Jedna aplikacija za budžetsko knjigovodstvo

Marko Borak, Boško Bogojević, Zoran Ćirović
Akademija tehničko-umetničkih strukovnih studija Beograd, odsek VISER
Beograd, Srbija
marko.borak@viser.edu.rs, bosko.bogojevic@viser.edu.rd, zoran.cirovic@viser.edu.rs

Sažetak—**Ovaj rad prezentuje realizaciju jedne aplikacije za budžetsko knjigovodstvo. Aplikacija je razvijana za potrebe rada više grupa operatora različitih privilegija, koji rade sa različitih lokacija. Specifičnost aplikacije ogleda se u posebnim mogućnostima: grupe operatera mogu da definišu specifično knjiženje, a da se pri tome omogući održavanje zajedničkih podataka koji su potrebni za formiranje zajedničkih izveštaja, postoji kontinuirano praćenje finansija svakog odseka, automatsko knjiženje uplata studenata i kontrola naloga od strane različitih vrsta rukovodilaca.**

Ključne riječi-REST; Vue; element-plus; .Net Core; servisi; budžet; knjigovodstvo; šifre; knjiženje, fakture

UVOD

Osnivanjem Akademije tehničko-umetničkih strukovnih studija Beograd, pojavila se praktična potreba za jednim objedinjenim računovodstvenim sistemom, koji bi uvažavao specifičnosti pojedinih odseka. Akademija je nastala spajanjem pet visokih škola, sadašnjih pet odseka. Svih pet škola su koristili različite programe za knjigovodstvene potrebe. Spajanjem u jedno pravno lice tj. Akademiju, pokazalo se neophodno imati jedinstven program koji bi obuhvatio rad više "starih" službi.

Nakon analize potreba budućih korisnika i uvidom u realno stanje resursa definisani su neophodni zahtevi koje aplikacija treba da ispunjava. Postavljeni zahtevi su: (i) vođenje budžetskog knjigovodstva u zakonskim okvirima [1], [2], (ii) kontinuirano praćenje finansija svakog odseka posebno, (iii) knjiženje uplata studenata i (iv) kontrola knjiženja na nivou odseka odnosno izvoda.

Pre realizacije sopstvenog sistema, autori rada su istražili ponudu domaćeg poslovnog softvera za budžetsko knjigovodstvo koji bi mogao da odgovori potrebama Akademije, a da je pri tome u prihvatljivom cenovnom opsegu. Analizirane su tri aplikacije: *Softek* [3], *Obrazovni informator* [4] i *Abcsoft* [5]. Sve aplikacije imaju relativno veliki broj korisnika i iskustvo u radu sa budžetskim knjigovodstvom.

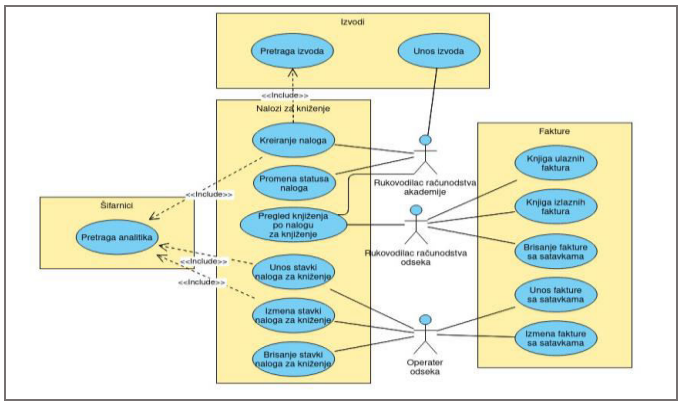
Osobine navedenih aplikacija testirali smo nakon instalacije i u saradnji sa zaposlenim operaterima u službi računovodstva. Takođe, obavili smo razgovore sa eventualnim prodavcima. Sve tri aplikacije ispunjavaju zahtev (i). Aplikacija *Softek* obuhvata širok spektar primena i opcija koje nisu od interesa za rad naše institucije. Aplikacija nije prilagođena obrazovnim institucijama, naročito ako one imaju veći broj studenata i potrebe da integrišu studentske finansije u informacioni sistem. Od preko 250 referenci koje su objavljene na sajtu, samo je jedna visokoškolska ustanova. Aplikacija delimično odgovara na zahteve (ii) odnosno (iii), dok (iv) ne zadovoljava. Poslovna aplikacija kompanije „Obrazovni informator" [4], zaposlenima u Akademiji dostupna je izvesno vreme. Ova aplikacija može da odgovori na zahtev (ii) ali ne nudi opciju neprekidnog i trenutnog stanja svakog odseka, i nema deo koji bi bio poseban za praćenje studentskih uplata (iii). Takođe, ne postoji podržan rad sa različitim vrstama korisnika kao ni odgovarajuće kontrole knjiženja na nivou odseka odnosno Akademije. Posebno dobra strana ove aplikacije je mogućnost neprekidnog praćenja finansijskog plana. Treba dodati da ista kompanija svojom izdavačkom delatnošću prati sve promene koje se tiču aktuelnih propisa što predstavlja veliku pomoć u radu operatera. Što se tiče aplikacije *Abcsoft*, ona nudi veći broj aplikacija prilagođenih određenim kategorijama. Jedna od njih je i aplikacija za budžetsko knjigovodstvo. Osim prvog zahteva nismo našli druge opcije koje bi bile nama od interesa.

