APLIKACIJA ZA ANALIZU POTROŠNJE NA X-ICI

Živko, Dominik

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:842222

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-24**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

APLIKACIJA ZA ANALIZU POTROŠNJE NA X-ICI

Završni rad

Dominik Živko

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek,01.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Dominik Živko
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4302, 26.07.2018.
OIB studenta:	12350081631
Mentor:	Izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Aplikacija za analizu potrošnje na X-ici
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	01.09.2021.
Datum potvrde ocjene Odbora:	08.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTIRADA

Osijek, 22.09.2021.

Ime i prezime studenta:	Dominik Živko
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4302, 26.07.2018.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za analizu potrošnje na X-ici**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD **1**
1.1. Zadatak završnog rada 2
2. TEORIJSKE OSNOVE I KORIŠTENE TEHNOLOGIJE **3**
2.1. Kotlin 4
2.2. JavaFX 5
3. TEHNIČKA IZVEDBA **6**
3.1. Terminologija 6
3.2. Struktura aplikacije 6
3.2.1. Raspored klasa 6
3.2.2. Životni ciklus 9
3.2.3. Lokalna pohrana 11
3.3. Podatkovne strukture 12
3.4. Autentikacija i mrežna komunikacija 14
3.4.1. Metode prikupljanja podataka 14
3.4.2. Prijava 16
3.4.3. Sigurnost 17
3.5. Obrada i prikaz podataka 17
4. PRIKAZ RADA APLIKACIJE **19**
4.1. Instalacija 19
4.2. Opis sučelja 20
4.2.1. Pregled 22
4.2.2. Prikaz računa 23
4.2.3. Kalendar 24
4.2.4. Opća statistika 25
4.2.5. Statistika artikala 26
4.2.6. Upravljačka traka 28
5. ZAKLJUČAK **29**
LITERATURA **30**
SAŽETAK **31**
ABSTRACT **32**
PRILOZI **33**

1. UVOD

Za vrijeme visokog školovanja, jedna od pogodnosti koju redoviti studenti imaju na raspolaganju je subvencionirana prehrana. Uz državnu potporu, studenti na određenim mjestima mogu jesti po vrlo povoljnim cijenama, sve dok uz sebe imaju svoju studentsku iskaznicu, poznatu kao X-icu. Očitavanjem kartice, studentima se može odobriti potpora pri kupnji određene količine prehrambenih proizvoda. Kako je cijeli sustav digitalan, podatke o kupljenim proizvodima je lako pratiti i učiniti dostupnima preko interneta.

U postojećem sustavu koji otkriva te podatke nije stavljen fokus na sažeti prikaz podataka koji bi zainteresiranim korisnicima pružio uvid u informacije korisne za osobnu evidenciju potrošnje ili prehrane. To je osnovni problem kojeg ovaj rad nastoji riješiti. Aplikacija dokumentirana stranicama ovog rada pretvara izvorni prikaz računa u sažeti grafički prikaz raznih statističkih podataka o računima prikupljenim kroz vrijeme korištenja.

Odabrani naziv aplikacije je *UniStat*, a ime autora aplikacije korišteno u svrhe organizacije podatkovne strukture je *Altline*. Ova imena se pojavljuju na nekolicini mjesta u proizvodnom okruženju aplikacije.

U prvom poglavlju nakon uvoda razmatra se trenutna razvijenost područja teme ovog rada i postoje li slična javno dostupna rješenja. Također se opisuju glavne tehnologije korištene pri izradi aplikacije.

Drugo poglavlje glavnog dijela rada, tj. poglavlje broj 3, bavi se tehničkom izvedbom projekta. Definirana je osnovna terminologija korištena u ostatku rada te je detaljno opisana struktura izrađene aplikacije. Opisane su metode kojima su postignute sve važne mogućnosti aplikacije kao i razlozi koji su doveli do korištenja tih rješenja.

Posljednje poglavlje u glavnom dijelu rada demonstrira postupak instalacije i krajnje uporabe aplikacije. U njemu je prikazan i detaljno opisan izgled korisničkog sučelja aplikacije uz rezultate analize korisničkih podataka koji služe kao primjer dostupan za preuzimanje uz samu aplikaciju.

1.1. Zadatak završnog rada

Zadatak ovog rada je napraviti JavaFX aplikaciju koja prikazuje stanje potrošnje studenata preko subvencionirane kartice (tzv. X-ice). Podatke o potrošnji potrebno je dobaviti iz baze podataka SRCE-a, odnosno kroz internetske stranice koje pružaju uvid u račune prijavljenog studenta. Potrebno je tablično ili grafički prikazati stanje potrošnje po danima, tjednima ili mjesecima. Sve podatke o potrošnji treba pohraniti u lokalnu bazu podataka.

2. TEORIJSKE OSNOVE I KORIŠTENE TEHNOLOGIJE

Neslužbena javno dostupna rješenja na temu uvida u studentsku prehranu i potrošnju nisu dostupna u vrijeme nastanka ovog rada. Jedina javno dostupna točka uvida u takve podatke je službeni sustav ISSP (Informacijski Sustav Studentskih Prava) kojeg održava SRCE (Sveučilišni Računski Centar) Sveučilišta u Zagrebu. Prema [1], „Namjena sustava je da omogući korištenje različitih iznosa subvencije za prehranu u studentskim restoranima sukladno razinama prava propisanim od strane Ministarstva te precizno evidentira potrošnju studenata.”. Taj sustav pruža najosnovnije podatke o potrošnji, odnosno iznose i sadržaje pojedinih računa. Ovaj rad nastoji unaprijediti prikaz potrošnje na osnovu tih informacija.

Ciljane platforme ove aplikacije su osobna računala, osobito ona s operacijskim sustavom Windows 10 za kojeg je prvobitno dizajnirana i na kojem je testirana. U teoriji, program podržava pokretanje na bilo kojem popularnijem operacijskom sustavu, ali ispravnost rada svih značajki aplikacije na drugim sustavima u ovom slučaju nije testirana. Ovakvo pokretanje istog programskog paketa na različitim sustavima omogućeno je korištenjem Java platforme[1]. Instrukcije Java programskog jezika se nakon prevođenja izvršavaju na virtualnom stroju prilagođenom sustavu na kojem je instaliran. Prednost ovog pristupa je uklanjanje potrebe za pisanjem različitih verzija aplikacije za različite operacijske sustave.

Razvojno okruženje korišteno pri izradi aplikacije je IntelliJ IDEA[2]. Ergonomski dizajn ovog okruženja omogućuje brz i jednostavan rad na različitim projektima u Java ekosustavu. Ovo okruženje izravno podržava Kotlin programski jezik i ima brojne alate koji korisnika navode na iskorištavanje punog potencijala tog jezika. Također, podrška za VCS (*Version Control Systems*), odnosno sustave za upravljanje verzijama projektnog koda, povećava produktivnost i pomaže pri stabilnom procesu izrade aplikacija. VCS korišten za ovaj projekt je Git s repozitorijem na usluzi GitLab. Točna lokacija repozitorija izvornog koda može se pronaći među prilozima ovog rada.

U ovom radu korištena je Java 11 kao osnovno programsko sučelje. Na sloju iznad toga korišten je Kotlin programski jezik, a okvir za izgradnju samih funkcionalnosti aplikacije je JavaFX. Pojedinosti ovih tehnologija opisane su u sljedećim potpoglavljima.

[1] Više informacija o Java platformi: https://www.oracle.com/java/
[2] IntelliJ IDEA: https://www.jetbrains.com/idea/

2.1. Kotlin

Kotlin[3] je programski jezik s fokusom na sažetost koda. Inačica koja je korištena u ovoj aplikaciji prevodi svoj kod u potpunosti u Java kod, odnosno kod koji se izvršava na JVM (*Java Virtual Machine*). Ovo efektivno znači da Kotlin služi kao alat koji pojednostavljuje pisanje Java aplikacija.

Prema [2, 3], neke od važnijih korištenih značajki u odnosu na Javu uključuju:

- **Pojednostavljeni zapis lambda funkcija** što pridonosi sažetosti koda i promiče principe funkcijskog programiranja.
- **Sigurnost od nepostojećih vrijednosti** (eng. *null safety*). Svaka varijabla ima definiranu poništivost (eng. *nullability*) što povećava svijest programera i može značajno smanjiti učestalost pojave grešaka vezanih uz nepredviđene nepostojeće vrijednosti varijabli.
- **Zaključivanje tipa** (eng. *type inference*). Na osnovu dostupnih informacija iz konteksta, tipovi varijabli u kodu se automatski zaključuju. Rezultat toga je čitljiviji kod bez redundantnih deklaracija tipova svake pojedine varijable. Uz zaključivanje tipa, Kotlin prevoditelj zaključuje i druge pojedinosti koje je moguće znati iz konteksta.
- **Funkcije proširenja** (eng. *extension functions*). One omogućuju jednostavno proširenje programskih sučelja klasa tako da se ona mogu koristiti na isti način kao i ostatak sučelja.
- **Posebne klase** poput podatkovnih klasa (eng. *data class*) znatno sažimaju verziju istog koncepta koju bi bilo potrebno napisati u izvornoj Javi.

[3] Kotlin: https://kotlinlang.org/

2.2. JavaFX

Kako bi izrada funkcionalne i uredne aplikacije bila brža i jednostavnija, poželjno je koristiti valjanu platformu. Tu ulogu ovdje ima JavaFX[4]. Ukratko, to je platforma za izradu desktop aplikacija i aplikacija koje se mogu izvršavati na različitim sustavima, kao i na mobilnim uređajima.

Vidljive iz [4], neke od korištenih značajki su:

- Opcija izrade grafičkog sučelja u *markup* jeziku (FXML) uz programski alat za vizualni dizajn (Scene Builder[5]).
- Stiliziranje grafičkog sučelja po uzoru na CSS.
- Mogućnost povezivanja podataka s prikazom (*data binding*) uz jasnu MVC (*model-view-controller*) strukturu.
- Sučelje za izradu grafikona i ilustracija za statističke analize suvremenog izgleda.

