SVEUČILIŠTE U SLAVONSKOM BRODU

Stručni prijediplomski studij Informatika i informacijske tehnologije

SEMINARSKI RAD

Flutter aplikacija za Android okruženje: TROŠKOVNIK

Kristina Briški

Slavonski Brod, 2025.

SVEUČILIŠTE U SLAVONSKOM BRODU

Stručni prijediplomski studij Informatika i informacijske tehnologije

SEMINARSKI RAD

Flutter aplikacija za Android okruženje: TROŠKOVNIK

Kolegij: Mobilne aplikacije

Ime i Prezime: Kristina Briški

Nastavnik: Tomislav Katinić, dipl.inf.

Slavonski Brod, 2025.

SAŽETAK

Sažetak

Seminar se fokusira na razvoj Flutter aplikacije za Android okruženje:Troškovnik, koja omogućuje praćenje troškova korisnika putem jednostavnog sučelja i pohrane podataka u lokalnu SQLite bazu podataka. Aplikacija nudi mogućnost unosa različitih vrsta troškova, uključujući naziv, iznos i mjesec kada je trošak nastao. Jedna od ključnih funkcionalnosti aplikacije je mogućnost generiranja ukupnih troškova po mjesecima, što omogućuje lakše praćenje financija i planiranje budućih izdataka.

U izradi ove aplikacije korištene su tehnologije Flutter i SQLite, što omogućuje izradu nativne aplikacije za Android uređaje. Aplikacija je dizajnirana s ciljem jednostavnosti korištenja, intuitivnim sučeljem te niskim zahtjevima za resurse, što je čini pogodnom za širu primjenu među korisnicima svih dobnih skupina.

Ključne riječi: mobilna aplikacija, Flutter, SQLite, upravljanje osobnim financijama, troškovi.

SADRŽAJ

1. UVOD

U današnjem digitalnom dobu, upravljanje osobnim financijama postalo je ključna vještina za svakog pojedinca. Troškovi koji nastaju svakodnevnim životom, bilo da se radi o hrani, prijevozu, rekreaciji ili drugim potrebama, često su teško praćeni bez odgovarajućeg sustava. Zbog toga je razvoj mobilnih aplikacija za praćenje i upravljanje troškovima postao vrlo popularan. Jedna od takvih aplikacija je aplikacija *Troškovnik*, koja omogućuje korisnicima jednostavno praćenje svih svojih financijskih izdataka na mobilnim uređajima.

Aplikacija *Troškovnik* omogućuje korisnicima unos troškova (naziv i iznos), prikaz ukupnih troškova po mjesecima, te daje mogućnost detaljnog pregleda svih unesenih troškova u formi popisa. S ciljem omogućavanja jednostavnog praćenja financija, aplikacija koristi SQLite bazu podataka za pohranu svih unesenih podataka. S obzirom na jednostavnost korištenja, ova aplikacija pruža korisnicima učinkovit alat za praćenje svojih financija.

2. TEORIJSKI DIO

Prilikom izrade Flutter aplikacija za Android okruženje: Troškovnik, vodila sam se mišlju kako što jednostavnije organizirati i pratiti financijske troškove tijekom godine kroz mjesece. Ideja je bila omogućiti unos troška, njegov naziv i iznos, mjesec kada se trošak dogodio i brisanje troška. Trošak bi se trebao spremiti u bazu podatka na temelju koje se može vidjeti mjesečna potrošnja i sveukupna potrošnja.

Aplikacija koristi SQLite kao sustav za upravljanje podacima, čime se omogućuje pohrana i dohvat troškova unesenih u aplikaciji. Korisnik može unositi troškove koji se automatski spremaju u bazu podataka, a pomoću funkcionalnosti za filtriranje po mjesecima može dobiti uvid u ukupne troškove po svakom mjesecu i sve troškove ukupno. Kod razvoja aplikacije korišten je Flutter framework, koji omogućuje brzo i učinkovito stvaranje mobilnih aplikacija s niskim troškovima razvoja. Flutter aplikacija se izrađuje unutar Android studio, prilikom čega je potrebno koristiti dodatak koji će omogućiti izradu Flutter aplikacije unutar razvojnog okruženja, za izradu aplikacije koristit će se Dart programski jezik u VS Code uređivaču, dok će se aplikacija izvršavati uz pomoć emulatora za Android.