Dakle, analizom zahteva i istraživanjem postojećih rešenja u okviru cenovnog ranga [3]-[5], došlo se do zaključka da je najbolje razviti sopstveno rešenje.

OSNOVNI KONCEPTI REŠENJA

Početni zahtevi nisu bili precizni ni kompletni, što je tipično u sličnim projektima. Očekivali smo da će se tokom razvoja zahtevi menjati, a ciljevi proširivati. Zato smo razvoj fokusirali na inkrementalnoj tehnici razvoja softvera uz agilne tehnike poštujući korisničke zahteve, [6]. Očekivano je da isti ili sličan način razvoja primenjujemo u budućim fazama razvoja informacionog sistema.

Definisane su vrste korisnika aplikacije: operater jednog odseka, rukovodilac odseka i rukovodilac akademije. Postavljeni su slučajevi korišćenja. Na Sl. 1. prikazan je deo slučajeva korišćenja.



Slika 1. Deo slučajeva korišćenja

Imajući u vidu da se operateri Službe računovodstva nalaze smešeni na matičnim odsecima, na različitim lokacijama i sa različitim resursima, odabrano je klijent-server rešenje, pri tome je za klijentsku aplikaciju odabrana veb aplikacija. Na ovaj način su resursi klijentske aplikacije i sigurnosni mehanizmi kontrolisani preko klijent-server arhitekture i preko karakteristika veb čitača. Istovremeno je svim krajnjim korisnicima veb interfejs bio poznat i omogućavao najbržu obuku. Sa druge strane, razvojni tim je mogao da preko serverske aplikacije lako kontroliše izmene i nadzor nad podacima.

Serversko rešenje je realizovano na Linux platformi uz primenu .Net Core platforme. Za server baze podataka odabran je MS SQL server. Na serveru su postavljene produkciona i test baza. Baza upravlja korisničkim privilegijama, obezbeđuje sigurni pristup podacima, njihov integritet kao i automatizovano kreiranje kopija. Do podataka se pristupa preko veb servisa koji realizuju CRUD operacije (eng. Create, Read, Update, Delete), poslovnu logiku i kontrolere kao i poglede na strani klijenata potrebni u radu sa knjigovodstvom Akademije. Servisi su realizovani kao .NET Core servisi, [7] a postavljeni su na drugom Linux serveru. Servisi kontrolišu korisnički pristup, logiku, upravljaju podacima. Servisi omogućavaju da operateri jednog odseka mogu da pristupaju specifičnim podacima i rade specifična knjiženja. Operateri jednog odseka mogu da generišu specifične izveštaje, vrše sopstvene kontrole i organizuju podatke na način koji to njima odgovara. Istovremeno, administratori računovodstva mogu da vide podatke od svih operatera na način kako to sami operatori vide, ali i u vidu posebno generisanih izveštaja.

Tokom razvoja serverske aplikacije, istovremeno je razvijana i veb aplikacija za klijente. Veb aplikacija je realizovana pomoću Vue radnog okvira, [8] i primenom specijalizovane biblioteke *Element-plus*, [9]. Izbor ove biblioteke urađen je poređenjem karakteristika sličnih biblioteka i zahtevom za brz razvoj jednog modernog korisničkog interfejsa jedne poslovne aplikacije sa poslednjom verzijom Vue okvira. Primena Vue okvira omogućio je komponenti pristup, a ovo je dalo efikasan i dosledan pristup u projektovanju i razvoju korisničkog interfejsa. Aplikacija je dostupna samo preko virtuelne privatne mreže. Autorizacija u komunikaciji obezbeđena je upotrebom JWT-a (eng. JSON Web Token).

Korišćeni alati odnosno tehnologije su besplatni odnosno otvorenog koda i implementiraju se na Linux serveru posredstvom virtuelne mašine. Primena virtuelne mašine omogućava potpuno čuvanje rezervnih kopija kao i sigurni potpuni oporavak svih podataka i funkcija.

SERVERSKA APLIKACIJA

Serverska aplikacija je zasnovana na primeni MVC (eng. Model View Controller) arhitekture. U nastavku dajemo opis osobina serverske aplikacije.

