UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Funkcionalno programiranje

Zadaća 1

Tuzla, decembar/prosinac 2025.

Zadatak 1

Napisati aplikaciju za menadžment kino projekcija i rezervacija karata koristeći SQLite bazu podataka. Program treba da podržava operacije za **administratora** i **korisnika**.

Administratorske operacije:

- Dodavanje filma
- Izmjena filma
- Brisanje filma
- Kreiranje (zakazivanje) projekcije za film
- Izmjena projekcije (sala, vrijeme početka, cijena karte)
- Otkazivanje projekcije
- Pregled svih rezervacija za odabranu projekciju

Korisničke operacije:

- Pregled dostupnih projekcija (po datumu ili po nazivu filma)
- Prikaz slobodnih/zauzetih sjedišta za odabranu projekciju
- Kreiranje rezervacije (rezervacija tačno jednog sjedišta)
- Otkazivanje rezervacije (po ID-u rezervacije)
- Pretraga rezervacija po imenu korisnika

Postavka sistema

Kino ima **tačno dvije sale** koje moraju postojati u sistemu odmah nakon pokretanja (seed u bazi ili automatsko kreiranje ako ne postoje):

- Sala 1: **30** sjedišta
- Sala 2: **50** sjedišta

Brojevi sjedišta su numerisani od **1** do **N** (N je broj sjedišta sale).

Funkcionalni zahtjevi

Potrebno je osmisliti **model podataka** i **SQL šemu baze** (tabele/kolone/ključevi) tako da pokrije sljedeće funkcionalnosti.

Administratorske operacije:

- Upravljanje filmovima (dodavanje, izmjena, brisanje)
- Zakazivanje projekcije (odabir filma, odabir sale 1 ili 2, datum i vrijeme početka, cijena karte)
- Otkazivanje projekcije
- Pregled rezervacija za odabranu projekciju (uključujući koja su sjedišta zauzeta)

Korisničke operacije:

- Pregled projekcija za odabrani datum
- Pretraga projekcija po nazivu filma (substring)
- Prikaz stanja sjedišta za odabranu projekciju (slobodno/zauzeto za sjedišta 1..N)
- Kreiranje rezervacije za tačno jedno sjedište (projekcija + broj sjedišta + ime korisnika)
- Otkazivanje rezervacije po ID-u

Pravila sistema

Nije dozvoljeno da postoje **dvije aktivne rezervacije** za isto sjedište na istoj projekciji.

Rezervacija je validna samo ako je broj sjedišta u opsegu **1..N**, gdje je N broj sjedišta sale u kojoj se projekcija održava.

Otkazivanje rezervacije mora **osloboditi sjedište** (nakon otkaza mora biti moguće ponovo rezervisati isto sjedište).

Brisanje filma treba biti definisano jasnim pravilom. Primjer: nije dozvoljeno obrisati film ako postoje zakazane projekcije za taj film.

Aplikacija treba imati dvije “ulazne tačke” u meniju: **Admin** i **Korisnik**. Admin dio može biti zaštićen jednostavnom šifrom/PIN-om (konstanta u kodu).

Persistence (SQLite)

Sve operacije moraju čitati/pisati stanje u SQLite bazu tako da se stanje nakon restart-a programa **ne gubi**.

Potrebno je:

- Kreirati bazu i tabele pri pokretanju (ako ne postoje)
- Inicijalizovati dvije sale (1 i 2) sa brojem sjedišta 30 i 50 (ako ne postoje) ●

Implementirati upite za dodavanje/izmjenu/brisanje i pretrage koje su potrebne za tražene menije

Napomena: način na koji ćete modelirati podatke (tipovi u F# i tabele u bazi) je dio zadatka. Extras

1. Dodati opciju “prikaz sale” kao tekstualnu mapu gdje je slobodno sjedište označeno sa `.` a zauzeto sa X.
2. Dodati jednostavno logovanje akcija (npr. u tabelu `AuditLog` ili u tekstualni fajl).
3. Implementirati da pretraga projekcija podržava kombinovanje filtera (datum + substring naziva filma).

