# Sistem za upravljanje ličnim finansijama

## Članovi tima:

Anastasija Savić SV 7/2020
Katarina Vučić SV29/2020

## Motivacija:

Finansijska stabilnost je danas ključna za svakodnevni život. Dobra organizacija i uvid u finansije može napraviti veliku razliku. Razvijanje sistema koji pomaže ljudima da efikasno upravljaju svojim novcem direktno doprinosti kvalitetnijem i zadovoljnijem životu. Kroz transparentnost i jasan uvid u troškove i prihode, naš sistem može podsticati ljude da donose odgovorne finansijske odluke.

Mladi često ulaze u svet finansija bez dovoljno znanja i iskustva. Naš sistem pruža neophodne informacije i strategije za uspešno upravljanje novcem. Na taj način, pomažemo ljudima da definišu svoje ciljeve i pronađu put do njihovog ostvarenja. To je dodatna motivacija za dobru organizaciju i odgovorno upravljanje novcem.

## Pregled problema:

Postoje mnogobrojna rešenja ove prirode, kao što su na primer: *Mint*, *YNAB (You Need A Budget)* ili *PocketGuard*.

Problem koji se rešava našim projektom je nedostatak efikasnih alata i strategija za upravljanje ličnim finansijama. Ovo može rezultovati finansijskim stresom, nepotrebnim troškovima i nedostatkom štednje za budućnost.

A. *Pregled literature*

Postojeća rešenja obuhvataju aplikacije za praćenje troškova, budžetiranje i planiranje štednje. Međutim, mnogi od tih alata imaju ograničenja u smislu kompleksnosti i dubljeg uvida u finansijske navike korisnika. Nedostaje integracija naprednih tehnologija, kao što su *CEP (Complex Event Processing)*, *backward chaining*, *forward chaining* za analizu i optimizaciju finansijskih odluka.

B. *Nedostaci postojećih rešenja*

Pomenuti postojeći alati pružaju generične savete i strategije, što može biti nedovoljno prilagođeno individualnim potrebama korisnika. Mnogi alati ne pružaju dovoljno dubok uvid u finansijske trendove i obrasce, što može otežati identifikaciju potencijalnih problema i pravljenje efikasnih strategija za njihovo rešavanje.

C. *Prednost našeg rešenja*

- Personalizacija - naš sistem će pružiti visok stepen personalizacije, prilagođavajući savete i strategije korisnicima na osnovu njihovih specifičnih ciljeva, navika i preferencija.
- Analitička dubina - integracija naprednih tehnologija kao što su *CEP*, *backward* i *forward chaining* omogućiće dublji uvid u finansijske trendove i obrasce. Tako ćemo olakšati identifikaciju problema i pravljenje efikasnih strategija za upravljanje novcem.
- Automatizacija - softver će koristiti napredne tehnologije za automatizaciju procesa analize i donošenja odluka. Korisnicima će biti olakšano donošenje finansijskih odluka i postizanje finansijske stabilnosti.
- Savremeni pristup - korz integraciju savremenih tehnologija i pristupa, implementacija će biti korak ispred postojećih rešenja, pružajući korisnicima sveobuhvatnije i efikasnije mogućnosti za upravljanje svojim novcem

## Metodologija rada:

Sistem se sastoji od klijenata. Klijenti koriste sistem za upravljanje svojim ličnim finansijama. Imaju mogućnost praćenja svojih troškova, upravljanju budžetom i pravljenju finansijskih planove.