A. *REST*

Ovaj koncept je primenjen u razvoju svih funkcija sistema. REST (eng. Representational State Transfer) je od svog prvog uvođenja kao novog pojma [10], brzo postao veoma popularan u razvoju klijent-server aplikacija [11]. Komunikacija između klijenta i servera se vrši pomoću HTTP metoda. Zahtevi moraju biti kompletni i nezavisni, bez praćenja stanja. Resurs može biti sve ono što je moguće adresirati na veb-u. Neki od resursa korišćeni u našoj aplikaciji su: izvod, analitike, kontni plan, nalozi za knjiženje…

RESTful veb servisi izrađeni su na arhitekturi REST-a. Servisi su realizovani pomoću biblioteke otvorenog koda ASP.NET Core.

Servisi se sastoje od više realizovanih kontrolerskih klasa. Ovi kontroleri bazirani su na primeni .NET Core klase *ControllerBase*. Klijentska aplikacija komunicira sa serverskom aplikacijom pomoću kontrolera. Dakle, kontroleri vrše obradu zahteva i vraćanje odgovarajućih pogleda u skladu sa poslovnom logikom. Pri tome koriste modele kreirane na osnovu tabela u bazi podataka, odnosno modele koji su prilagođeni pogledima (eng. ViewModel) i vrše promene u bazi podataka.

Ovaj sistem ima veliki broj resursa odnosno pratećih API-ja za rad sa tim resursima. Na Sl. 2. dajemo primer realizacije šifarnika „Kontni plan“:

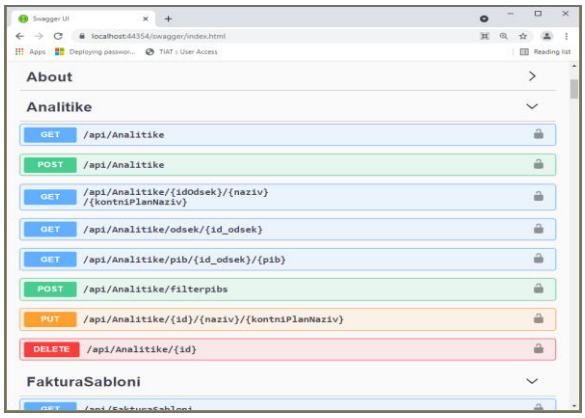
```
[Route("api/[controller]")]
[ApiController]
1 reference
public class KontniPlanoviController : ControllerBase
{
    private readonly ATUSSrDbContext _context;
    0 references
    public KontniPlanoviController(ATUSSrDbContext context)...
    // GET
    [HttpGet]
    0 references
    public async Task<ActionResult<IEnumerable<KontniPlan>>> GetKontniPlanovi()...
    // GET
    [HttpGet("{id}")]
    0 references
    public async Task<ActionResult<KontniPlan>> GetKontniPlan(string id)...
    // PUT
    [HttpPut("{id}")]
    0 references
    public async Task<IActionResult> PutKontniPlan(string id, KontniPlan kontniPlan)...
    // POST
    [HttpPost]
    0 references
    public async Task<ActionResult<KontniPlan>> PostKontniPlan(KontniPlan kontniPlan)...
    // DELETE
    [HttpDelete("{id}")]
    0 references
    public async Task<IActionResult> DeleteKontniPlan(string id)...
    2 references
    private bool KontniPlanExists(string id)...
}
```

Slika 2. Prikaz rada za kontni plan

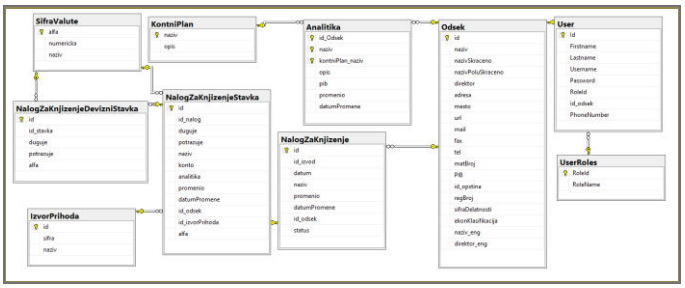
Svi servisi su realizovani i dokumentovani preko OpenAPI standarda, [12]. Pogled na jedan deo automatski kreirane dokumentacije moguće je videti na Sl. 3. Zahvaljujući primeni ovog standarda dokumentacija se izvodi sve vreme tokom razvoja, greške u razumevanju servisa su gotovo isključene. Na primer, razvojni programeri koji koriste servise na klijentskoj strani mogu da jasno vide detalje API-ja, da ih testiraju nezavisno od ostatka aplikacije što doprinosi efikasnijem timskom radu i razumevanju između timova.

