varijabla u komponenti za prijavu, već je bila potrebna globalna varijabla koja će definirati grupu.

Pošto je ovo prvo susretanje autora s konceptima *router*-a i *router-link*-ova krivo je koncipiran rad programa i kroz više sati istraživanja nije se moglo otkriti zašto se podaci ne uspijevaju slati na *parent* komponentu. Pošto se sintaktički sve uredno posložilo bilo je nemoguće da je krivo isprogramirano. Tako da je autor kontaktirao kolegu Ivana Miljančića koji je nakon nekog vremena provjeravanja koda uspio otkriti u čemu je poteškoća.

Poteškoća je dolazila kod toga što se podaci nisu smjeli slati u *parent* kroz *router-link*, već kroz *router-view*. Rješenje je zapravo u tome što je *router-view* element u kojem se sve komponente učitavaju, stoga je i *router-view* parent koji treba prihvatiti podatke od *child view* elemenata.

Stoga, kako je i spomenuto, pismeno se ostavlja zahvala Ivanu za pomoć.

7. Isporuka i korištenje aplikacije

Kroz isporuku i korištenje aplikacije opisati će se pakiranje i isporuka aplikacije, odnosno što je potrebno za isporučiti aplikaciju te korisničke upute gdje je objašnjeno kako korisnik može koristiti navedenu aplikaciju.

7.1. Pakiranje i isporuka aplikacije

Aplikacija je postavljena na web server te je dostupna svim korisnicima s web preglednikom i s podacima za pristup aplikaciji (e-mail i lozinka).

Web aplikacija je dostupna na sljedećoj poveznici: <https://kokoshinjac.huxian.me/#/>

Moguće je i isporučiti aplikaciju lokalno, te se koraci za ostvariti isto nalaze na repozitoriju projekta:

<https://github.com/fox1806/Kokoshinjac>

7.2. Korisničke upute za korištenje aplikacije

Korisnik započinje s korištenjem aplikacije prijavom u istu. Unose se email i lozinka te Firebase provjerava jesu li podatci ispravni. Ovisno o mailu koji je unesen korisniku se na navigacijskoj traci otkrivaju funkcionalnosti.

Odabirom opcije unos zaposlenici unose vrijednosti o količini hrane i količini sakupljenih jaja. Vrijednosti se spremaju u Firestore te su dostupni na pregled kroz funkcionalnost Pregled unosa. Voditelju su na pregledu unosa dostupni svi unosi u bazu.

Voditelj ima prikaz cijelog mjeseca u stilu kalendara u funkcionalnosti Pregled i po datumima je prikazano koliko je sakupljeno jaja i potrošeno hrane.

Voditelj također odabirom funkcionalnosti Troškovnik ima uvid u prihode i rashode koji se automatski računaju te prikazuju kroz izračun profita za tekući mjesec.

Zaključak

Ovu aplikaciju se započelo raditi zbog potrebe obiteljskog gospodarstva gdje su se podaci najčešće vodili u excel tablicama ili ručno na papiru što ne omogućuje korisniku jednostavan pregled i kontrolu nad podacima. Kroz izradu aplikacije, uz veći rast funkcionalnosti je zapravo porasla želja za nastavak rada na istoj i oba autora smatraju kako postoji potencijal za daljnji razvitak. Svi zahtjevi koji su postavljeni na početku projekta su izvršeni ili su potrebne sitne korekcije kako bi doživljaj korištenja bio potpun. Prilikom izrade same aplikacije najviše je vremena utrošeno na izradu komponenata koji čine srž aplikacije.

Literatura i izvori

<https://v3.vuejs.org/guide/introduction.html> (10.06.2021.)

<https://stackoverflow.com/questions/52938939/firebase-rules-write-for-auth-user-but-delete-only-its-own-data> (10.06.2021.)

<https://firebase.google.com/docs/build?authuser=0> (10.06.2021.)

<https://github.com/vuejs/awesome-vue> (10.06.2021.)

Popis slika

Slika 1: Use case akter1	9
Slika 2: Use case akter2	10
Slika 3: Prozor bez prijave	11
Slika 4: Prozor za prijavu	11
Slika 5: Prozor za pregled	12
Slika 6: Prozor za unos	12
Slika 7: Prozor za pregled unosa	12
Slika 8: Prozor za troškovnik	13
Slika 9: Dijagram klasa	13
Slika 10: Dijagram objekta	14
Slika 11: Model podataka	14
Slika 12: Sekvencijalni dijagram akter 1	15
Slika 13: Sekvencijalni dijagram akter2	15
Slika 14: grafički prikaz učestalosti rada	16
Slika 15: Unos u bazu	17
Slika 16: Čitanje podataka	17
Slika 17: Čitanje podataka - pregled unosa	18