Zadatak 2

Uvod

Instalacija neophodnih alata:

- `fable`: Ovo je F# to JavaScript (primarno) compiler. Omogućava pisanje F# koda koji se kompajlira u JavaScript, što je korisno za razvoj web aplikacija. ● `femto`: Ovaj alat automatski upravlja NuGet i `npm` paketima u F# projektima koji koriste Fable. Pomaže u održavanju sinhronizacije između .NET i JavaScript ekosistema.

```
dotnet tool install -g fable
dotnet tool install -g femto
```

Kreiranje Novog F# Projekta

Nakon globalne instalacije fable i femto programa, potrebno je kreirati novi projekat.
Inicijalizacija Projekta: kreiranje novog direktorija i inicijalizacija F# konzolne aplikacije:

```
mkdir new_proj
cd new_proj
dotnet new console -lang f# -o app
```

Inicijalizacija Vite: Vite je alat za razvoj frontend aplikacija, koji omogućava brzo učitavanje modula.

```
npm init vite vproj
```

Odabrati **`Vanilla`** i **`JavaScript`** kako bi se kreirao osnovni frontend projekat bez dodatnih biblioteka ili framework-a.

Kompajliranje Aplikacije: Kompajliranje F# aplikacije u JavaScript i priprema za web:

```
fable app -o build
cp vproj/index.html build/.
```

Također je potrebno izmjeniti **`build/index.html`** tako da učitava **`/Program.js`**

`Instalacija Dependencies-a:` Instaliranje potrebnih paketa, uključujući **`vite`**:

```
cp vproj/package.json .
npm install
```

U **`package.json`** file-u izmjeniti vrijednost pod **`scripts.dev`** json ključem: `"dev": "fable watch app -o build & vite dev build &"`

Pokretanje Aplikacije: Pokretanje razvojnog servera:

```
npm run dev
```

Nakon pokretanja aplikacije, otvoriti u browser-u adresu koju je vite ispisao, npr: `VITE v5.0.10 ready in 152 ms`

```
➜ Local: http://localhost:5173/
```

U browseru otvoriti **`localhost:5173`** adresu, i u konzoli (`Desni klik->Inspect->Console`) bi trebali imati ispis “Hello from F#”.

Instalacija Elmish i Feliz-a

`Elmish:` Biblioteka koja omogućava upotrebu Elm arhitekture unutar F# aplikacija. Unutar foldera projekta (“proj1”):

```
femto app/ install Elmish
```

`Feliz:` Moderni DSL za izgradnju React UI-a u F#. Nudi intuitivniji i deklarativniji način za izgradnju korisničkog interfejsa.

```
femto app/ install Feliz
```

`Dodavanje Elmish.React paketa:` Integracija Elmish-a sa React-om. `dotnet add app/ package Fable.Elmish.React`

Izrada zadatka

Nakon ovoga, projekt je setup-ovan i potrebno je pristupiti izradi zadatka primjenjujući koncepte Elmish Framework-a i Feliz biblioteke u svom F# kodu. Ovo uključuje definisanje modela, update funkcija i prikaza koristeći Elmish u F#.

Zadatak

Potrebno ja napisati web aplikaciju koja simulira ručni kalkulator. Aplikacija kada se starta prikazuje tipke i operacije kalkulatora. Podržane operacije:

- Sabiranje
- Oduzimanje
- Množenje

- Dijeljenje
- Brisanje

Iznad cifara i operatora potrebno je rezervisati prostor za prikazivanje brojeva i

rezultata. Stanja Aplikacije

Definisati 3 stanja u kojima se kalkulator može nalaziti:

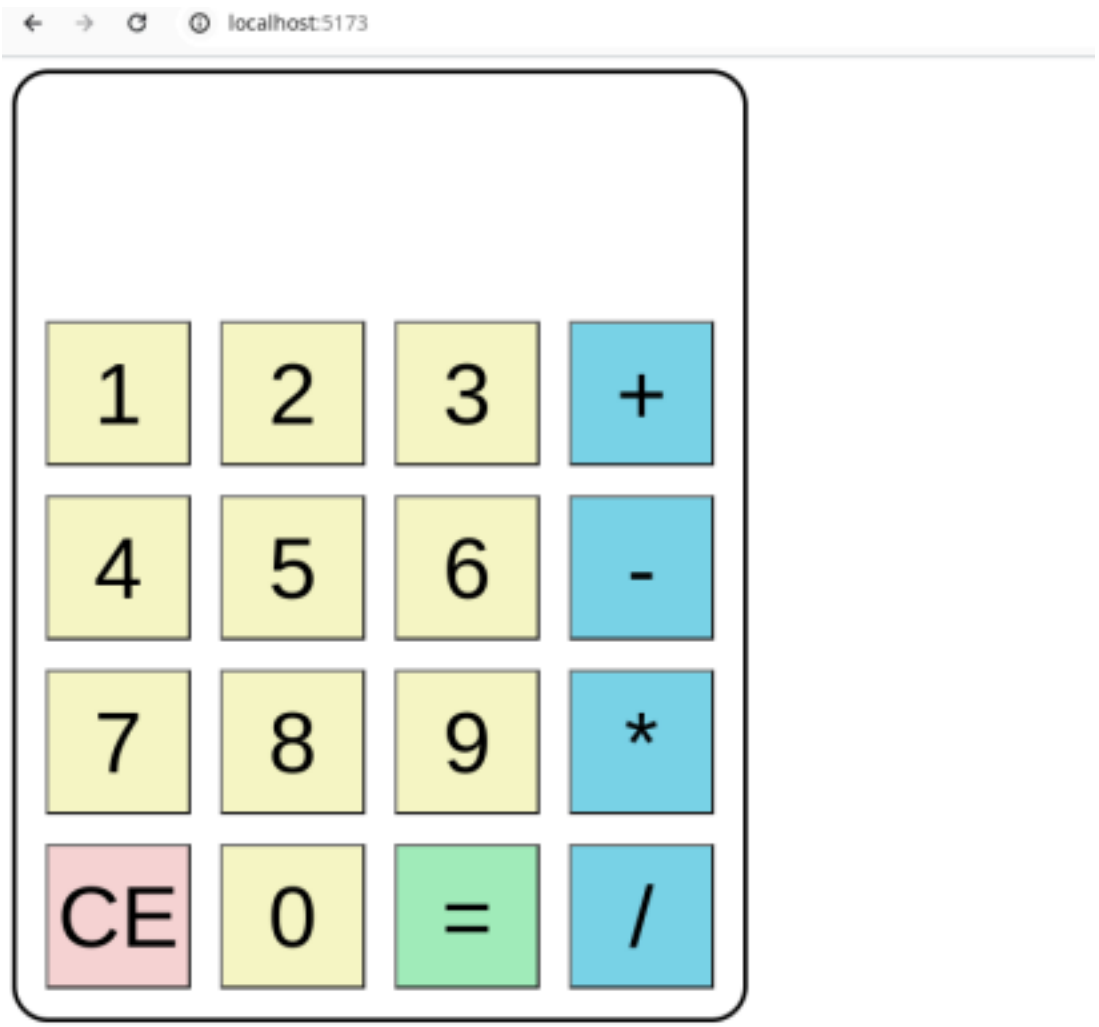
- Stanje 1 - korisnik unosi prvi broj. Broj se pojavljuje na vrhu kalkulatora, poravnat' na desnu stranu. Svaka cifra se dodaje na kraj trenutnog broja. Kalkulator prelazi u sljedeće stanje kada korisnik klikne na neki operator. Ukoliko korisnik klikne na tipku za brisanje, cijeli broj treba da nestane i kalkulator ostaje u stanju 1. Ukoliko korisnik klikne na tipku =, kalkulator treba da ignoriše input korisnika - što dalje znači da unos brojeva treba da nastavi gdje je stao, ili eventualno da operator prebaci kalkulator u sljedeće stanje. Ukoliko prije unešenih cifara korisnik unese operator, kalkulator treba da ignoriše input korisnika. Korisnik, treba da može nakon toga unositi cifre kao da se ništa nije desilo.
- Stanje 2 - do ovog stanja moguće je doći samo ukoliko je korisnik unio prvi broj i stisnuo neki od operatora. Stanje 2 predstavlja međustanje prije unosa drugog broja gdje se korisnik može predomisliti koji operator želi da koristi. Kada se dođe u ovo stanje pritiskom nekih od operatora potrebno je i dalje prikazivati u rezultatu prvi unešeni broj. Ukoliko korisnik stisne neki drugi operator, prethodni operator je potrebno zaboraviti i novi operator se treba zabilježiti. Ukoliko korisnik stisne tipku = potrebno je ignorisati njegov input i ostati u trenutnom stanju. Tipka za brisanje treba da izbriše rezultat, a kalkulator da se vrati u Stanje 1, bez prikazanih cifara. Konačno, ukoliko korisnik stisne neku od cifara potrebno je prebaciti kalkulator u Stanje 3.
- Stanje 3 - predstavlja stanje u kojem korisnik unosi drugi broj. Prvi broj i operator su već unešeni. Cifre treba da jednostavno proširuju drugi broj kao što je to bio slučaj u stanju 1. Tipka za brisanje prebacuje kalkulator u stanje 1 gdje su sve cifre izbrisane. Ukoliko korisnik stisne neki od operatora **ILI** tipku = potrebno je proračunati rezultat na osnovu prvog broja, unešenog operatora i trenutno prikazanog drugog broja. Kalkulator treba da se prebaci u Stanje 1, gdje je trenutni broj stanja 1 dobiveni rezultat. Dakle, rezultujući broj je moguće dalje proširivati dodavajući cifre na desnu stranu.