B. *ORM*

Deo modela podataka u bazi je prikazan na Sl. 4. U serverskoj aplikaciji rad sa podacima se ostvaruje primenom ORM biblioteke EntityFramework. Mapiranje i organizacija modela urađena je primenom FluentAPI-a [13], kojim se definišu svojstva pojedinih entiteta kao i način povezivanja. Deo koda za entitet Analitika prikazan je na Sl. 5.



Slika 3. Pogled na stranicu *Swagger*



Slika 4. Prikaz dela modela podataka

```
public virtual DbSet<Analitika> Analitike{get; set;}
public virtual DbSet<Faktura> Fakture{get; set;}

protected override void OnModelCreating(ModelBuilder
modelBuilder){

modelBuilder.Entity<Analitika>(entity =>
{
   entity.HasKey(e => new { e.IdOdsek, e.Naziv, e.KontniPlanNaziv });
   entity.ToTable("Analitika");

   entity.Property(e => e.IdOdsek).HasColumnName("id_Odsek");
   entity.Property(e => e.Naziv)
  .HasMaxLength(50)
  .HasColumnName("naziv");
   entity.Property(e => e.KontniPlanNaziv)
  .HasMaxLength(6)
  .HasColumnName("kontniPlan_naziv");
   entity.Property(e => e.DatumPromene)
  .HasColumnType("datetime")
  .HasColumnName("datumPromene");
   entity.Property(e => e.Opis)
  .HasMaxLength(150)
  .HasColumnName("opis");
   entity.Property(e => e.Pib)
  .HasMaxLength(50)
  .HasColumnName("pib");
   entity.Property(e => e.Promenio)
  .HasMaxLength(50)
  .HasColumnName("promenio");
   entity.HasOne(d => d.IdOdsekNavigation)
  .WithMany(p => p.Analitike)
  .HasForeignKey(d => d.IdOdsek)
  .OnDelete(DeleteBehavior.ClientSetNull)
  .HasConstraintName("FK_Analitika_Odsek");
    entity.HasOne(d => d.KontniPlanNazivNavigation)
  .WithMany(p => p.Analitike)
  .HasForeignKey(d => d.KontniPlanNaziv)
  .OnDelete(DeleteBehavior.ClientSetNull)
.HasConstraintName("FK_Analitika_KontniPlan");
    entity.HasOne(d => d.PibNavigation)
  .WithMany(p => p.Analitike)
  .HasForeignKey(d => d.Pib)
  .HasConstraintName("FK_Analitika_Firma");
     });
```

Slika 5. Definisanje entiteta *Analitika*

Kontekstna klasa ATUSSrDbContext objedinjuje modele u skupove podataka. Ova klasa daje i okvir koji se koristi za pristup do pojedinih skupova odnosno do pojedinačnih entiteta. Klasa je formirana koristeći EntityModel, odnosno klasu DbContext.

KLIJENTSKA APLIKACIJA

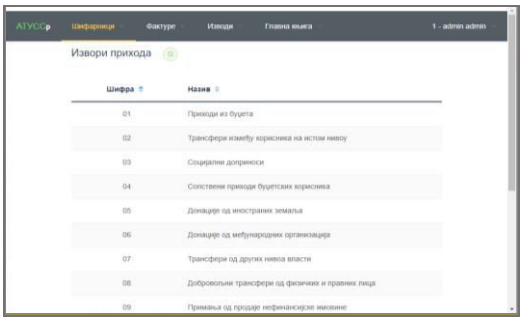
A. *Šifarnici*

Šifarnici čine deo aplikacije koji omogućava definisanje podataka koje se koriste tokom knjiženja. Neki šifarnici su zajednički za sve odseke, dok su drugi specifični za pojedine odseke. Na primer, rad sa firmama je zajednički, tj. jednom uneti podaci o nekoj firmi dostupni su svim odsecima. Firma se može koristiti za izlazne odnosno ulazne fakture kao i pri knjiženju. Međutim, analitike koje odseci postavljaju i pomoću kojih knjiže, zavise od odseka do odseka. Ove analitike omogućavaju specifično knjiženje za svaki odsek. Tako definisane analitike po odsecima imaju zajedničke podatke za različite odseke, a to su propisana konta, [14].

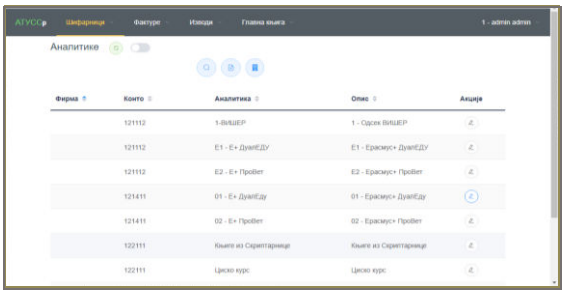
U okviru šifarnika nalaze se: opštine, kontni planovi, izvori prihoda, firme, analitike, šifre valuta i računi.