Flutter i Dart zajedno omogućuju razvoj vrlo brzih, nativnih aplikacija za više platforma, a kod u Flutteru uglavnom se sastoji od widgeta za prikazivanje UI-a i logike koja upravlja podacima, unosima i drugim funkcijama koje aplikacija obavlja. Flutter se koristi za izradu korisničkog sučelja, dok Dart nudi snažne mogućnosti za poslovnu logiku, upravljanje podacima i asinkrono programiranje. Glavni dijelovi Fluttera:

- **Widgets**: Glavni gradivni blok Flutter aplikacija. Svaka komponenta na ekranu, poput tekstova, slika, tipki, okvira, itd., je widget. Flutter je baziran na "everything is a widget" principu, što znači da čak i layouti (kao što su redci i stupci) i animacije također mogu biti predstavljeni kao widgeti.
- **Stateful & Stateless Widgets**:
    1. **StatefulWidget**: Widget koji može promijeniti svoj izgled tijekom vremena (npr. na temelju korisničkog unosa, animacija).
    2. **StatelessWidget**: Widget koji ne mijenja svoj izgled nakon što je inicijaliziran (npr. statički tekst).

- **Hot Reload**: Glavna prednost Fluttera je "hot reload" funkcionalnost koja omogućava programerima da vide promjene u kodu u stvarnom vremenu, bez potrebe za ponovnim pokretanjem aplikacije.
- **Material Design**: Flutter uključuje gotovu podršku za **Material Design** (Google-ov dizajn smjernica), ali također mogu se koristiti i **Cupertino** widgeti za izgled aplikacija specifičan za iOS.
- **Navigation & Routing**: Flutter omogućava jednostavno upravljanje navigacijom između različitih ekrana pomoću Navigator widgeta i routes. Mogu se koristiti različite strategije za navigaciju, uključujući i "push" i "pop" ekrana.
- **Flutter Engine**: Nisko-nivo komponenta koja omogućava izvršavanje Flutter aplikacija, kao i renderiranje grafike, korisničkog sučelja, i mnoge druge stvari kao što su rad sa shaderima i fontovima.

Dart je programski jezik koji se koristi za razvoj u Flutteru. Dart je objektno-orijentirani, kompajlirani jezik, što znači da se aplikacije u njemu kompajliraju u nativni kod (na Androidu i iOS-u), čime se postiže vrlo dobra brzina. Glavni dijelovi Dart jezika:

- **Varijable i Tipovi Podataka**: Dart je tipizirani jezik, ali omogućava dinamičko tipiziranje, podržava null-safety, što znači da se varijablama može dodijeliti ili vrijednost ili null, ali uz jasnu kontrolu. Osnovni tipovi podataka: int, double, String, bool, List, Map, Set, itd.
- **Funkcije**: koriste za definiranje ponašanja u aplikaciji. Sintaksa funkcija je jednostavna i može imati parametre i povratne vrijednosti.
- **Classes (Klase)**: Dart je objektno-orijentirani jezik, što znači da koristi **klase** za definiranje objekata. Klasa može imati **atribute** (varijable) i **metode** (funkcije).
- **Async & Await**: Dart podržava asinhrono programiranje s **Future**, **async**, i **await**, što je korisno ako se radi s neblokirajućim zadacima, kao što su mrežni zahtjevi ili rad s bazama podataka.
- **Collections**: Dart podržava kolekcije kao što su **List**, **Set**, i **Map.**
- **Error Handling**: Dart koristi **try-catch** blokove za hvatanje i upravljanje iznimkama.
- **Packages**: Dart ekosustav nudi mnoge pakete za različite funkcionalnosti, kao što su mrežni zahtjevi, rad s bazama podataka, i animacije. Paketi se dodaju pomoću **pubspec.yaml** datoteke.
- **Streams**: Dart podržava rad s **Streamovima**, što je korisno za asinkrono slanje podataka ili za rad s događanjima.