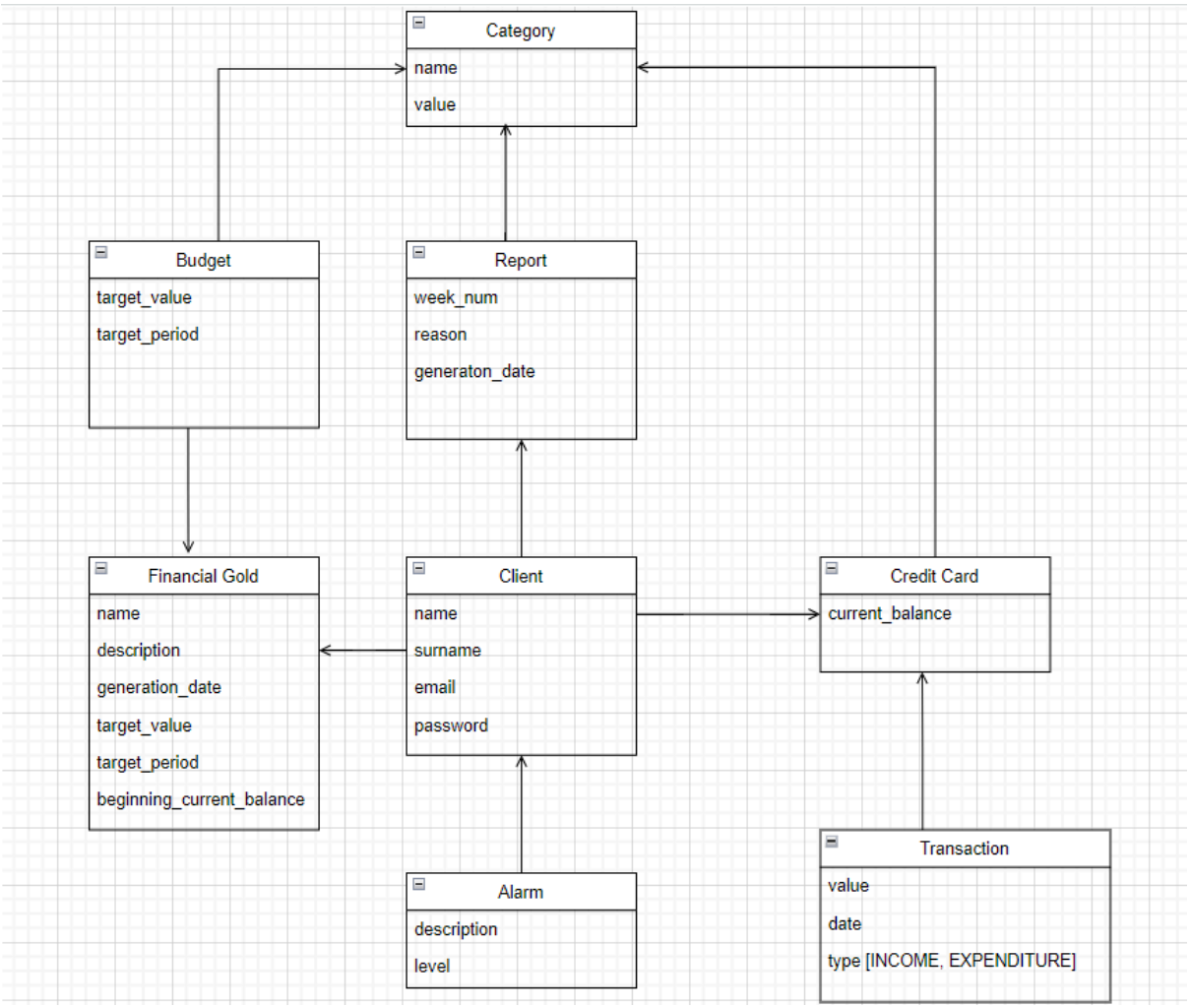
## Ulaz u sistem:

Ulaz u sistem predstavlja istorija prethodnih transakcija (prihodi i rashodi) kao i trenutno stanje na računu. Klijent postavlja svoj finansijski cilj, mesečni budžet, kao i perioda za koji taj cilj želi da ostvari.

## Izlaz iz sistema:

Izlaz iz sistema su preporučene strategije za ostvarivanje trenutnog finansijskog cilja na osnovu prethodnih analiza i unetih parametara. Takođe, omogućen je lakši uvid u troškove i identifikacija potencijalnih problema.

## Baza znanja:



## CEP:

1. nivo

   pravilo: Nakon svake nedelje vrši se provera koliko je korisnik procentualno zadovoljio svoj trenutni finansijski cilj i koliko je trenutno novca ušteđeno. Ušteda se računa kao razlika trenutnog stanja na računu i stanja na računu na početku ostvarivanja cilja.
2. nivo

   pravilo: ako je procenat 100% korisnik je dostigao finansijski cilj - *flag1*

   pravilo: ako je procenat manji od 100% i nedeljni prosek potrošnje je veći od 20% budžeta, detektujemo promenu štednje i nagli skok troškova - *flag2*

   pravilo: ako je procenat manji od 100% i klijent je premašio budžet, ustanovljava se prekoračenje budžeta - *flag3*
3. nivo

   pravilo: za *flag1*, postavi da je rezultat analize povoljan i obavesti klijenta

   pravilo: za *flag2, flag3* postavi da je rezultat analize nepovoljan (dodatno za *flag3* treba dodati da se izvrši *backward chaining*)

## Backward chaining:

U okviru aplikacije ćemo imati dva *backward chaining*-a.