Na narednim slikama prikazani su dva pogleda za dve vrste šifarnika: Sl. 6. Izvori prihoda, Sl. 7. Analitika. Posebna pažnja posvećena je izradi šifarnika Analitika. Radi se o šiframa koje su specifične za pojedine odseke i na osnovu kojih svaki od odseka može da radi na sopstveni način knjiženja. Pogled za prikaz Analitika poseduje i prekidačku kontrolu na vrhu koja je dostupna samo rukovodiocima. Pomoću ove kontrole rukovodilac može da menja pogled iz uloge koja je podrazumevano standardni operater jednog odseka u ulogu rukovodioca.

Rukovodilac odseka poseduje dodatne mogućnosti za upravljanje podacima na nivou odseka. To su privilegije brisanja i promene podataka koje su uneli operateri istog odseka, kao i posebne kontrolne funkcije. Posebna prava uključuju mogućnost kontrole rada drugih operatera. Rukovodilac ima uvid u unete podatke kao i podatak ko i kada je uneo isti. Rukovodilac Akademije ima iste mogućnosti kao i rukovodilac odseka ali nad svim operaterima Akademije. Osim toga, rukovodilac ima neke specifične funkcije, na primer on otvara i zatvara naloge.



Slika 6. Šifarnik *Izvori prihoda*



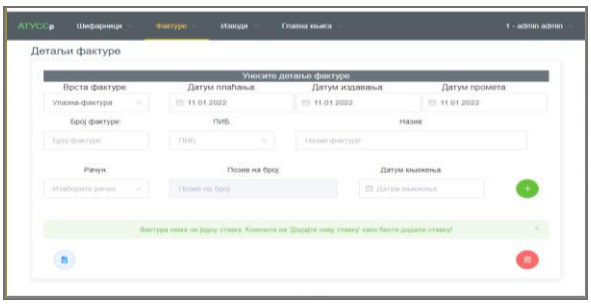
Slika 7. Šifarnik *Analitika*

B. *Fakturisanje*

Fakturisanje je podeljeno na pregled faktura, dodavanje nove fakture, knjigu ulaznih i knjigu izlaznih faktura. Postoji i mogućnost snimanja šablona sa karakterističnim podacima fakture za često korišćene fakture, čime se olakšava rad za uplate i isplate koje se često ponavljaju.

Na Sl. 8. je prikazana stranica za dodavanje nove fakture. Svaka faktura može imati jednu ili više stavki.

Za potrebe dodavanja i izmene postojećih faktura korišćenje su Vue komponente kako bi se realizovala modularna implementacija. Posebne komponente su napravljene za prikaz detalja fakture, prikaz stavki faktura i realizacija izveštaja u Microsoft Word i Excel programima.

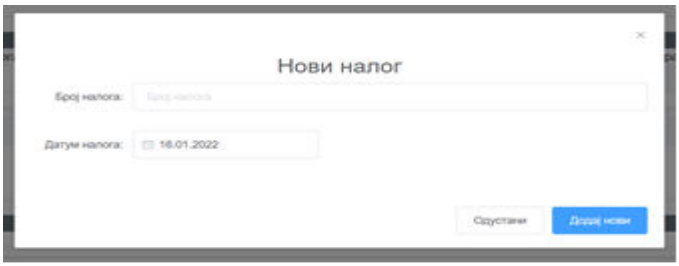


Slika 8. Dodavanje nove fakture

Pregled svih faktura se vrši preko knjiga ulaznih i izlaznih faktura koje sadrže, pored neophodnih podataka za fakture, ukupnu sumu za izabrani vremenski period, čime se mogućnost greške značajno smanjuje.

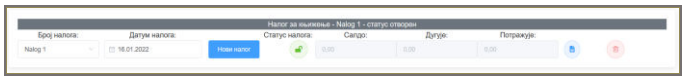
C. *Knjiženje*

Uz poštovanje početnih postavljenih zahteva (i)-(iv), knjiženje naloga predstavlja složenu poslovnu operacija. U našem slučaju knjiženje razdvajamo na unos tj. otvaranje novih naloga, odnosno pregled već unetih naloga. Nalog za knjiženje je dokument koji se koristi za evidenciju poslovnih promena. Otvaranje naloga se vrši preko modalne forme za unos naziva i datuma naloga. Ova forma je prikazana na Sl. 9.