3. TEHNIČKA DOKUMENTACIJA PROJEKTA

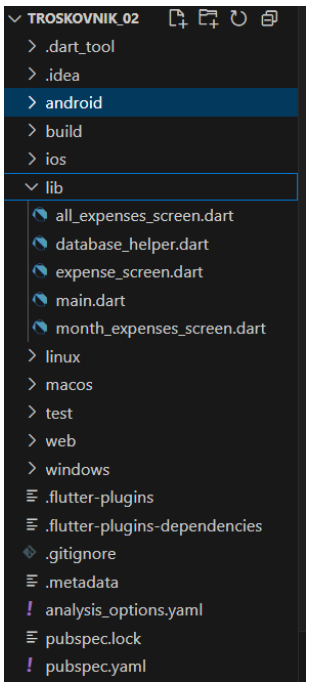
Prilikom izrade Flutter aplikacije korištene su vanjske biblioteke koje omogućuju jednostavniji dohvat i obradu podataka. Za upravljanje vanjskim bibliotekama koristi se YAML sustav skripti. Dodavanje vanjskih biblioteka unutar Flutter projekta je jednostavna operacija i sami proces provodi se na sljedeći način. Prvi korak je taj da se unutar dependencies sekcije dodaje ime i broj verzije željene biblioteke. Nakon što se naprave izmjene, potrebno je pokrenuti naredbu **flutter pub get i flutter pub upgrade** kojom se pokreće preuzimanje i implementacija nove biblioteke unutar projekta i na kraju **flutter run** da se pokrene aplikacija.

```yaml
dependencies:
  flutter:
    sdk: flutter
  sqflite: ^2.4.1
  path: ^1.8.0

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8

dev_dependencies:
  flutter_test:
    sdk: flutter
```

Slika 1: Prikaz biblioteka koje su korištene prilikom izrade Flutter aplikacije

Što je i vidljivo na slici 2, projekt je podijeljen u pet manjih dijelova:

1. **main.dart** - pokreće aplikaciju i navigira između ekrana
2. **database_helper.dart** - bitna za interakciju s bazom podataka, u njemu su definirane sve metode za unos, dohvat i manipulaciju podacima u bazi
3. **expense_screen.dart** - unos troškova, početni zaslon
4. **all_expenses_screen.dart** - prikaz svih troškova, drugi zaslon
5. **month_expenses_screen.dart** - prikaz ukupnih troškova po mjesecima, treći zaslon



Slika 2: Prikaz strukture aplikacije

3.1. Tehničke karakteristike alata i sklopovlja

Za izradu aplikacije korišteno je osobno računalo i razvojni alati. Popis tehničke karakteristike alata i sklopovlja bit će vidljivo u tablici, a kratko je objašnjen svaki od korištenih razvojnih alata: Android Studio, Flutter, Visual Studio Code i emulator.

1. **Android Studio** je službeno razvojno okruženje (IDE – Integrated Development Environment) za izradu Android aplikacija. Razvio ga je Google, a temelji se na IntelliJ IDEA platformi tvrtke JetBrains. To je moćan alat koji omogućuje programerima da razvijaju, testiraju i debagiraju Android aplikacije za različite uređaje i verzije Android operativnog sustava, optimizaciju performansi aplikacija i upravljanje bazama podataka unutar aplikacija (npr. SQLite).
2. **Flutter** je open-source SDK (Software Development Kit) koji je razvila tvrtka Google za izradu multiplatformskih aplikacija iz jedne baze koda. Pomoću Fluttera mogu se razvijati mobilne aplikacije (Android i iOS), web aplikacije, desktop aplikacije (Windows, macOS, Linux), pa čak i aplikacije za embedded uređaje.
3. **Visual Studio Code (VS Code)** besplatan je i lagan uređivač koda koji je razvio Microsoft. Omogućuje programerima pisanje, uređivanje i debuggiranje koda u različitim programskim jezicima. Vrlo je popularan zbog brzine i fleksibilnosti, a koristi se uz ekstenzije za Flutter i Dart.