1. *Backward chaining* za obaveštenja o velikim troškova unutar jednog kućanstva. Korisnik ima mogućnost da bude deo kućanstva (*Household*). Kućanstvo ima svoje stablo hijerarhije. Korisnici koji su „iznad" u hijerarhiji dobijaju obaveštenje ukoliko su korisnici „ispod" napravili neke veće troškove. *Backward* prolazi kroz stablo i proverava ko dobija to obaveštenje. Motivacija iza ove funkcionalnosti je mogućnost uvida u finansijske troškove dece u kućanstvu ili uvid u troškove u velikim firmama.

   Korisnici koji su „ispod" u hijerarhiji stabla, ne dobijaju obaveštenje ukoliko su korisnici „iznad" napravili veći trošak.
2. *Backward chaining* ćemo koristiti za analizu troškova. Analiza troškova se pokreće kada korisnik dođe do prekoračenja mesečnog budžeta. Ovaj backward chaining se radi u sklopu forward chaining-a, pa naredni opis daje i detaljnu sliku forward chaining-a.

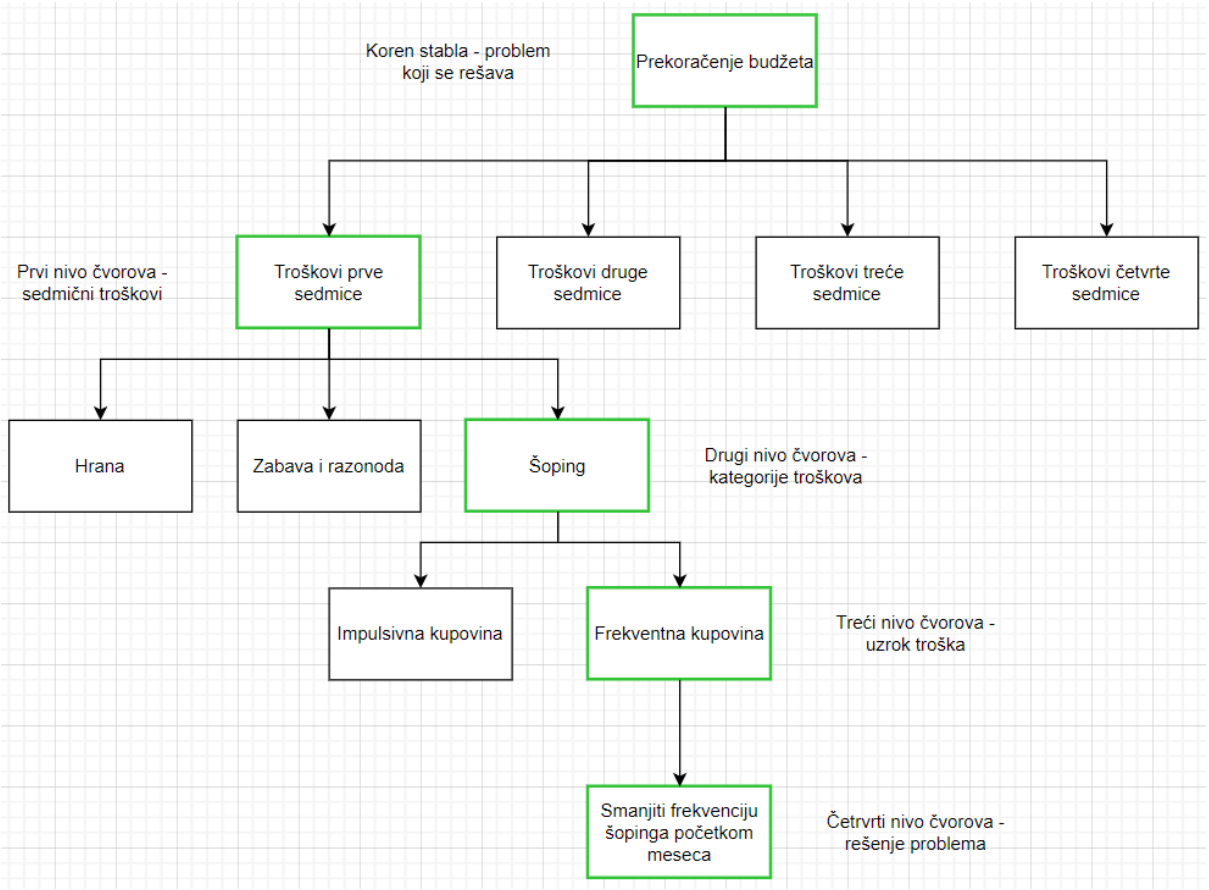
- Nivoi stabla:
    1. Koren stabla - identifikacija problema, korisnik je prekoračio svoj mesečni budžet, što dovodi do finansijskih problema
    2. Analiza troškova - sistem analizira poslednje troškove kako bi identifikovao gde se najviše troši novac i šta su najčešći uzroci prekoračenja budžeta
    3. Identifikacija uzroka - na osnovu analize troškova, sistem identifikuje konkretne uzroke prekoračenja budžeta, kao što su visoki troškovi hrane van kuće ili neplanirani troškovi poput popravki ili medicinskih troškova
    4. Predlog rešenja - na osnovu identifikovanih uzroka, sistem predlaže konkretne korake koje korisnik može preduzeti kako bi rešio problem prekoračenja budžeta. Koraci mogu da budu postavljanje ograničenja na određene kategorije troškova, pravljenje detaljnijeg plana budžeta ili traženje alternativnih opcija za smanjenje troškova