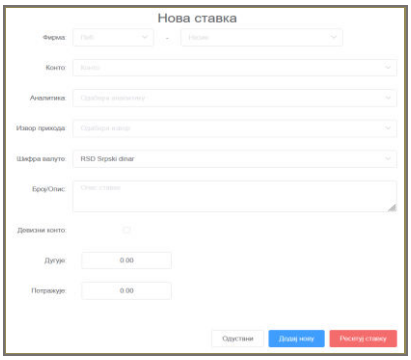
Slika 9. Forma za kreiranje novog naloga za knjiženje

Nakon otvaranja naloga, isti se prikazuje sa osnovnim podacima. Na Sl. 10. možemo videti statusnu liniju naloga, gde postoji status naloga: otvoren ili zatvoren. Samo na otvorenom nalogu je moguće vršiti izmene. Pored statusa naloga, prikazuju se informacije o finansijskom statusu naloga: duguje, potražuje i saldo, a nudi se i opcija izvoza naloga u Word dokument. U okviru statusne linije naloga za knjiženje postoji opcija brisanja istog. Brisanje je moguće samo ukoliko nema stavki na nalogu.



Slika 10. Statusna linija naloga za knjiženje

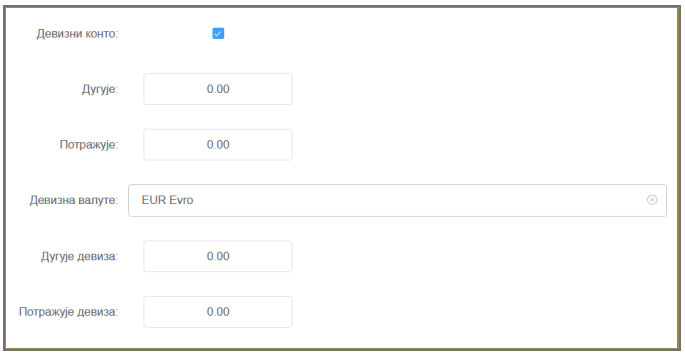
Unos stavki naloga je omogućen tako da svi operatori mogu istovremeno da unose podatke na već postojeći nalog. Operater iz svakog odseka unosi svoje početno stanje i na osnovu njega kreće u dalje knjiženje. Unos nove stavke se realizuje preko forme na Sl. 11.



Slika 11. Dodavanje stavke naloga

Prilikom kreiranja nove stavke naloga postoji mogućnost odabira firme za koji će biti vezana stavka. Ukoliko stavka nije vezana za firmu polje ostaje prazno. Ovde je firmu moguće pretraživati u okviru šifarnika firmi. Takođe je potrebno odabrati konto za koji će stavka naloga biti vezana. Izbor analitike je vezan za ranije odabran konto. Nakon odabranog konta moguće je pridružiti samo analitike koje su otvorene na

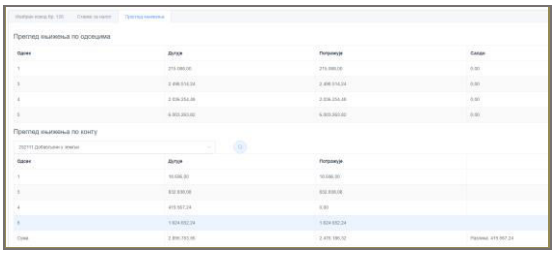
tom kontu. Izvor prihoda se bira na osnovu šifarnika izvora prihoda. Šifra valute [15] ima podrazumevanu vrednost (RSD), jer se u praktičnoj primeni koristi navedena valuta za najveći broj stavki. Opis stavke predstavlja tekstualno polje gde se može dodatno opisati stavka naloga. Nakon toga se unose vrednosti u polja duguje odnosno potražuje. U aplikaciji sadrži i funkcije knjiženja deviznih naloga tako što se oni vezuju za dinarske naloga. Označavanjem opcije devizni naloga otvaraju se dodatna polja za unos devizne valute, kao i devizne vrednosti naloga. Na Sl. 12. prikazana su polja za unos devizne stavke naloga.



Slika 12. Unos devizne stavke

Nakon dodatih stavki operater može sačuvati sadržaj formiranog naloga. Nad svakom stavkom postoje akcije izmene i brisanja, gde operater može prilikom unosa samostalno ispraviti grešku. Iz razloga što više operatora unosi podatke u sistem, postoji realizovana sinhronizacija prikaza stavki naloga. Na ovaj način se u svakom trenutku može pratiti stanje knjiženog naloga.

U okviru pregleda naloga postoji kontrola knjiženja. Ovo je bio jedan od posebnih zahteva koji značajno ubrzava i olakšava rad operatera. U pomenutom prikazu, rukovodilac računovodstva odseka i rukovodilac računovodstva akademije mogu uraditi brz pregled knjiženja po odsecima kao i pregled knjiženja po kontu. Primer jednog pregleda kontrole knjiženja je dat je na Sl. 13.