[4] JavaFX: https://openjfx.io/
[5] Gluon Scene Builder: https://gluonhq.com/products/scene-builder/

3. TEHNIČKA IZVEDBA

3.1. Terminologija

Središnji pojam korišten kroz čitav projekt je pojam računa. To je glavna podatkovna jedinica koja čini osnovu za svu analizu koja se provodi u programu. Korišten pojam za račun unutar izvornog koda aplikacije je *bill*. Značenje te riječi nije u potpunosti jednako evidencijskom računu, ali je dovoljno slično da zamijeni druge, nezgrapnije opcije poput *receipt* ili *invoice*.

Uz svaki račun vežu se pojmovi vrijednosti, subvencije i troška (eng. *value*, *subsidy*, *cost*). Pojam vrijednosti računa predstavlja ukupan novčani iznos kojeg restoran naplaćuje. To je iznos prije primjene subvencije koja predstavlja potporu studentu pri otplati računa. Preostali iznos predstavlja studentov osobni trošak.

Unutar projekta, internetska lokacija s koje se preuzimaju podaci ima jednoznačni naziv *webserver*. Uzeto je da značenje ove riječi predstavlja apstraktni entitet koji nudi uslugu prijave i odjave korisnika te ponudu njegovih podataka.

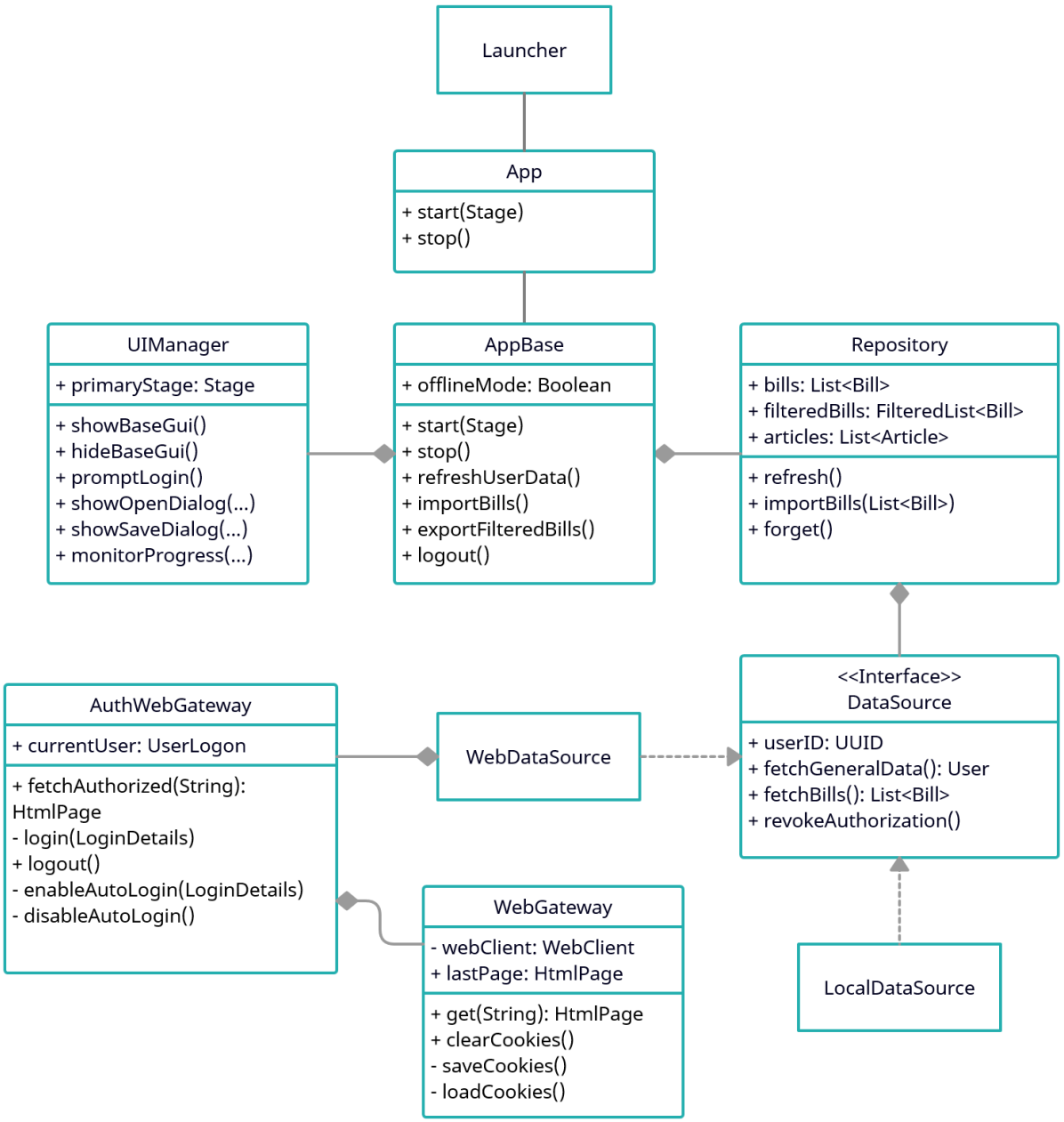
3.2. Struktura aplikacije

Za definiranje strukture aplikacije korišten je *Koin*[6] programski okvir za ubrizgavanje ovisnosti (eng. *dependency injection framework*). Ovime je omogućeno jasno određivanje strukture između klasa bez vođenja brige o instanciranju i omogućavanju pristupa pojedinim komponentama sustava.

3.2.1. Raspored klasa

Pojednostavljena struktura glavnog uređenja aplikacije prikazana je dijagramom 3.1. Na njemu su prikazane osnovne klase koje čine program. Svaka JavaFX aplikacija mora sadržavati jedan objekt koji predstavlja instancu aplikacije. On čini polaznu točku pokretanja i zaustavljanja programa. Ovdje tu ulogu ima objekt *App*. Ubrzo nakon pokretanja, kontrola se prebacuje na *AppBase* instancu koja predstavlja podlogu za čitavu aplikaciju. Unutar tog tipa nalaze se okidači za pokretanje glavnih operacija poput osvježavanja korisničkih podataka te izvoza i uvoza računa.

[6] Koin: https://insert-koin.io/



Dijagram 3.1. *Osnovna struktura aplikacije*

UIManager je zadužen za upravljanje operacijama vezanim za korisničko sučelje. Na ovog upravitelja prebačena je briga o stvaranju i prikazu glavnog prozora kao i raznih obrazaca.

Sljedeća važna komponenta aplikacije je spremište podataka predstavljeno klasom *Repository*. Repozitorij čini točku pristupa za sve korisničke podatke koji se preuzimaju ili učitavaju i na kraju prikazuju na sučelju. Ova klasa brine o osvježavanju i trajnoj pohrani tih podataka. Kako bi se repozitorijem mogao omogućiti pristup podacima ostatku aplikacije, potrebno je imati definiran izvor podataka. Ovu apstrahiranu ulogu ima sučelje *DataSource*. Podaci se mogu

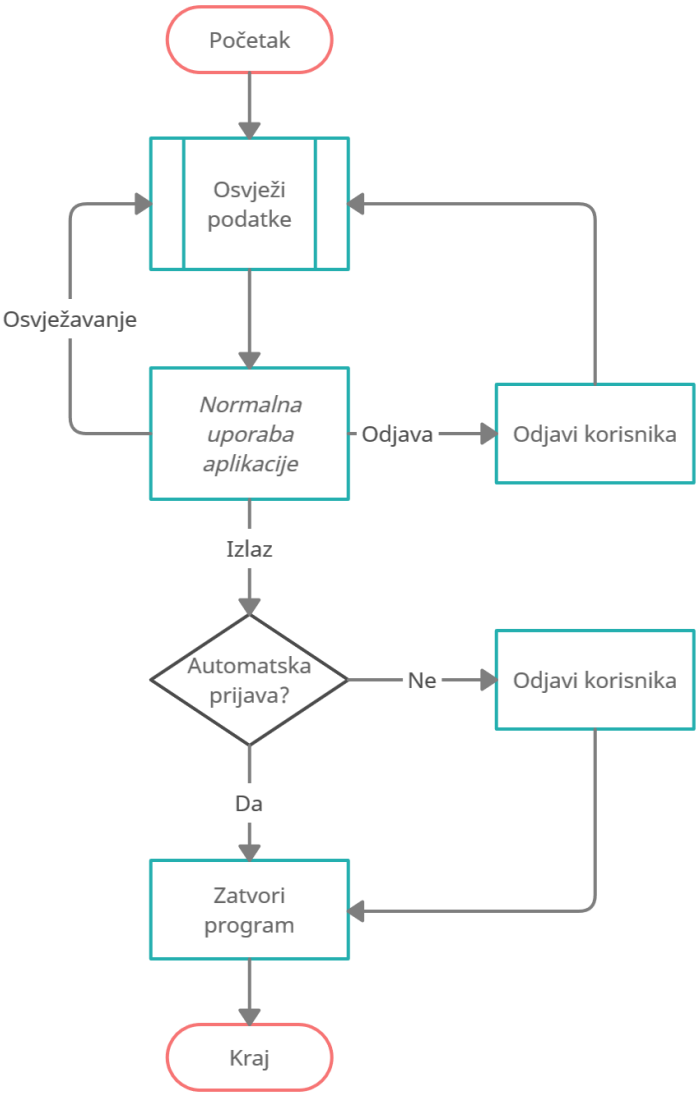
dobaviti na različite načine, odnosno iz različitih izvora, pa tako postoje dvije implementacije spomenutog sučelja:

- *WebDataSource*
- *LocalDataSource*

Prva implementacija podatke dohvaća sa stranica SRCE-a, dok druga iz prethodno spremljene datoteke učitava listu računa zamišljenog korisnika u svrhu demonstracije rada aplikacije bez potrebe za prijavom na stvarni korisnički račun.

WebGateway je osnovna pristupna točka informacijama na internetu. Ova klasa omogućuje pristup raznim internetskim sadržajima u obliku HTML dokumenata i brine o pohrani internetskih kolačića. *AuthWebGateway* koristi ovu uslugu i prilagođava ju isključivo za komunikaciju s uslugom uvida u račune. *Auth* dio imena označava da je osiguran autorizirani pristup podacima, tj. da se postupak prijave automatski izvršava ako poslužitelj to zatraži.