4. **Emulator** je virtualni uređaj koji simulira pravi Android telefon ili tablet na računalu. Omogućuje testiranje i razvoj Android aplikacija bez potrebe za fizičkim uređajem. Emulator može biti sporiji od pravog uređaja, ali je odličan alat za razvoj i testiranje aplikacija u različitim uvjetima!

Računalo		
	Model	HP 15s-eq2011nm
	OS	Windows 10 Pro
	Procesor	AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz
	Memorija	16 GB
Razvojni alati		
	Android Studio	Ladybug 2024.2.1 Patch 1
	Flutter	Verzija: 3.24.3, channel stable, Dart 3.5.3, DevTools 2.37.3
	Visual Studio Code	1.97.2
	Emulator	Pixel 3a API 34, Android 14.0, UpsideDownCake, x86_64
Ostali alati (uređivač teksta, alati za izradu skica i dijagrama, …)		
	Word dokument	Uređivač teksta
	Snipping tool	Izrada slika/snimki zaslona ekrana aplikacije i VS code

3.2. Opis klasa, atributa i metoda

Klase koje su korištene u aplikaciji su one koje sadrže logiku za svaki zaslon, kao što su **MyApp**, **ExpenseScreen**, **AllExpensesScreen**, **MonthExpensesScreen** i **DatabaseHelper** za rad s bazom podataka. Atributi su varijable koje pohranjuju stanje, poput **TextEditingController** za unos, varijable za pohranu odabranih mjeseci, ili mape i liste za pohranu podataka iz baze. Metode su funkcije koje izvode specifične zadatke, poput dodavanja novog troška (**_addExpense**), dohvaćanja podataka iz baze (**_loadExpenses**) i prikazivanja tih podataka na ekranu.

Prikaz i opis korištenih klasa, atributa i metoda u projektu su podijeljene u pet dijelova: main.dart, database_helper.dart, expense_screen.dart, all_expenses_screen.dart, month_expenses_screen.dart:

1.main.dart

Klasa:

MyApp: Glavna klasa aplikacije koja koristi MaterialApp widget za osnovno postavljanje aplikacije, definiranje teme i početnog ekrana.

Atributi:

Nema atributa u main.dart jer se aplikacija sastoji samo od MyApp widgeta bez ikakvog unutrašnjeg stanja koje treba pohraniti.

Metode:

build(): metoda koja definira izgled aplikacije.

runApp(): metoda koja pokreće aplikaciju.

2. database_helper.dart

Klasa:

DatabaseHelper: Pomoćna klasa za interakciju s SQLite bazom podataka. Služi za unos, dohvaćanje i brisanje podataka (troškova).

Atributi:

Nema klasičnih atributa u smislu varijabli koje se pohranjuju unutar instance objekta. Umjesto toga, koristi se metoda **_getDatabase()** koja se poziva za svaki zahtjev prema bazi podataka.

Metode:

_getDatabase() - Dohvaća bazu podataka.

insertExpense(**name, amount, month):** Umeće novi trošak u bazu podataka.

getExpenses(): Dohvaća sve troškove iz baze.

getExpensesByMonth(month): Dohvaća troškove za određeni mjesec.

getTotalExpenseForMonth(month): Dohvaća ukupni zbroj troškova za određeni mjesec.

deleteExpense(id): Briše trošak iz baze prema njegovom id.

3.expense_screen.dart

Klasa:

ExpenseScreen: zaslon za unos novih troškova gdje korisnik može unijeti naziv, iznos i odabrati mjesec.

Atributi:

_nazivController: TextEditingController - za unos naziva troška.

_iznosController: TextEditingController - za unos iznosa troška.

selectedMonth: String? - Varijabla koja pohranjuje odabrani mjesec.

months: List <String> - Lista mjeseci koja se koristi u DropdownButton za odabir mjeseca.

Metode:

_addExpense(): Metoda koja dodaje novi trošak u bazu podataka ako su svi podaci uneseni.

build (BuildContext context): Vraća widgete za unos podataka i navigaciju na druge ekrane.

4. all_expenses_screen.dart

Klasa:

AllExpensesScreen: zaslon koji prikazuje sve unesene troškove.