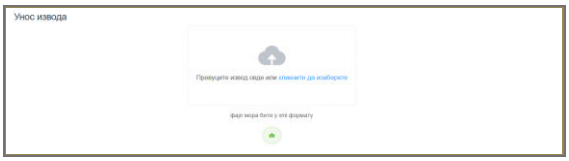


Slika 13. Pregled knjiženja na nalogu

D. *Izvodi*

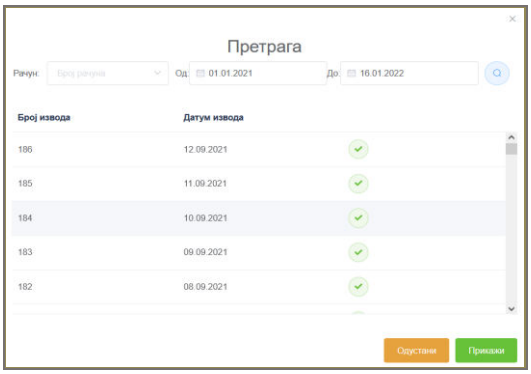
Deo aplikacije koji radi sa izvodima sastoji se iz tri dela: unosa izvoda, prikaza izvoda i pretrage po izvodima na osnovu zadatih kriterijuma.

Unos izvoda se vrši na osnovu XML fajla koji je preuzet sa portala Trezor, [14], gde se mogu naći opis zapisa ovih dokumenata. Preuzeti fajl se dalje unosi u aplikaciju preko forme za unos izvoda. Potrebno je samo odabrati fajl i on će biti najpre parsiran, a zatim i sačuvan u bazi podataka. Forma za unos XML fajla izvoda je prikazana na Sl. 14.



Slika 14. Unos XML fajla izvoda

Pretraga izvoda predstavlja formu gde je potrebno odabrati broj računa kao i opsega datuma u kojima se vrši pretraga izvoda za izabrani račun. Na Sl. 15. prikazana je forma za pretragu izvoda, sa prikazanim rezultatima za odabrani period pretrage.



Slika 15. Pretraga izvoda po računu i datumu

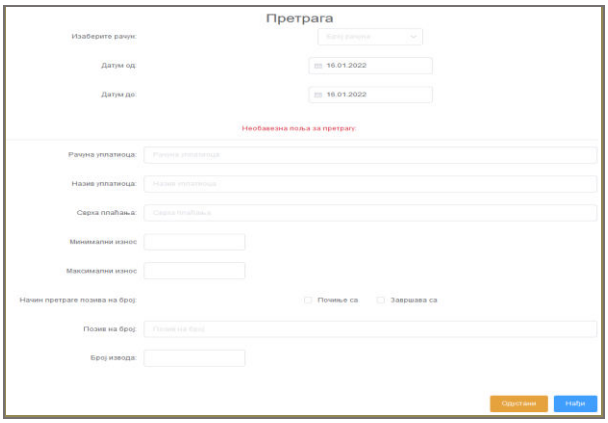
Kao rezultat pretraga najpre se prikazuje broj izvoda i datum izvoda, odakle operater odmah može da pogleda odabrani izvod.

U prikazu izvoda pored podataka sa samog izvoda urađeno je sumiranje i grupisanje stavki na osnovu poziva na broj, tako da se odmah mogu videti zbirni podaci na osnovu odseka. Takođe na samom prikazu izvoda obeležene su stavke koje pripadaju određenom odseku. Na osnovu ovih parametara rukovodilac računovodstva odseka i rukovodilac računovodstva akademije vrše dodatnu kontrolu podataka koji se kasnije koriste u knjiženju. Ova funkcionalnost je prikazana na Sl. 16.

Slika 16. Prikaz izvoda sa trenutnim stanjem uplata studenata

Na osnovu iskazanih potreba realizovana je pretraga stavki izvoda za veći broj izvoda u određenom opsegu datuma. Obavezni podaci za ovu pretragu su broj računa i opseg datuma u kojima se vrši pretraga. Pored ovoga pretragu stavki izvoda

moguće je uraditi po računu uplatioca, nazivu uplatioca (prikazaće se stavke koje sadrže uneti kriterijum u nazivu uplatioca). Takođe je moguće tražiti stavke i po iznosu, tako što se zadaje minimalni i maksimalni iznos u kome stavka treba da bude. Realizovana je i opcija pretrage stavki po pozivu na broj. Ovde je pored tačnog poziva na broj moguće uneti deo istog, tako što se uneti pojam nalazi na početku ili kraju poziva na broj. Forma za pretragu stavki izvoda je prikazana na Sl. 17.



Slika 17. Pretraga stavki izvoda

ZAKLJUČAK

Informacioni sistem za službu računovodstva Akademije projektovan je tako da se realizuju postavljeni zahtevi: (i) vođenje budžetskog knjigovodstva po zakonu i pratećim aktima, (ii) kontinuirano praćenje finansija svakog odseka posebno, (iii) knjiženje uplata studenata i (iv) kontrola knjiženja na nivou odseka odnosno izvoda.

