UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE


# Funkcionalno programiranje

Zadaća 1

Tuzla, decembar/prosinac 2024.

# Zadatak 1

Napisati aplikaciju za menadžment filmova koristeći SQLite bazu podataka. Program treba da podržava operacije:

- Dodavanja filma
- Izmjena filma
- Brisanje filma
- Pretraga filmova po imenu
- Pretraga filmova po žanru
- Pretraga filmova po ratingu

Svaku instancu filma je neophodno čuvati u bazi podataka kao i u memoriji pomoću korisnički definiranog tipa `Movie`. Movie se treba moći konstruisati koristeći:

- `ID` - unikatan identifikator filma
- `Title` - naslov filma
- `Genre` - žanr filma
- `ReleaseYear` - godina izdavanja
- `Rating` - ocjena filma od 1 do 10

`Movie` treba da naslijedi apstraktnu klasi `DatabaseEntity` i implementira metod `BindCommitParameters`. Ovaj metod treba da veže placeholder stringove za njihove vrijednosti. Ukoliko je vrijednost placeholder-a "`@Title`", vezivanje vrijednosti se vrši sa:

```
cmd.Parameters.AddWithValue("@Title", this.Title)
```

Gdje `this`.Title predstavlja `string`.

Primjetiti da nasljeđivanjem klase `DatabaseEntity`, klasa `Movie` će dobiti i metod `Commit` kojim može snimiti svoje stanje u bazu podataka. Koristeći ovu funkcionalnost implementirati interfejs `ISaveable` koji treba da definira query string za dodavanje/update-anje filma u bazi podataka, te pozove metod `Commit`. Ovaj metod prilikom izvršavanja commit-a će pozvati prethodno implementiran `BindCommitParameters` za vezivanje placeholder-a definiranog query stringa.

```
let sqlQuery = @"INSERT OR REPLACE INTO Movies(Title)
VALUES(@Title);"
```

U stringu iznad `@Title` je placeholder koji se može zamjeniti pozivom `AddWithValue`.

Za pretragu je potrebno implementirati interfejs `ISearchable` koji sadrži metod `FromReader`. `SqliteDataReader` sadrži podatke o jednom redu `SELECT` query-a. Na primjer, ukoliko je `SELECT` query:

```
SELECT Title, Rating FROM Movies;
```

U reader-u će se nalaziti `Title` i `Rating` (`string` i `double`) te ih je moguće dohvatiti kao:

```
reader.GetString(0);
reader.GetDouble(1);
```

Get metod govori koji je tip podatka koji dohvatate iz reader-a, dok indeks argument govori koji po redu je dati podatak u `SELECT` query-u (`Title` je na indeksu 0, `Rating` je na indeksu 1).

Primjer programa koji instancira jedan film, snima ga u bazu podataka te pretražuje bazu po žanru.

```
open DatabaseAPI
open MovieManagement
open System.Collections.Generic

let createDB = "CREATE TABLE IF NOT EXISTS Movies (
    Id TEXT PRIMARY KEY,
    Title TEXT NOT NULL,
    Genre TEXT NOT NULL,
    ReleaseYear INTEGER NOT NULL,
    Rating REAL NOT NULL
);"
let db = new Database("baza.db", createDB)
let movie = Movie(db, "movie1", "The Shawshank Redemption", "Drama",
1994, 9.3)
(movie :> ISaveable).Save()

let searchMoviesByGenre (database: Database) (genre: string) =
     let sql = "SELECT Id, Title, Genre, ReleaseYear, Rating FROM
Movies WHERE Genre = @Genre;"
    let parameters = Dictionary<string, obj>()
    parameters.Add("@Genre", genre)
    database.Search<Movie>(sql, parameters)

let parameters = Dictionary<string, obj>()
parameters.Add("@Genre", "Drama")

let movies = searchMoviesByGenre db "Drama"

for m in movies do
  printfn "%A" (m.Title)
```

Klase `Database`, `DatabaseEntity` te interfejsi `ISaveable` i `ISearchable` su dati u

prilogu zadaće kao C# projekat pod imenom `DatabaseAPI`. Za implementaciju zadaće potrebno je napraviti dodani projekat pod imenom `MovieManagement` te od svega toga napraviti .NET Solution (predavanje 7).