Ograničenja Aplikacije

Maksimalni broj cifara sa kojima kalkulator radi je 10. Korisnik treba biti ograničen tako da ne može unijeti više od 10 cifara u kalkulator Ukoliko se proračuna vrijednost koja je veća od 10 **cifara**, potrebno je ispisati za rezultat + ili - ∞ u zavisnosti od predznaka broja. Program treba da radi operacije sa `int64` tipom u pozadini. Bilo koja operacija sa vrijednošću ∞ treba da rezultuje istom vrijednošću beskonačnosti osim dijeljenja sa nulom.

Ukoliko korisnik pokuša bilo koji broj (pa i beskonačnost) dijeliti sa nulom, kalkulator treba da prikaže vrijednost **NaN**. Bilo koja operacije sa vrijednošću **NaN** treba da rezultuje istom **NaN** vrijednošću.

Mini demonstracija korištenja kalkulator web aplikacije se nalazi ispod.



Predavanje Zadaće

Zadaću je potrebno predati na classroom do navedenog datuma. Zadaća se sastoji od **JEDNOG** foldera čije je ime u formatu: `ime_prezime_index`

Unutar tog foldera treba da se nalaze 2 foldera: Zadatak1 i Zadatak2. Prvi folder treba da sadrži solution koji ste kreirali prilikom izrade prvog zadatka. Drugi folder treba da sadrži vaš kompajliran web site zajedno sa F# kodom i svim dodatnim resursima.

Dati folder (`ime_prezime_index`) je potrebno zapakovati u tar.gz arhivu. Ime arhive treba da bude: `ime_prezime_index.tar.gz`

Arhivu je potrebno okačiti na Classroom. Obratiti pažnju da predajete GZIP arhivu, ne ZIP preimenovan u GZIP. Zadaće koje se ne mogu otpakovati će biti odbačene.