Sada, pošto je prošlo više od pola godine od primene ovog sistema i pošto je u sistemu obrađeno na desetine hiljada podataka, kao i jedan završni račun, možemo da konstatujemo da je realizovani sistem, ovaj koji je prezentovan u radu, uspešno odgovorio na postavljene zahteve i da se može primeniti na druge Akademije odnosno visokoškolske ustanove.

Osim navedenih zahteva, aplikacija je ispunila dodatna očekivanja zaposlenih u Službi računovodstva: pre svega nudi bolje korisničko iskustvo, samostalno održavanje i rad na podacima za knjiženje, posebne opcije u radu sa studentskim uplatama i knjiženjem. Implementirani su sigurnosni mehanizmi rada u kompanijskom okruženju, zaštita pristupa i sigurno čuvanje podataka. Poslovna logika je obuhvatila: rad sa šifarnicima, unosa specifičnih analitika po odsecima, objedinjeni rada sa različitim privilegijama, rad sa izvodima, fakturama, knjiženje i kreiranje neophodnih i posebnih izveštaja. Projekat je rađen timski, koristeći aktuelne softverske arhitekture, alate. U toku rada, tim koji je radio na ovom sistemu, neprekidno prati potrebe korisnika, usavršava funkcije, a u planu su novi delovi informacionog sistema.

LITERATURA

[1] MFIN Republike Srbije, Uprava za trezor. (2021), „Budžetsko računovodstvo i izveštavanje“, sa www.trezor.gov.rs/sr/about/treasuryjobs/budzetskoracunovodstvo/, poslednji pristup 10.01.2002.

[2] Paragraf. (2020), “Uredba o budžetskom računovodstvu”, sa https://www.paragraf.rs/propisi/uredba_o_budzetskom_racunovodstvu.html, poslednji pristup 10.01.2022.

[3] Softek. (2020), Program za budžetsko knjigovodstvo, sa https://www.softek.rs/knjigovodstveni-programi/program-za-budzetsko-knjigovodstvo/, poslednji pristup 10.01.2022.

[4] Obrazovini infomator. (2022), “Računovodstveni časopis specilazovan za budžetske korisnike, https://www.obrazovni.rs/bilten.php, poslednji pristup 10.01.2022.

[5] Abcsoft. (2021), “Knjigovodstveni programi Abcsoft agencije (softver za knjigovodstvo, softver za racunovodstvo)“, sa https://abcsoft.co.rs/knjigovodstveni-programi-softver-poslovni-program-poslovni-softveri-programi-racunovodstvo-robno-materijalno/, poslednji pristup 10.01.2022.

[6] K.H. Judy, “Agile Principles and Ethical Conduct,” 42nd Hawaii International Conference on System Sciences, 2009 **DOI:** 10.1109/HICSS.2009.53

[7] A. Lock, “ASP.NET Core in Action,” Manning Publications, 2021

[8] Vue.js. (2022), „Introduction“, sa https://vuejs.org/, poslednji pristup 10.01.2022.

[9] Element+. (2022), „Design Disciplines“, https://element-plus.org/en-US/guide/design.html, poslednji pristup 10.01.2022.

[10] Fielding Roy, Architectural Styles and the Design of Network-based Software Architectures, PhD Thesis, University of California, Irvin, CA,USA, 2000.

[11] Drupal. (2022), „Usage statistics for RESTful Web Services“, sa https://www.drupal.org/project/usage/restws, poslednji pristup 10.01.2022.

[12] Swagger. (2022), „Documentation From Your API Design“, sa https://swagger.io/solutions/api-documentation/, poslednji pristup 10.01.2022.

[13] Microsoft. (2020), „Fluent API – Configuring and Mapping Properties and Types“, sa https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/fluent/types-and-properties, poslednji pristup 10.01.2022.

[14] MFIN Republike Srbije, Uprava za trezor. (2019), „Specifikacija elektronskih formata za razmenu podataka naloga i izvoda“, https://www.trezor.gov.rs/files/services/espp/ostaledatoteke/Specifikacija%20elektronskih%20formata%20za%20razmenu%20podataka%20naloga%20i%20izvoda.pdf, poslednji pristup 10.01.2022.

[15] Narodna Banka Srbije. (2022), „Šifranik valuta i zemalja“, sa https://www.nbs.rs/SifarnikValutaIZemalja/valute.faces?lang=lat, poslednji pristup 10.01.2022.

ABSTRACT

This paper presents the realization of an application for budget accounting. The application was developed for the needs of several groups of operators of different privileges, working from different locations. The specificity of the application is reflected in special features: groups of operators can define specific posting, while enabling the maintenance of common data needed to form joint reports, there is continuous monitoring of finances of each department, automatic posting of student payments and control of orders by different type of executives.

AN APPLICATION FOR BUDGET ACCOUNTING

Marko Borak, Boško Bogojević, Zoran Ćirović