3.2.2. Životni ciklus

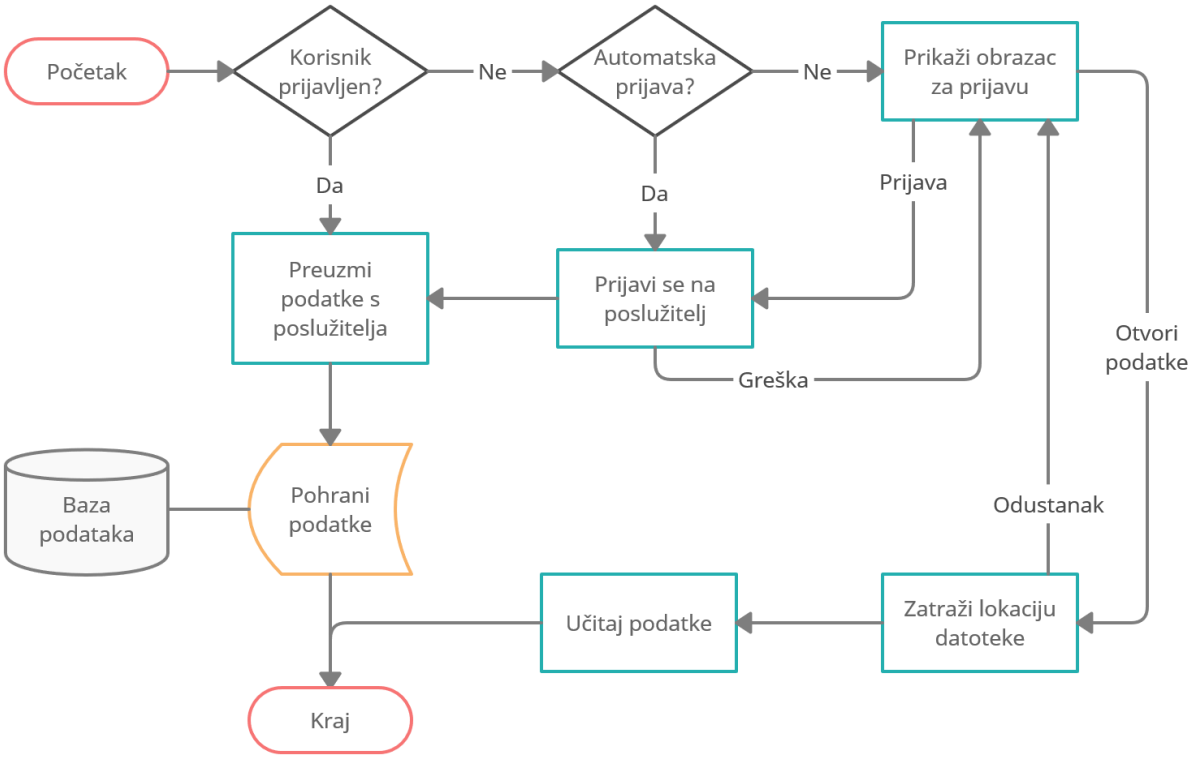


Dijagram 3.2. *Životni ciklus aplikacije*

Životni ciklus aplikacije prikazan je dijagramom 3.2. Prva operacija koja se izvršava po pokretanju aplikacije jest osvježavanje podataka. U sklopu te operacije je osiguravanje valjanog izvora podataka u slučaju da on nije prisutan. To znači da ako nema trenutno prijavljenog korisnika, sustav će automatski zatražiti prijavu ili otvaranje podatkovnog paketa kako bi se podaci mogli osvježiti. Završetkom operacije osvježavanja prikazuje se glavni prozor i omogućena je normalna uporaba aplikacije, tj. prikazuju se podaci, a korisnik ima mogućnost pozivanja drugih operacija koje nastavljaju ciklus. Korisnik može zatražiti odjavu, što će ga odjaviti s poslužitelja i same aplikacije, a zatim će se ponovno pokrenuti operacija osvježavanja podataka, prikazujući novi obrazac za prijavu. Postupak zatvaranja aplikacije razlikuje se s

obzirom na to je li uključena opcija automatske prijave. Ako je opcija uključena, odjava korisnika s poslužitelja se preskače.

Detaljan tijek izvođenja operacije osvježavanja podataka prikazan je dijagramom 3.3.



Dijagram 3.3. *Tijek izvođenja operacije osvježavanja korisničkih podataka*

Postupak osvježavanja podataka zahtijeva prijavljenog korisnika ili odabranu podatkovnu datoteku. Ako je korisnik već prijavljen, podaci će se odmah preuzeti s poslužitelja i pohraniti u lokalnu bazu podataka. Ako to nije slučaj, provjerava se postoje li uvjeti za automatsku prijavu te se ona provodi ako uvjeti postoje, a u suprotnom se prikazuje obrazac za prijavu. Korisnik se kroz taj obrazac može prijaviti, ili može odabrati opciju otvaranja lokalno spremljenih podataka. Tada se oni odmah učitavaju bez potrebe za autentikacijom. Osvježavanje podataka nakon što su otvoreni lokalni podaci nije potrebno te je korisniku onemogućeno.

3.2.3. Lokalna pohrana

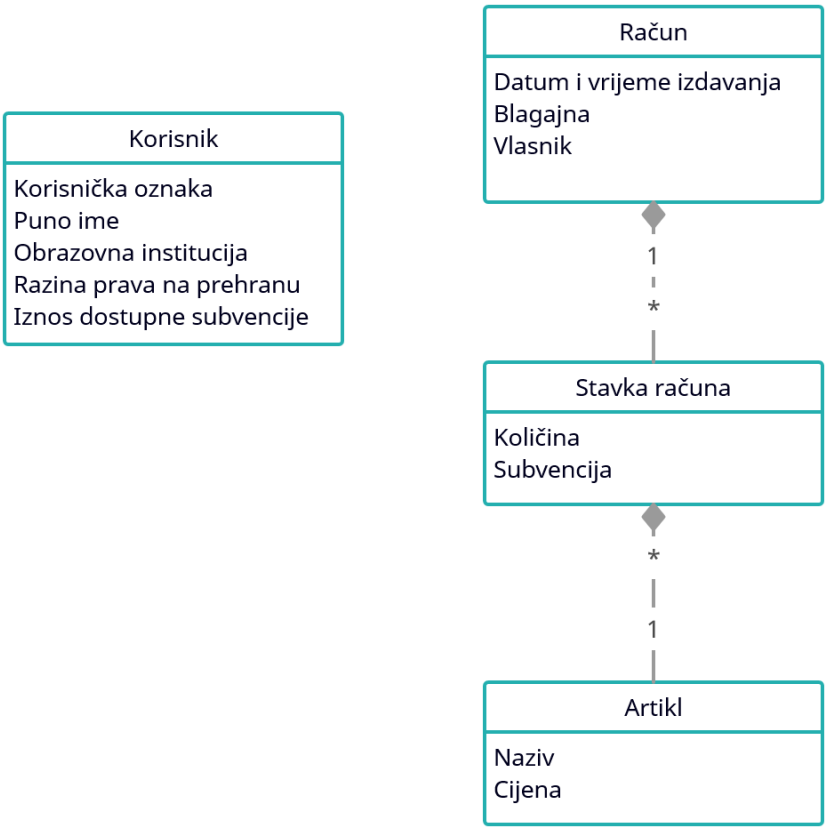
Programski podaci, tj. podaci vezani uz unutarnji rad aplikacije, kao i oni korisnikovi, pohranjuju se u posebnu mapu na korisnikovom sustavu. Ta mapa se nalazi na predviđenom mjestu pohrane aplikacijskih podataka ove vrste. Na Windows operacijskom sustavu, smještaj podatkovne mape je na lokaciji na koju pokazuje sustavna varijabla *%APPDATA%*. Izvorno, to se nalazi na lokaciji "C:\Users*Korisnik*\AppData\Roaming\" gdje *Korisnik* označava ime odgovarajućeg korisničkog računa na sustavu. Unutar te mape stvara se mapa *Altline* koja predstavlja naziv autora, a u nju se postavlja mapa *UniStat* kao konačni smještaj za aplikacijske podatke koje čine baza podataka i dnevnici izvršavanja programa (eng. *logs*).

Praćenje operacija i životnog tijeka aplikacije vođenjem dnevnika izvršavanja postignuto je korištenjem *Log4j*[7] biblioteke. Ona omogućava detaljan zapis *log* poruka o događajima u programu poput izvršavanja ključnih operacija i hvatanja grešaka. Zapis grešaka programerima može dati ključne informacije o postojećim nedostacima programskog koda i znatno ubrzati proces ispravka. U sklopu biblioteke je mogućnost zapisa poruka u datoteku na korisnikovom računalu za slučaj da ju želi samostalno proučiti ili podijeliti s autorom aplikacije ako se dogodila nekakva greška.

[7] Log4j 2: https://logging.apache.org/log4j/2.x/

3.3. Podatkovne strukture

Internetska usluga s koje se preuzimaju korisnikovi podaci pruža brojne informacije o korisniku. U nastavku su navedeni svi podaci koje preuzima ova aplikacija.



Dijagram 3.4. *Model korisničkih podataka*

Dijagram 3.4 prikazuje model korisničkih podataka koji se koriste u ovom projektu. Podaci o korisniku uključuju samo njegove osnovne informacije i one koje su vezane uz sustav subvencionirane prehrane. Podaci o računima uključuju sve relevantne karakteristike računa poput mjesta i vremena izdavanja, te količine, cijene i subvencije svakog kupljenog artikla. Iz tih podataka moguće je definirati određene izvedenice poput ukupne vrijednosti, subvencije i troška vezanog uz pojedini račun.