Microsoft.Data.Sqlite referenca.

## Extras

1. Dodati interfejs za brisanje objekata u `DatabaseAPI` projektu te ga implementirati u `MovieManagement` projektu.
2. Za generisanje ID-eva koristiti `NanoID` generator. Nuget link. Github link.

# 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.

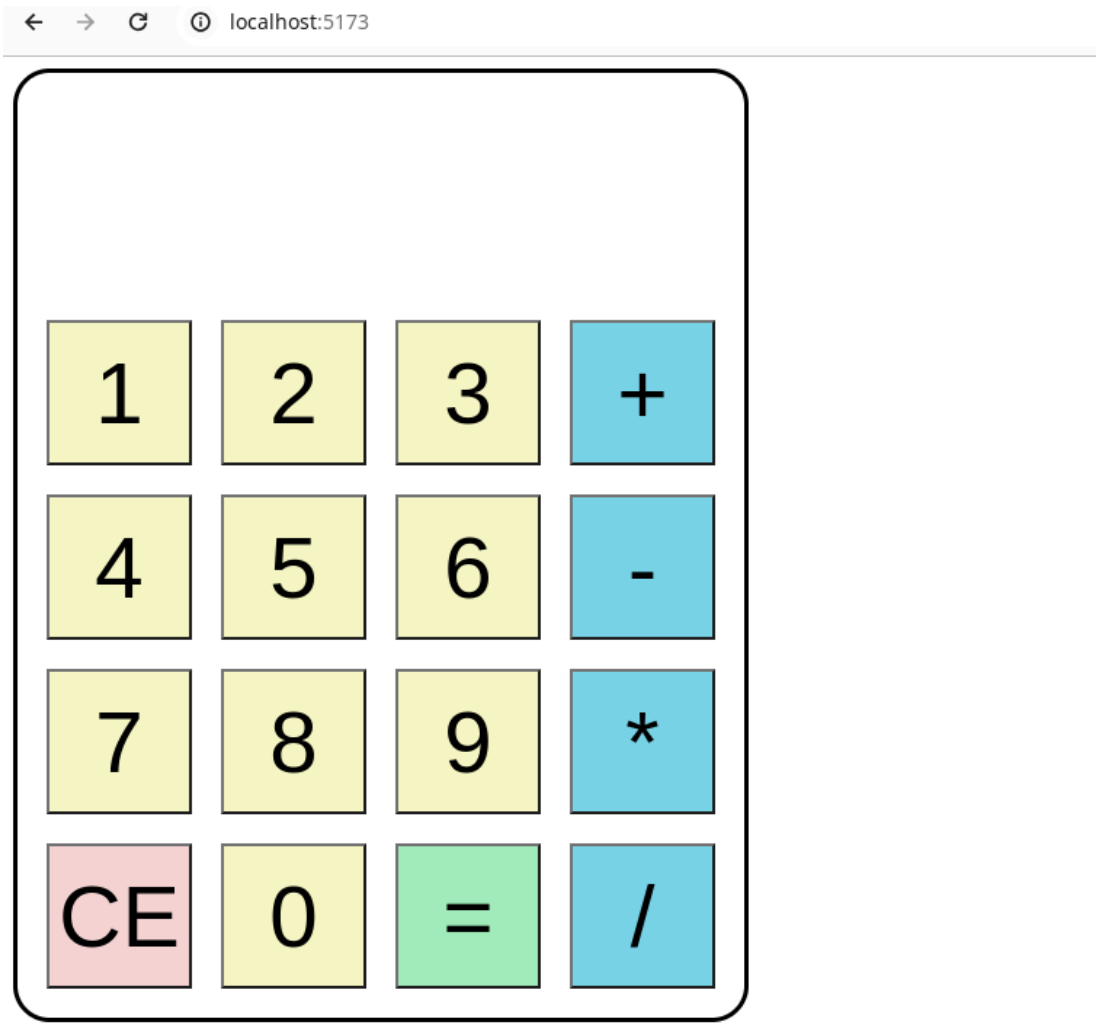
## Ograničenja Aplikacije

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.