Na osnovu ovoga možemo da zamislimo stablo za *backward chaining*. Na *Slici 1* možemo videti stablo za primer problema mesečnog prekoračenja budžeta. Zelenom bojom je označeno kako smo vršili obilazak. Čvor za obilazak biramo tako što proverimo ukupnu potrošnju te sedmice. Ukoliko je ta potrošnja veća od 20% mesečnog budžeta, nastavljamo obilazak tog čvora. Ukoliko nastavimo obilaziti čvor svaki put ćemo ažurirati *criterion* vrednost objekta (inkrementalno uvećavamo). Promena te vrednosti će izvršiti okidanje određenog pravila. Okinuta pravila će ažurirati vrednosti *exceedingBudgetReason* objekta. Atributi tog objekta su vremenska odrednica *time*, kategorija kupovine *category*, enumeracija vrsta kupovine *purchaseType* (tipovi su impulsivna i frekventna kupovina), novčani iznos *monetaryAmount* i *criterion integer*. Na početku rekurzije

ćemo imati objekat koji će za sve svoje atribute imate *null* vrednosti, a na kraju će atributi tog objekta biti popunjeni.
U rekurzivnoj funkciji ćemo svaki put inkrementalno povećavati *int* promenljivu *criterion* i nakon toga izvršiti rekurziju. Različite vrednosti *criterion*-a će okidati sledeća pravila:

- *criterion* je 0 - sedmični troškovi - poziva se pravilo koje će ažurirati atribute *time* i *monetaryAmount budgetExceeding* objekta. Za vremensku odrednicu će uzeti sedmicu u kojoj je trošak bio veći od 20% ukupnog budžeta.
- *criterion* je 1 - kategorija troškova - poziva se pravilo koje prolazi kroz sve troškove sedmice koja je odrabrana u prethodnom pravilu. Vrši akumulaciju troška po kriterijumu (npr. Hrana, Zabava i razonoda, Šoping, …). Bira kriterijum sa najvećim ukupnim troškom. Prilikom akumulacije sume troška, vrši i brojanje transakcija i ukoliko je broj veći od unapred definisanog broja transakcija, smatraće se frekventnom kupovinom. U suprotnom će se označiti kao impulsivna kupovina.
- *criterion* je 2 - vraćamo criterion na 0 i rekurzivno nastavljam da obilazimo narednu sedmicu

Kada obiđemo jednu sedmicu, okida se pravilo koje dodaje razlog prekoračenja budžeta. Nakon toga *backward* ide dalje i proverava sledeću sedmicu. *Criterion* se vraća na 0. Ukoliko je trošak u drugoj sedmici veći od 20%, ponovo radi istu analizu. Kraj rekurzije je označen kada smo obišli sve sedmice odabranog meseca.
Radi lakšeg razumevanja *Slika 2* prikazuje pseudo kod za *backward chaining*. Ovaj kod je značajno pojednostavljen i podložan promenama.