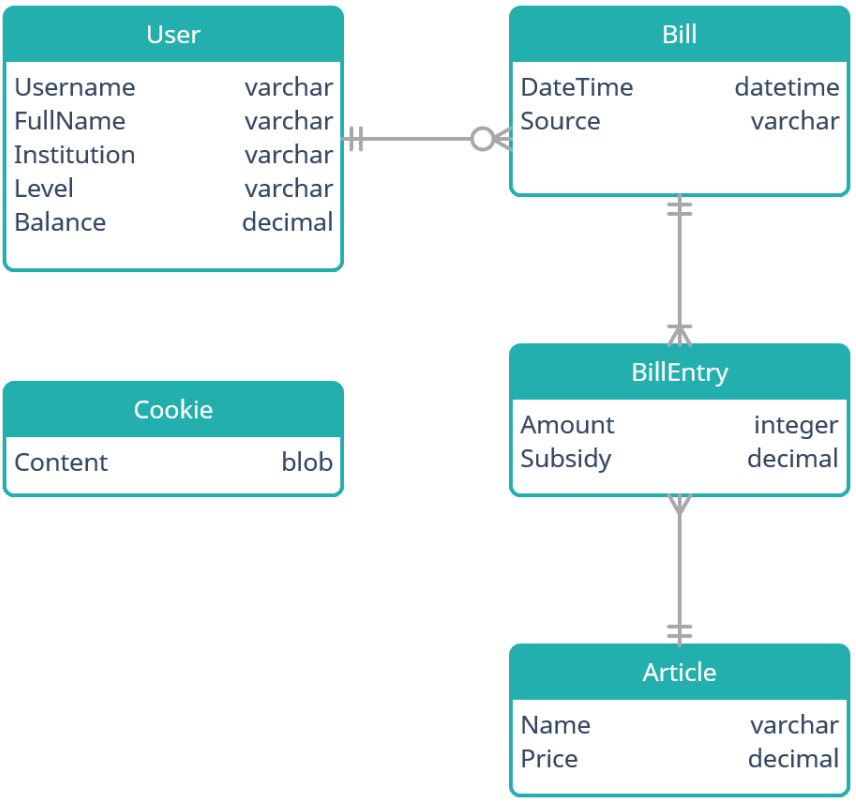
Spomenuti podaci se nakon preuzimanja pohranjuju u lokalnu *SQLite* bazu podataka. Pomagalo korišteno za rad s bazom podataka je *Exposed*[8]. Ova biblioteka omogućava rukovanje bazama

[8] Exposed - Kotlin SQL Framework: https://github.com/JetBrains/Exposed

podataka bez potrebe za pisanjem SQL koda što ubrzava rad i smanjuje vjerojatnost pogrešaka. Neke od mogućnosti koje nudi su:

- Definiranje tablica u Kotlin klasama s automatski generiranim primarnim ključevima željenog tipa
- Automatsko stvaranje definiranih tablica pri pokretanju programa
- Stvaranje DAO (*Data Access Object*) objekata za pristup podacima

Struktura baze podataka koja odgovara prethodno definiranom modelu prikazana je na dijagramu 3.5. Tablica *Cookie* drži serijalizirane podatke o internetskim kolačićima koji se trajno pohranjuju ako je uključena opcija automatske prijave. Iz popisa atributa u tablicama izuzeti su identifikacijski primarni ključevi jer su oni automatski generirani *Exposed* sustavom i nisu relevantni za ovaj dio opisa.



Dijagram 3.5. Relacijski dijagram baze podataka

Podaci o računima koje trenutni korisnik posjeduje mogu se izvesti u datoteku i kasnije otvoriti kao samostalni skup računa za pregled. Takva datoteka je JSON tipa i izrađuje se korištenjem

Kotlin Serialization[9] tehnologije. Tako izvezeni računi mogu se pripojiti trenutno učitanom skupu računa korištenjem opcije uvoza računa. Pripajanje računa moguće je obaviti i za korisnika prijavljenog na aplikaciju. Tada taj čin trajno proširuje listu računa tog korisnika.

Uz bazu podataka postoji poseban sustav za pohranu aplikacijskih postavki i sličnih informacija. Taj sustav je dio Java platforme pod nazivom *Java Preferences API*[10], a u ovom radu je korišten za pohranu jednostavnih podataka za koje nije predviđeno mjesto u glavnoj bazi podataka.

3.4. Autentikacija i mrežna komunikacija

3.4.1. Metode prikupljanja podataka

Korisnički podaci se preuzimaju s usluge ISSP na web stranicama SRCE-a. Iako taj sustav posjeduje implementaciju REST API-a[11], ona ne otkriva podatke u obliku koji je potreban za svrhe ovog projekta. Uz to, za pristup podacima koji jesu dostupni preko tog sučelja potrebna je specifična dozvola od strane pružatelja usluge. Iz tih razloga, odabrani način pristupa podacima je „struganje" sadržaja (eng. *web scraping*) s HTML stranica koje poslužitelj šalje na zahtjev. To uključuje pronalazak potrebnih informacija izravno u sadržaju dokumenta predviđenog za prikaz u internetskom pregledniku.

Treba uzeti u obzir određene nedostatke koje spomenuti pristup ima u odnosu na izravan pristup mrežnom sučelju:

- Podaci se moraju ručno pronaći, filtrirati i preoblikovati u format koji odgovara podatkovnom modelu aplikacije
- Sadržaj stranica podložan je čestim promjenama koje mogu utjecati na lokaciju podataka unutar dokumenta što zahtijeva izmjenu izvornog koda programa kako bi se prilagodio novom rasporedu
- Brzina preuzimanja podataka je osjetno manja jer poslužitelj ulaže vrijeme u sastavljanje stranice pogodne za prikaz, iako velika većina tog sadržaja neće biti iskorištena

[9] Kotlin Serialization: https://github.com/Kotlin/kotlinx.serialization
[10] Java Preferences API - sustav za pohranu preferencija: https://docs.oracle.com/javase/8/docs/technotes/guides/preferences/index.html (pristupljeno: 15.8.2021.)
[11] ISSP REST API: https://issp.srce.hr/api/Help

Za pristup internetskom sadržaju korištena je biblioteka *HtmlUnit*[12]. Ona modelira HTML dokumente i omogućava interakciju sa stranicama uz podršku za *JavaScript*. Pogodna je za pozadinsko preuzimanje podataka s web stranica sa ili bez sudjelovanja korisnika.

U vrijeme nastanka ovog rada, relevantan sadržaj web stranice s popisom studentovih računa izgleda kako je prikazano na slici 3.6. Ova lista sadrži samo sumaciju podataka o svakom računu. Od tih podataka uzimaju se oni o restoranu, datumu dnevnika i vremenu računa. Za pristup detaljima pojedinih računa potrebno je posjetiti njihove odgovarajuće stranice. Primjer jedne takve stranice prikazan je na slici 3.7. Iz popisa sastavnica računa uzimaju se svi dostupni podaci osim ukupnih iznosa, koji se mogu izračunati iz ostalih informacija o računu.

Restoran	Datum dnevnika	Vrijeme računa	Iznos računa [Kn]	Iznos subvencije [Kn]	Autorizacija*	
Restoran sa samoposluživanjem, Linija 1	07.05.2021	10:50	21.03	14.13	Skenirano	SASTAVNICE RAČUNA
Restoran sa samoposluživanjem, Linija 1	10.05.2021	11:55	19.03	12.71	Skenirano	SASTAVNICE RAČUNA
Restoran sa samoposluživanjem, Linija 2	11.05.2021	12:41	46.58	30.21	Skenirano	SASTAVNICE RAČUNA
Restoran sa samoposluživanjem, Linija 1	12.05.2021	12:45	18.27	12.17	Skenirano	SASTAVNICE RAČUNA
Restoran sa samoposluživanjem, Linija 1	14.05.2021	13:20	30.63	19.91	Skenirano	SASTAVNICE RAČUNA

Slika 3.6. *Lista studentovih računa na službenoj web stranici*

Naziv artikla	Kol.	Cijena jednog artikla [Kn]	Cijena ukupno [Kn]	Iznos subvencije [Kn]
Svinjski odrezak u umaku od šampinjona	1	18,45	18,45	9,22
Mlinci	1	3,83	3,83	2,73
Sendvič miješani	1	6,50	6,50	3,25
sok bistri nektar jabuka, udio voć.min.50% brick	1	4,00	4,00	2,00
			32,78	**17,20**

Slika 3.7. *Sastavnice računa na službenoj web stranici*

[12] HtmlUnit - pozadinski web preglednik: https://htmlunit.sourceforge.io/

Podaci o računima na poslužitelju ostaju dostupni samo nekoliko mjeseci nakon njihova nastanka, zbog čega je lokalna pohrana neophodna za ispunjavanje cilja ovog projekta. Preuzeti sadržaj se nakon preuzimanja pohranjuje u lokalnu bazu podataka.

Novopreuzeti računi često će velikim dijelom biti jednaki onima koji su preuzeti ranije, zbog čega je potrebno moći prepoznati koji računi su novi, a koji su poznati od prije. Kod preuzimanja podataka težnja je minimizirati opseg komunikacije s poslužiteljem radi smanjenja opterećenja poslužitelja i vremena potrebnog za preuzimanje. Taj zadatak je otežan zbog činjenice da službeni sustav praćenja računa ne pokazuje informaciju o točnoj sekundi izdavanja računa već je taj podatak zaokružen na minute. Naime, ako se studentu izda više računa u istoj minuti, oni će imati isto vrijeme izdavanja. Kako ti računi mogu potencijalno imati identične ukupne iznose, za pouzdanu provjeru jednakosti dvaju računa potrebno je provjeriti njihove stavke, a to zahtijeva dodatni poziv za web stranicom. Unatoč tome, uredan poredak novopreuzetih računa omogućava učinkovitu provjeru postojanja preuzetog računa u lokalnoj pohrani. Prolaskom kroz sortiranu listu novih računa silaznim poretkom (od najnovijeg prema najstarijem) može se pretpostaviti da je došao kraj novim računima čim je pronađen prvi račun koji je u cijelosti identičan prethodno preuzetom računu.

Pri izradi sustava preuzimanja računa ustanovljeno je da se lista računa koju daje poslužitelj ponekad može prikazati u nepravilnom redoslijedu. Razlog ovakvom nepredvidljivom ponašanju ostaje nejasan. Kako bi se osiguralo pravilno preuzimanje svih računa kod osvježavanja podataka, dobivena lista računa se uvijek sortira u očekivani poredak po datumu dnevnika.

3.4.2. Prijava

Računi pojedinca i svi njegovi relevantni podaci su privatno vlasništvo te je za pristup istima potrebna ovjera autentičnosti korisnika u obliku prijave. Pri pokušaju pristupa osobnim podacima, poslužitelj zatražuje prijavu korisnika kroz uslugu AAI@EduHr[13]. Budući da se pristup struganja podataka obavlja putem simulacije web preglednika, prijava se također vrši na simuliran način. Aplikacija kroz obrazac traži od korisnika njegove podatke za prijavu koji se potom unose u službeni obrazac za prijavu i šalju sustavu na provjeru. Ako sve prođe bez problema, poslužitelj dopušta pristup podacima tako što šalje zatražen internetski sadržaj.