_AllExpensesScreenState: Privatna klasa koja upravlja stanjem zaslona i logikom za učitavanje i brisanje troškova

Atributi:

expenses: **List<Map<String, dynamic>>** - Lista troškova koja se prikazuje na ekranu.

Metode:

createState(): Vraća instancu _AllExpensesScreenState.

_loadExpenses(): Dohvaća sve troškove iz baze podataka i ažurira listu.

initState(): Poziva se prilikom inicijalizacije ekrana. U ovom slučaju, koristi se za učitavanje svih troškova iz baze podataka.

_deleteExpense(int id): Asinkrona metoda koja omogućava brisanje troška iz baze podataka.

build(BuildContext context): Standardna metoda koja se koristi za izgradnju korisničkog sučelja.

5. month_expenses_screen.dart

Klasa:

MonthExpensesScreen: zaslon koji prikazuje ukupne troškove po mjesecima i ukupan iznos troškova.

_MonthExpensesScreenState: Privatna klasa koja upravlja stanjem ekrana i logikom za učitavanje i prikaz mjesečnih troškova).

Atributi:

monthlyTotals: **Map<String, double>** - Mapa koja pohranjuje ukupne troškove za svaki mjesec.

Metode:

createState(): Vraća instancu _MonthExpensesScreenState.

initState(): Inicijalizira stanje zaslona.

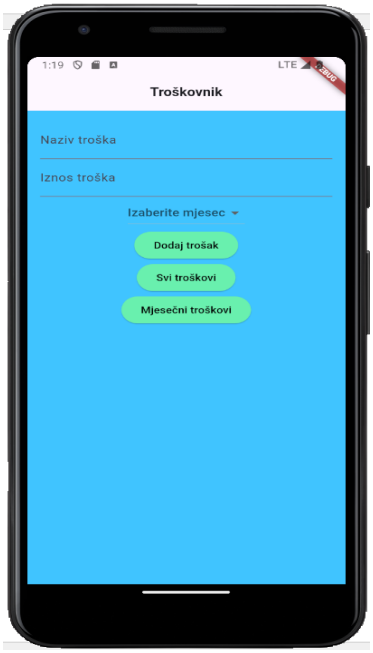
_loadMonthlyTotals(): Dohvaća ukupne troškove za svaki mjesec iz baze podataka i sprema ih u monthlyTotals.

build(): Vraća widget koji prikazuje popis mjeseci s ukupnim troškovima.

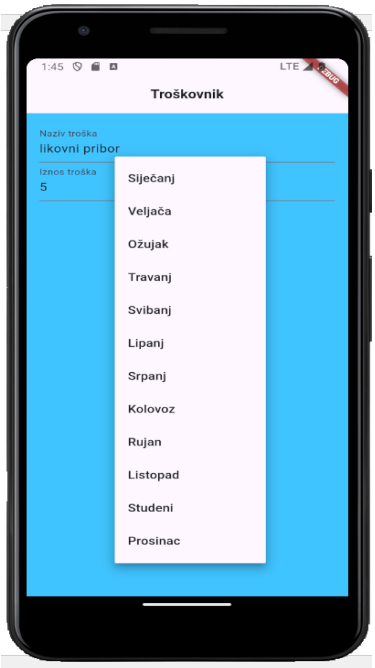
3.3.Korisničke upute

Aplikacija Troškovnik je vrlo jednostavna za korištenje, funkcionalna je i praktična. Sastoji se od tri zaslona s mogućnošću navigacije između njih. SQLite baza se koristi za upravljanje podacima unutar projekta (upis, dohvat i brisanje unesenog troška).

Na prvom zaslonu (expense_screen.dart) vidi se početni zaslon koji omogućuje navigaciju između preostala dva zaslona pomoću dva gumba: **Svi troškovi** i **Mjesečni troškovi.** Nalazi se mogućnost unosa troška pomoću gumba **Dodaj trošak** nakon upisa naziva troška, iznosa troška i mjeseca u koje je trošak napravljen. Naziv i iznos troška korisnik upisuje sam, a mjesec izabire u padajućem izborniku. Trošak se neće spremiti u bazu ako nisu upisana sva tri podatka (naziv, iznos, mjesec). Kada je trošak spremljen, polja za unos novog troška ponovno su prazna i imaju početni tekst (Naziv troška i Iznos troška).