*Slika 1 - primer stabla za backward chaining*

```
def backward_chaining(obj, week):
        if obj.criterion == 0:
                totalAmount, week = calculate_per_week(obj, week)  // u ovoj funkciji ćemo uvećati i sedmicu
                if totalAmount > (20% of budget):
                        obj.criterion++  // ovde se okinulo pravilo
                if week is not null:  // provera da li je week null, ako jeste obisli smo sve sedmice
                        backward_chaining(obj, week)
        if obj.criterion == 1:
                obj.criterion++  // okida drugo pravilo
                backward_chaining(obj, week)
        if obj.criterion == 2:
                obj.criterion = 0
                backward_chaining(obj, week)
```

*Slika 2 - backward chaining pseudo kod*

## Forward chaining:

U okviru aplikacije ćemo imati dva *forward chaining*-a.

- plan štednje - *forward chaining* ćemo koristiti za generisanje predloga za štednju.
    1. nivo
       pravilo: Ukoliko je korisnik u tekućem mesecu 3 puta bio u kupovini proizvoda određene kategorije, postavi *flag1*
       pravilo: Ukoliko je korisnik u tekućem mesecu 5 puta bio u kupovini proizvoda određene kategorije, postavi *flag2*. Uvek prvo proverava da li ima 5.
    2. nivo
       pravilo: Za *flag1*, proveravamo ukoliko je ukupan iznos je prešao 10% prihoda u tekućem mesecu, postavljamo *flag3*.
       pravilo: Za *flag2*, proveravamo ukoliko je ukupan iznos je prešao 25% prihoda u tekućem mesecu, postavljamo *flag4*.
    3. nivo
       pravilo: za *flag3* generišemo blago upozorenje
       pravilo: za *flag4* generišemo visoko upozorenje

- Ukoliko korisnik u tekućem mesecu bio 5 puta u kupovini proizvoda određene kategorije i iznos prelazi 200 000 dinara, generisaćemo upozorenje višeg nivoa. Na taj način predlažemo korisniku da obrati pažnju na moguću nepotrebnu potrošnju novca. Granice iznosa novca i kvantiteta će biti generisane na osnovu mesečnih prihoda korisnika. Tako npr. kod korisnika koji ima prosečnu platu, alarm o štednji će se ranije aktivirati, nego kod korisnika sa platom iznad proseka. Ukoliko korisnik ima određeni cilj za štednju, u zavisnosti od novčanog iznosa, granica za upozorenja se može menjati.

- *forward chaining* u kombinaciji sa *backward chaining*-om - delimično opisano u naslovu *„Backward chaining”*.
    1. nivo
       Proveravamo da li je trošak za određenu sedmicu veći od 20%.
    2. nivo
       Pronalazimo kategoriju koja je ima najveći trošak te nedelje.
    3. nivo
       Proveravamo da li je to bila impulsivna ili frekventna kupovina. Ukoliko se ta kupovina obavila u 5+ transakcija, smatramo je frekventnom, u suprotnom je impulsivna.
    4. nivo
       Generišemo izveštaj za korisnika u kome se nalazi sedmica, kategorija, ukupan trošak i razlog (impulsivna ili frekventna kupovina).

## Template:

U okviru aplikacije ćemo imati dva template-a.

1. Računanje mesečnog budžeta:

Mesečni budžet se računa na osnovu više unesenih parametara:

- mesečni prihodi
- redovni troškovi - u redovne troškove ubrajamo sve troškove koji se ponavljaju svakog meseca, npr. režije, kirija, namirnice
- finansijski ciljevi - podešavamo parametre vremenska odrednica i iznos koji je potrebno uštediti

Algoritam za računanje mesečnih prihoda ćemo objasniti kroz primer. Korisnik je uneo iznos mesečnih prihoda 120 000. U jednom mesecu potroši 70 000 na redovne troškove. Trenutni mesečni budžet korisnika je 120 000 - 70 000 = 50 000. Nakon toga proveravamo finansijski cilj i primećujemo cilj 200 000 za 5 meseci. To je 200 000 / 5 = 40 000 mesečno. Pošto nam je mesečni budžet 50 000, ovaj finansijski cilj će biti odobren. Konačan budžet za taj mesec je 50 000 - 40 000 = 10 000.

Ukoliko je mesečni iznos finansijskog cilja veći od mesečnog budžeta, korisnik dobija obaveštenje o ovome u vidu izveštaja.

2. Korisnik ima mogućnost da napravi *custom report*. Cilj ove funkcionalnosti je da se korisniku omogući pravljenje beleške (npr. spisak za kupovinu ili opomena da se finansijski odgovornije ponaša). Parametri *Tempate*-a su beleška, datum generisanja izveštaja i klijent koji ga je generisao.