Aplikacija također nudi mogućnost automatske prijave za koju se korisnik može opredijeliti. Zbog načina prijave koji se koristi, podaci za prijavu se u tom slučaju moraju lokalno pohraniti

[13] AAI@EduHr - autentikacijska i autorizacijska infrastruktura sustava znanosti i visokog obrazovanja: https://www.aaiedu.hr/

kako bi se kasnije mogli koristiti za ponovnu prijavu bez intervencije korisnika. Sigurnosne implikacije koje proizlaze iz ovog pristupa analiziraju se u potpoglavlju *Sigurnost*.

Komponenta aplikacije koja komunicira s poslužiteljem (*WebGateway*) ima mogućnost pohrane internetskih kolačića. Oni se koriste u svrhu praćenja identiteta klijenta s kojim poslužitelj komunicira. Obično se ovi kolačići ne pohranjuju trajno, već se brišu zatvaranjem aplikacije. To znači da se korisnik mora svaki put iznova prijaviti. Ipak, u slučaju da je odabrana opcija automatske prijave, kolačići se pohranjuju do sljedećeg pokretanja aplikacije te ako ih poslužitelj i dalje smatra važećima, prijava korisnika unosom podataka nije potrebna. Za slučaj da je prijava u međuvremenu postala nevažeća, podaci za prijavu spremljeni u aplikaciji se koriste za automatiziranu ponovnu prijavu korisnika.

3.4.3. Sigurnost

Iz opisa sustava prijave može se zamijetiti kako aplikacija ima pristup osjetljivim podacima korisničke prijave. Ti podaci se lokalno pohranjuju ako korisnik odabere opciju automatske prijave. Kako bi se smanjio novostvoreni sigurnosni rizik, poduzete su određene mjere zaštite od jednostavnog pristupa tim informacijama. Prije pohrane, korisnička oznaka i lozinka koje korisnik unosi za prijavu na AAI sustav sakrivaju se od čitanja u izvornom formatu (eng. *obfuscation*). Važno je napomenuti kako ovaj način prikrivanja podataka *nije* standardna enkripcija i služi samo za sprječavanje izravnog čitanja. Tako izmiješani podaci se tada spremaju u spremište određeno Java Preferences infrastrukturom. U slučaju Windows operacijskog sustava, to spremište se nalazi u Registry dijelu sustava.

Automatska prijava se isključuje pri prvoj ručnoj odjavi korisnika iz aplikacije kada se brišu njegovi podaci za prijavu. Također, ako je automatska prijava isključena, korisnik će uvijek biti odjavljen s poslužitelja neposredno prije zatvaranja aplikacije.

3.5. Obrada i prikaz podataka

Nakon što su podaci uspješno preuzeti ili otvoreni iz lokalne pohrane, mogu se analizirati. Sva analiza provodi se nad jednom osnovnom listom računa koja se nalazi u klasi *Repository*. Kako se aplikacija potencijalno može koristiti kroz duži period vremena, a korisnike ne moraju nužno zanimati svi računi koje posjeduju, omogućeno je postavljanje razdoblja za koje se oni razmatraju. Klasa *BillFilter* je zadužena za filtriranje računa i rukovanje s postavljenim

rasponom datuma. Svi računi izdani izvan zadanog raspona isključuju se iz konačne analize. Ovo primjerice omogućuje korisniku proučavanje skupa računa koji pripadaju točno određenoj akademskoj godini, mjesecu pa čak i jednom prometnom danu u menzi.

Pri analizi podataka korišteno je nekoliko pristupa za prikaz rezultata. Dio rezultata prikladan je za brojčani prikaz. Takvi podaci dobiveni su primjenom raznih analitičkih funkcija nad listom računa, poput zbrajanja vrijednosti određenog svojstva u svim računima. Kotlin izvorno pruža veliki broj takvih funkcija koje na intuitivan način pomažu pri rukovanju kolekcijama podataka. Najbolji primjer analize ovakve vrste je *BillSummary* klasa koja predstavlja sumaciju karakterističnih podataka iz određenog podskupa računa. Koristi se za sažimanje računa izdanih u različitim periodima kako je opisano u poglavlju 4.2.1.

Najveći dio statističkih podataka savršeno se uklapa u dijagrame, posebice stupčaste i kružne. Za izradu svih grafikona korištena je ugrađena podrška za vizualizaciju podataka JavaFX biblioteke. Velika količina podataka u određenim slučajevima može grafikone učiniti nepreglednima ako se, na primjer, veliki broj stavki prikazuje na koordinatnim osima. U takvim situacijama urednije je prikaz velikih količina teksta oko grafikona izostaviti. Nedostatak ključnih informacija o sadržaju prikaza ipak ne predstavlja problem jer je prikaz tog sadržaja omogućen prelaskom miša preko pojedinih elemenata dijagrama, odnosno preko stupaca i dijelova kružnice.

Jedan manje konvencionalan način prikazivanja rezultata analize uzima oblik kalendara. Ovdje se lista računa rastavlja na dane i raspoređeno prikazuje kroz mjesece na kalendaru. Kalendar je izveden u obliku tablice u kojoj stupci odgovaraju danima u tjednu, a reci predstavljaju tjedne u godini. Objekti tjedna sadržavaju liste od sedam dana, svaki sa svojom listom računa koja služi kao početna točka analize. Izgled i praktični opis kalendara nalazi se u poglavlju 4.2.3.

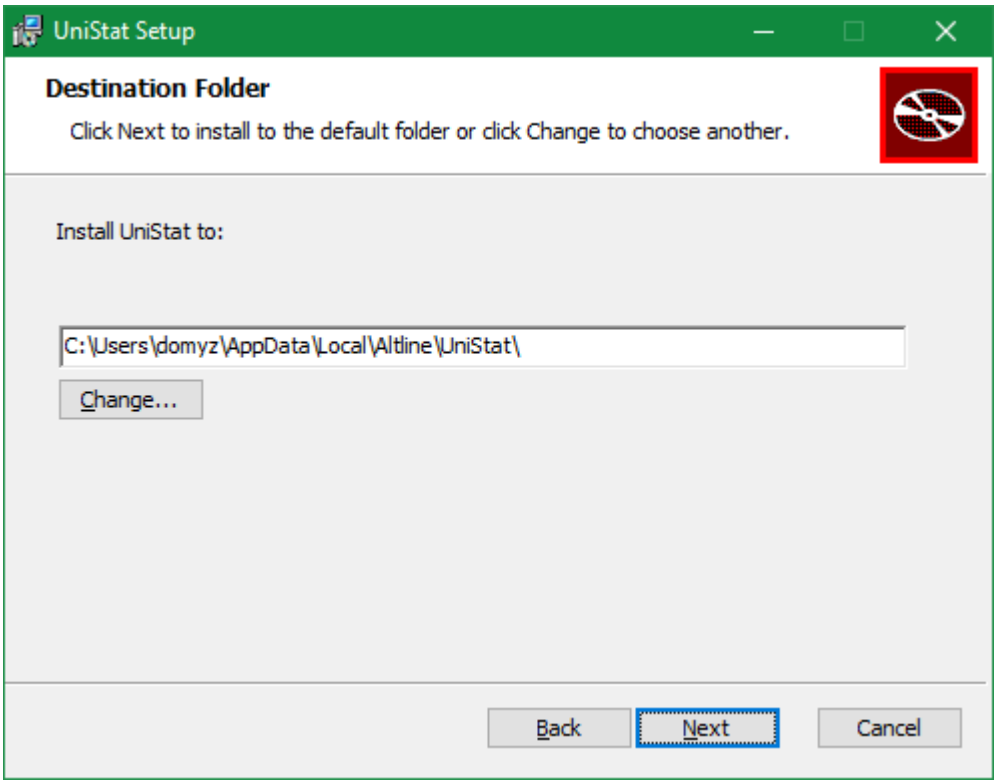
Izgled i funkcionalnost određenih komponenti korisničkog sučelja postignut je korištenjem *JFoenix*[14] biblioteke odabrane zbog alternativnog izgleda i ponašanja komponenti koje nudi.

[14] JFoenix - korisničko sučelje prema Material dizajnu: http://www.jfoenix.com/

4. PRIKAZ RADA APLIKACIJE

4.1. Instalacija

Programski paket u prilogu ovog rada namijenjen je instalaciji na Windows operacijskom sustavu. Pokretanjem instalacijske datoteke otvara se sustavni instalacijski čarobnjak u kojemu je moguće promijeniti zadanu lokaciju aplikacije kao što je vidljivo na slici 4.1.



Slika 4.1 *Instalacija aplikacije*

Instalacijom se programske datoteke raspakiraju na postavljenu lokaciju te se evidentira njena prisutnost na sustavu. Nakon instalacije stvaraju se prečaci za pokretanje programa na radnoj površini te u početnom izborniku sustava. Aplikaciju je uvijek moguće deinstalirati ponovnim pokretanjem programskog paketa i odabiranjem odgovarajuće opcije, ili putem sučelja za upravljanje programima dostupnog na upravljačkoj ploči operacijskog sustava.

Ako je na sustavu instalirana određena verzija programa, instalacija novije verzije na istu lokaciju obrisat će postojeću. Nadogradnja verzije programa pravilno se obavlja upravo na taj način.

4.2. Opis sučelja

Prvo pokretanje aplikacije od korisnika zahtijeva postavljanje izvora podataka. To se može obaviti ili prijavom na važeći AAI@EduHr račun, ili navođenjem lokacije JSON datoteke s podacima računa. Jedna takva datoteka je uključena uz programski paket aplikacije dostupan među prilozima ovog rada. Ta datoteka služi kao izvor podataka za demonstraciju rada aplikacije iz prve ruke u slučaju da pristup stvarnom korisničkom računu nije moguć.