Slika 3: Početni zaslon

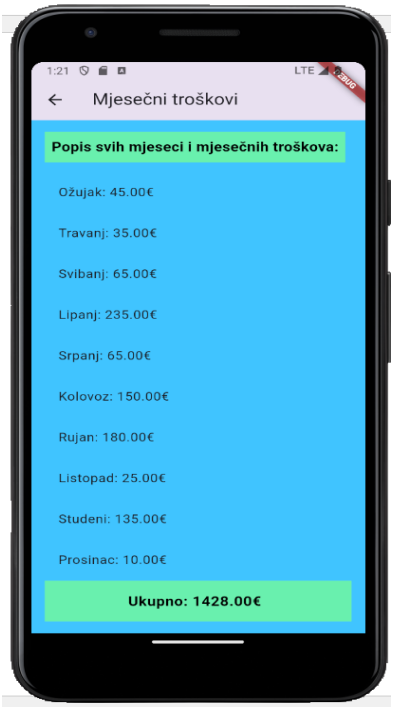


Slika 4: Unos troška, padajući izbornik za unos mjeseca

Drugi zaslon (all_expenses_screen.dart) nudi uvid u sve upisane troškove. Vidi se popis upisanih troškova jednih ispod drugih s nazivom troška, iznosom troška i mjesecom u kojem je napravljen. Na kraju popisa vidi se ukupan iznos svih troškova. Pokraj svakog troška postoji mogućnost brisanja troška, ikona kanta za smeće, ako je krivo unesen. Nakon brisanja podaci se ažuriraju u bazi pa je to vidljivo i na ovom zaslonu **Svi troškovi** i na zaslonu **Mjesečni troškovi**. Postoji mogućnost vraćanja na početni ekran klikom na strelicu u gornjem lijevom kutu.

Slika 5: Drugi zaslon - Svi troškovi

Treći zaslon (month_expense_screen.dart) nudi uvid u upisane troškove organizirane po mjesecima od siječnja do prosinca. Vidi se popis mjeseci od siječnja do prosinca i ukupan iznos troška za svaki mjesec. Na kraju popisa, vidi se i ukupan trošak za sve navedene mjesece. Postoji mogućnost vraćanja na početni ekran klikom na strelicu u gornjem lijevom kutu.



Slika 6: Treći zaslon – Mjesečni troškovi

4. ZAKLJUČAK

Aplikacija *Troškovnik* predstavlja jednostavan i učinkovit alat za upravljanje osobnim financijama, pružajući korisnicima jasnu sliku o njihovim troškovima i omogućujući im lakše praćenje troškova. Uz jednostavno sučelje, fleksibilnost unosa podataka (mogućnost brisanja troška), mogućnost pregleda i analize troškova po mjesecima te ukupan zbroj svih troškova, aplikacija korisnicima nudi kvalitetno praćenje njihovih financija. Razvoj ove aplikacije pokazuje prednosti korištenja mobilnih tehnologija za svakodnevne zadatke i upravljanje osobnim financijama. Aplikacija bi se mogla nadograditi dodavanje mogućnosti uređivanja troškova, mogućnost organiziranja troškova po godinama.

LITERATURA

[1] Flutter razvojni okvir (engl. framework) za razvoj aplikacija za različite platforme https://docs.flutter.dev/get-started/install (preuzeto listopad 2024.)

[2] Android Studio Ladybug | 2024.2.1 Patch 1 (preuzeto listopad 2024.)

[3] Visual Studio Code: 1.97.2 (pristupljeno veljača 2024.)

[4] https://urn.nsk.hr/urn:nbn:hr:211:290644 (pristupljeno veljača 2024.)

[5] https://appify.digital/blog/flutter-app-development (pristupljeno veljača 2024.)

[6] https://dart.dev/language (pristupljeno veljača 2024.)