Ako se korisnik odlučio prijaviti na AAI poslužitelj, u aplikacijski obrazac za prijavu mora unijeti one podatke koje koristi za prijavu na ostale usluge koje taj autentikacijski sustav podržava. Više detalja o ovome nalazi se u poglavlju 3.4. Alternativno, otvaranje podatkovne datoteke ne zahtijeva nikakvu prijavu te je uvijek moguće i bez internetske konekcije. Obje opcije vidljive su na obrascu za prijavu prikazanom na slici 4.2.

Slika 4.2. *Obrazac za prijavu*

Kod opcije prijave na poslužitelj, nakon uspješne autentikacije, vrši se postupak osvježavanja podataka. Korisnik za to vrijeme vidi napredak te operacije u koracima, odnosno kada se preuzimaju opći podaci, a kada računi, uz podatak koliko ih je preuzeto u stvarnom vremenu.

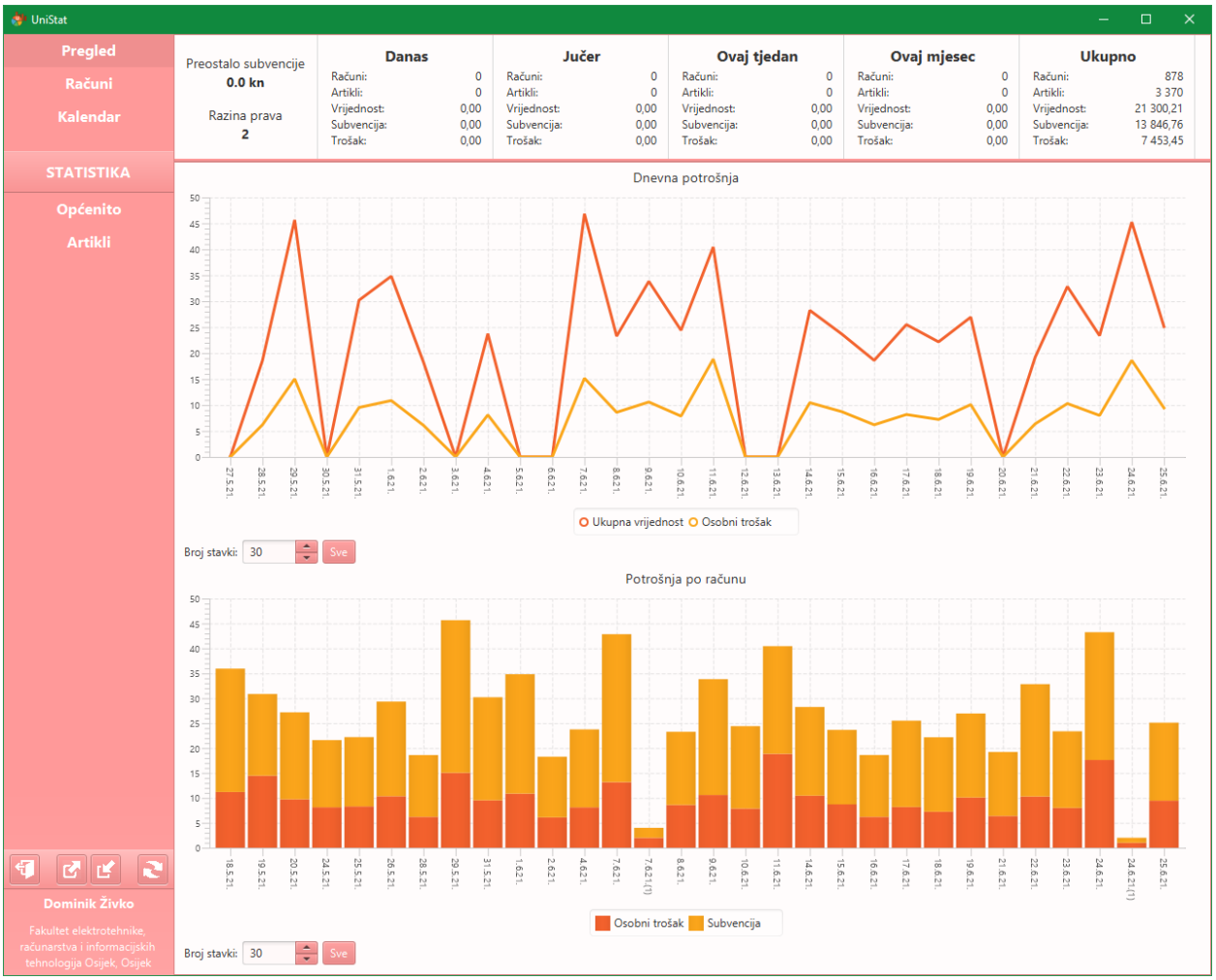
Po uspješnom preuzimanju podataka odnosno otvaranju datoteke, prikazuje se glavni aplikacijski prozor. Prozor se sastoji od prostora za glavni prikaz i od upravljačke trake koja se proteže po cijeloj njegovoj lijevoj strani. Glavni prikaz čine nekoliko zasebnih sučelja od kojih se u svakom trenutku može prikazivati samo jedno. Ti „ekrani" aplikacije nazvani su kartice (eng. *cards*), a pristupa im se preko upravljačke trake.

Ukupno postoji 5 kartica:

- Pregled
- Računi
- Kalendar
- Opća statistika
- Statistika artikala

Slijedi opis pojedinih dijelova i dodatnih mogućnosti.

4.2.1. Pregled



Slika 4.3. *Pregled najrelevantnijih podataka unutar aplikacije*

Pregled je kartica koja se prva prikazuje nakon otvaranja prozora, vidljiva na slici 4.3. Na ovom prikazu su vidljive najnovije promjene u stanju korisnikovog profila poput trenutnog iznosa subvencije na kojeg ima pravo. Uz to, vidljiv je i sažetak svih računa današnjeg odnosno jučerašnjeg dana, proteklog tjedna, proteklog mjeseca, te svih računa koji ulaze u analizu. Svaki od navedenih sažetaka uključuje:

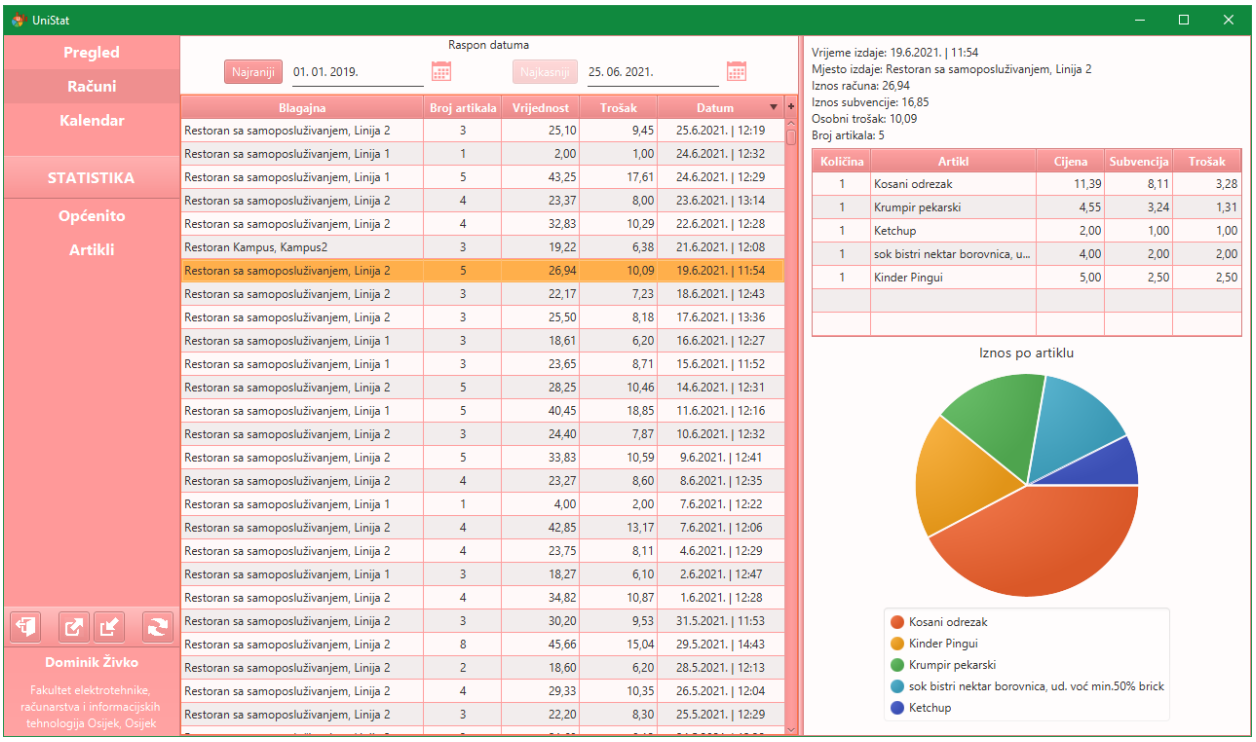
- broj računa u tom periodu
- broj artikala sadržanih u tim računima
- ukupnu vrijednost svih kupljenih artikala
- ukupan iznos subvencije na kupljene artikle
- iznos koji je korisnik platio nakon primjene subvencije (trošak)

Ispod spomenutih podataka nalaze se dva grafikona koji prikazuju trend potrošnje. Gornji, linijski grafikon prikazuje iznos dnevne potrošnje rastavljen na osobni trošak i udio subvencije, što se zbraja na ukupnu vrijednost računa. I ostali grafikoni sličnog tipa imaju iznose rastavljene na ovakav način. Donji, stupčasti dijagram sličan je prethodnom, ali se stupci ovdje odnose na iznose pojedinačnih računa umjesto na iznose generirane na dnevnoj bazi što predstavljaju točke na linijskom grafikonu.

Oba grafikona imaju vlastiti upravljač za broj stavki u dimenziji x-osi. Ovo omogućava namještanje željenog raspona računa koji se želi promatrati. S desne strane obaju prikaza uvijek se nalazi zadnji generirani račun odnosno dan na koji je izdan. Broj stavki stoga određuje koliko narednih elemenata apscise treba prikazati. Taj broj izvorno iznosi 30 što znači da prvi grafikon prikazuje trend potrošnje u zadnjih 30 dana, a drugi obuhvaća zadnjih 30 računa.

4.2.2. Prikaz računa

Ova kartica djeluje kao uvid u bazu podataka pohranjenih računa prijavljenog korisnika. Korisnik može vidjeti sve informacije o odabranom računu kroz poseban prikaz računa smješten na desnoj strani ekrana. Širina prikaza je ručno podesiva.



Slika 4.4. *Prikaz računa unutar aplikacije*

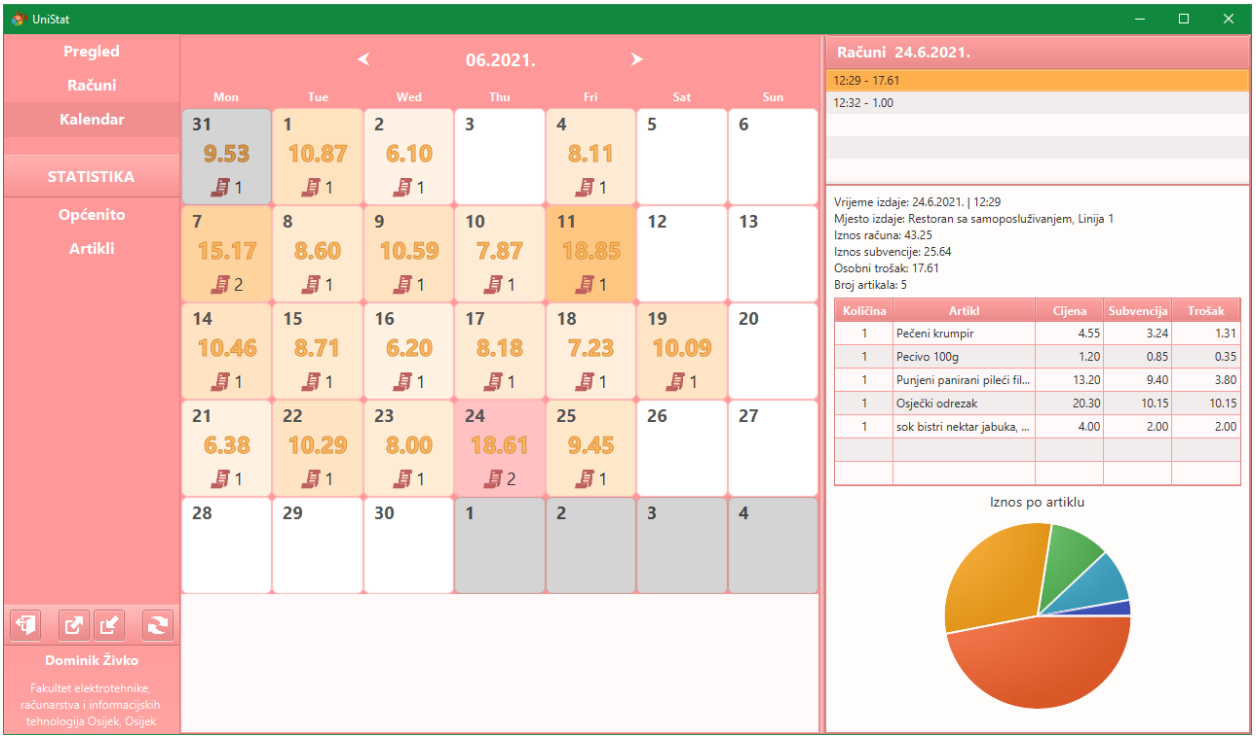
Iznad liste računa na slici 4.4. nalazi se podesivi raspon datuma prema kojemu se filtriraju dostupni računi. Lijevo polje se odnosi na donju granicu raspona, a desno odgovara gornjoj granici. Oba polja imaju pripadajuće dugme koje u njih postavlja datum graničnog računa.

Tablica računa sastoji se od stupaca restorana, tj. specifične blagajne restorana na kojoj je račun izdan, od broja artikala kupljenih pod računom, vrijednosti i troška vezanih uz račun te datuma i vremena njegovog izdavanja. Iz izbornika označenog znakom '+' moguće je prikazati ili sakriti pojedine stupce tablice. Jedan stupac koji je na početku sakriven je stupac subvencije računa.

Označavanjem jednog od računa u tablici, na desnoj strani kartice se prikazuju njegove stavke i sve ostale informacije. Kao vizualno pomagalo također je prikazan kružni dijagram udjela pojedinih artikala u ukupnoj vrijednosti računa.

4.2.3. Kalendar

Kalendarski prikaz računa vidljiv na slici 4.5. daje vizualizaciju potrošnje kroz godinu. To je prikaz standardnog kalendara gdje se za svaki dan prikazuje broj izdanih računa i studentov osobni trošak na subvencioniranu prehranu. Jačina boje pozadine u polju dana predstavlja relativnu veličinu plaćenog iznosa u odnosu na najveći i najmanji iznos u tom mjesecu. Prema tome, dan u odabranom mjesecu na koji je student platio najveći iznos bit će predstavljen najtamnijom bojom, dok će dan najmanjeg iznosa biti najsvjetlije osjenčan. Svi iznosi između ta dva ekstrema nose jačinu boje koja je negdje u tom spektru. Dani na kojima korisnik nema računa ostavljeni su prazni i nisu osjenčani.



Slika 4.5. *Kalendarski prikaz računa*

Bilo koji dan u kalendaru moguće je odabrati kako bi se s desne strane prikazali pripadajući računi, ako oni postoje. Širina prikaza računa može se proizvoljno podešavati. Na vrhu prikaza detalja nalazi se lista svih računa odabranog dana. Računi u listi se mogu raspoznati po vremenu izdavanja i trošku. Ispod liste se nalazi prikaz detalja odabranog računa isti kao u kartici *Računi*.

4.2.4. Opća statistika

Ovo je prva kartica u sekciji *Statistika* i odnosi se na općenite statističke podatke. Na njoj se nalaze tri grafikona:

- mjesečna potrošnja
- računi po blagajni
- potrošnja po blagajni

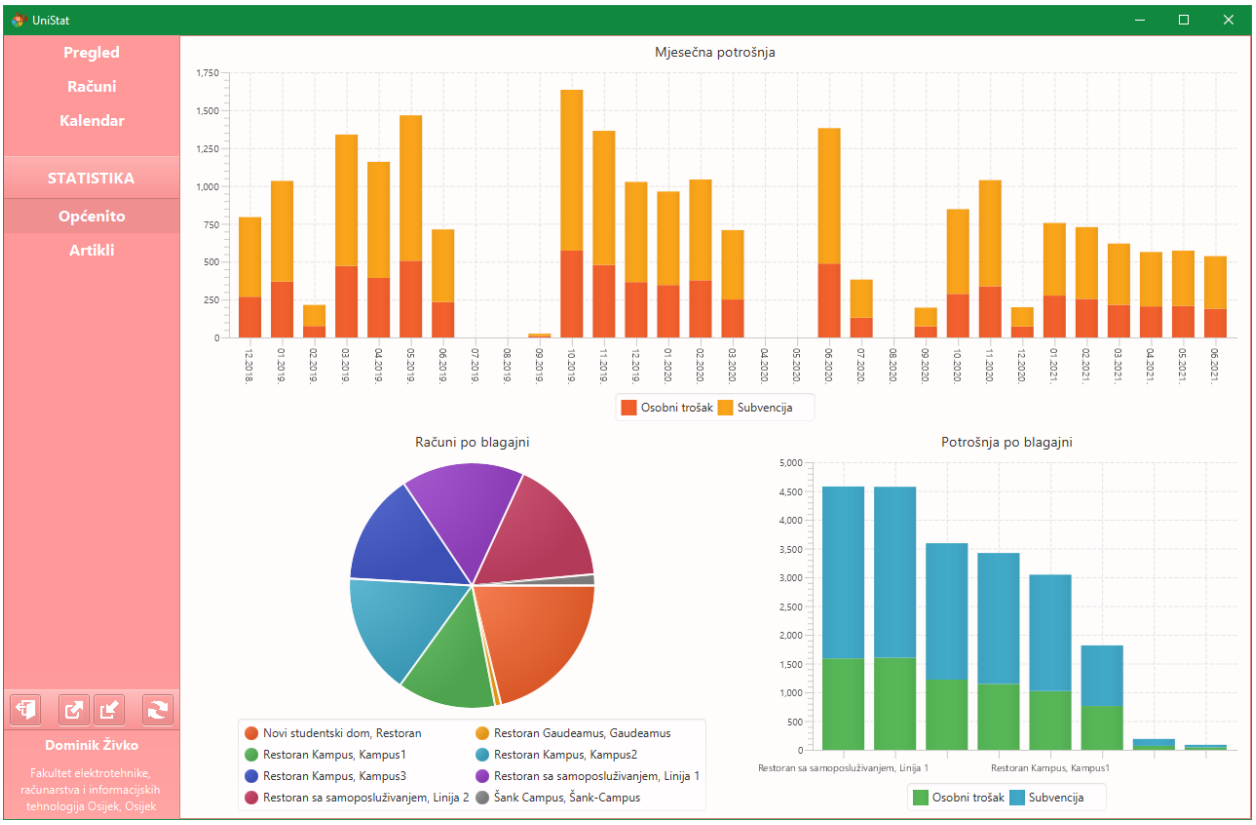
Stupčasti grafikon mjesečne potrošnje jednostavno objedinjava potrošnju svih računa po mjesecima izdavanja. Ukupna vrijednost računa rastavljena je na osobni trošak i udio subvencije.

Kružni dijagram broja računa po blagajni prikazuje učestalost prometa korisnika na svim blagajnama koje je koristio. Svaka blagajna koja se može naći u listi postojećih računa nalazi se

u ovom dijagramu, a udio određene blagajne u površini kružnice je veći što se blagajna više posjećuje.

Drugi stupčasti grafikon odnosi se na potrošnju po blagajni. Ovdje se također vide sve posjećene blagajne, ali su umjesto broja računa prikazani podaci o potrošnji.

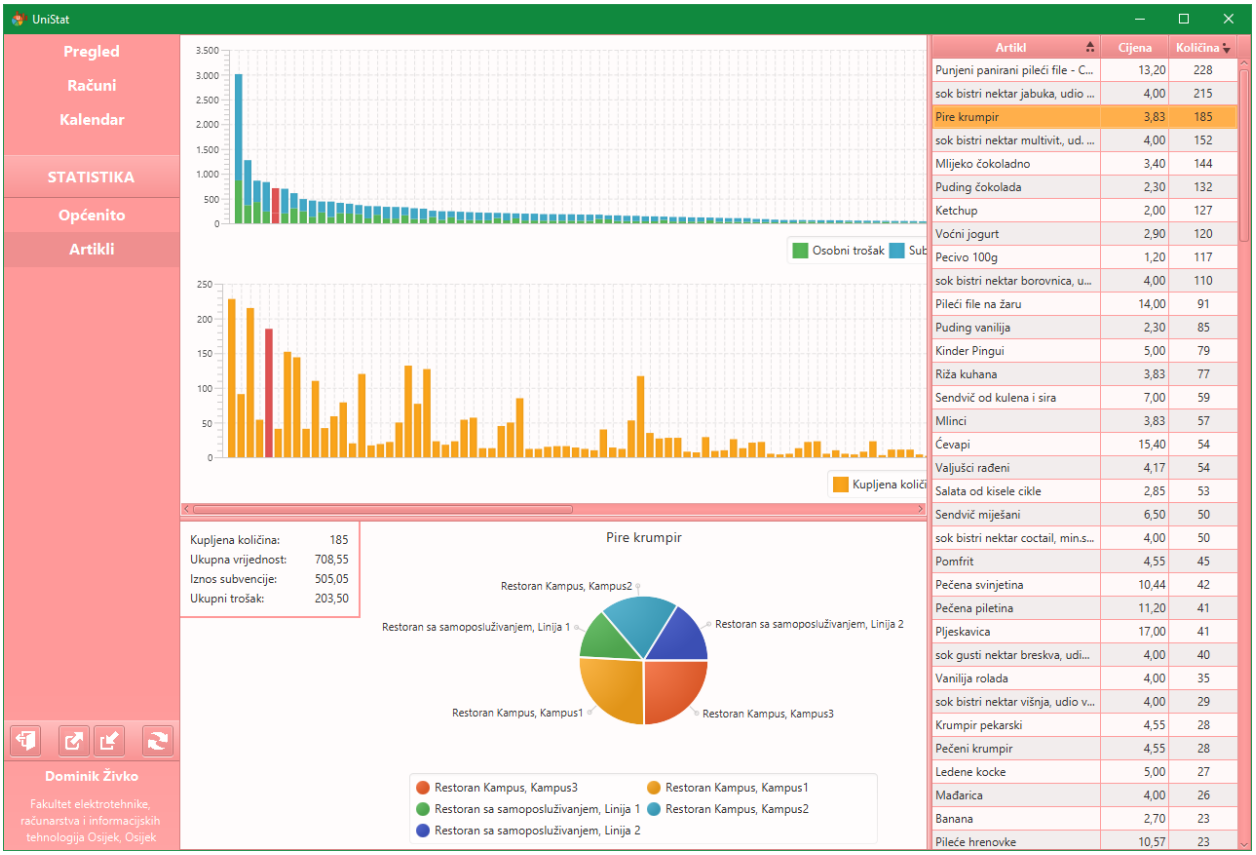
Opisani grafikoni vidljivi su na slici 4.6.



Slika 4.6. *Opća statistika računa*

4.2.5. Statistika artikala

Druga statistička kartica pod nazivom *Artikli*, vidljiva na slici 4.7., služi za promatranje statističkih podataka vezanih za pojedine artikle koje je korisnik barem jednom kupio.



Slika 4.7. *Statistika po artiklima*

Skup artikala vidljiv je u dvije forme, tabličnoj i grafičkoj. Tablična forma popisa artikala prebiva u proizvoljno proširivom desnom dijelu kartice i sadrži stupce s nazivom artikla, cijenom i ukupnom količinom kupljenih artikala te vrste. Grafički, ti artikli su prikazani na dva uparena stupčasta grafikona locirana u gornjem dijelu sučelja. Grafikoni su horizontalno usklađeni jedan s drugim tako što im stupci redom pokazuju na iste artikle. Gornji dijagram prikazuje ukupnu potrošnju vezanu za svaki artikl, dok donji dijagram prikazuje njihovu kupljenu količinu. Redoslijed artikala u grafikonima sortiran je silazno prema njihovoj ukupnoj vrijednosti. U slučaju da je broj artikala za koje korisnik ima uvid veći nego što bi pregledno stalo u prozor, grafikoni će se proširiti izvan vidljivog područja za njihov prikaz. Tada ih je moguće pomicati povlačenjem miša bilo gdje po prostoru prikaza kao i po posvećenom klizaču.

Klikom, bilo na redak u tablici ili na stupac u nekom od dijagrama, označava se pripadajući artikl te se u donjem dijelu prozora prikazuje detaljna statistika tog artikla. Taj prikaz se sastoji od podataka o kupljenoj količini, vrijednosti, subvenciji i trošku artikla kao i od kružnog dijagrama koji prikazuje udio kupljenog artikla po različitim blagajnama. Granicu između gornjeg i donjeg prikaza u prozoru također je moguće proizvoljno podešavati.

4.2.6. Upravljačka traka

U glavnom aplikacijskom prozoru uvijek je vidljiva upravljačka traka koja služi kao navigacijska kontrola i kao mjesto poziva glavnih aplikacijskih operacija. Te operacije su, redom:

- Odjava korisnika
- Izvoz računa
- Uvoz računa
- Osvježavanje podataka

Na dnu trake prikazane su osnovne informacije o prijavljenom korisniku, a to su njegovo puno ime i naziv obrazovne ustanove koju pohađa. (slika 4.8.)

Slika 4.8. *Upravljačka traka*

5. ZAKLJUČAK

Problem postavljen u uvodnom dijelu rada uspješno je riješen izrađenom aplikacijom. Umjesto kratkotrajnog i često nepreglednog uvida u svoje podatke o potrošnji u studentskoj menzi, studenti uz ovo pomagalo imaju priliku trajno sačuvati, analizirati i uspoređivati te informacije kroz jednostavno sučelje i pregledne grafikone.

Tijekom izrade programa bilo je potrebno suočiti se s brojnim izazovima, od konfiguracije jezičnih tehnologija, preko odabira i implementacije pogodne aplikacijske strukture, do stvaranja instalacijske datoteke koja se može pokrenuti na drugim sustavima. Svi postavljeni ciljevi na kraju su uspješno realizirani.

Trenutni sustav preuzimanja korisničkih podataka nije idealan budući da koristi princip struganja sadržaja web stranica. Poželjno unaprjeđenje tog sustava nedvojbeno bi bio prelazak na korištenje REST API-a, naravno ako potrebni podaci postanu dostupni u budućnosti.

U pogledu mogućih proširenja funkcionalnosti aplikacije uvijek je moguć napredak. Primjerice, sustav filtriranja računa može se unaprijediti dodavanjem mogućnosti odabira blagajni koje se žele razmatrati. Uz to, ponuda analitičkih podataka može se proširiti novim dijagramima i inovativnim vizualizacijama podataka koje bi udovoljile znatiželjnijim korisnicima.

LITERATURA

[1] ISSP, Sveučilišni računski centar, Sveučilište u Zagrebu, dostupno na: https://www.srce.unizg.hr/issp [19.8.2021.]

[2] Kotlin docs: Comparison to Java, Kotlin Foundation, JetBrains, 2021., dostupno na: https://kotlinlang.org/docs/comparison-to-java.html [19.8.2021.]

[3] P. Pedamkar, Java vs Kotlin, EDUCBA, 2020., dostupno na: https://www.educba.com/java-vs-kotlin/ [19.8.2021.]

[4] M. Pawlan, What is JavaFX?: Key Features, Oracle, 2013., dostupno na: https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm [19.8.2021.]

SAŽETAK

Postojeći sustav uvida u podatke o potrošnji na subvencioniranu prehranu prikazuje samo najosnovnije informacije o studentovim prikupljenim računima. U ovom radu je dokumentirano unaprjeđenje tog sustava u obliku računalne aplikacije. Aplikacija na osnovu podataka preuzetih sa službenih internetskih stranica Sveučilišnog računskog centra (SRCE) prikazuje razne statističke podatke o troškovima studentske prehrane. Opisane su metode preuzimanja podataka, njihova pohrana u lokalnoj bazi podataka, te obrada i vizualizacija korištenjem alata za prikaz podataka u JavaFX programskom okviru. Navedene su korištene prednosti Kotlin programskog jezika koji je izabran zbog svojih mogućnosti obrade podataka i sažimanja programskog koda. Ovaj rad uključuje opis instalacije i uporabe izrađene aplikacije, kao i cjelokupni opis korisničkog sučelja uz demonstraciju rezultata analize provedene nad primjerom skupa korisničkih podataka.

Ključne riječi: obrada podataka, podaci o potrošnji, račun, statistika, subvencionirana prehrana

ABSTRACT

Application for consumption analysis on the student X-card

The existing data service for subsidised student meals provides only the simplest form of data regarding the students' collected receipts. This paper documents an improvement of the service in the form of a desktop application. The application shows a statistical summary of a student's spending on subsidised meals by analysing data from the official SRCE database. The paper describes methods of data collection, local database storage, as well as data processing and visualisation with the JavaFX application framework. Also mentioned are the utilised advantages of the Kotlin programming language which is used for its data processing abilities and concise code style. The guide for installation and usage of the application is provided as the final part of the document, as is the presentation of the analysis results derived from an example dataset.

Keywords: data processing, receipt, spending data, statistics, subsidised meals

PRILOZI

Prilog 1. Izvorni kod aplikacije na GitLab repozitoriju:
https://gitlab.com/domy.zivko/unistat2

Prilog 2. Kopija izvornog koda aplikacije - verzija 1.2.0
(datoteka: UniStat-1.2.0-source.zip)

Prilog 3. Programski paket i demonstracijski podaci (datoteka UniStat-1.2.0.zip)
Uključene datoteke:

UniStat-1.2.0.msi - instalacijski paket aplikacije
UniStat-racuni.json - lista računa (za otvaranje u aplikaciji)

Prilog 4. Dokumenti ovog rada u pdf i docx formatima:

Aplikacija_za_analizu_potrosnje_na_X-ici.pdf
Aplikacija_za_analizu_potrosnje_na_X-ici.docx