FUNKCIONALNO-PROJEKAT

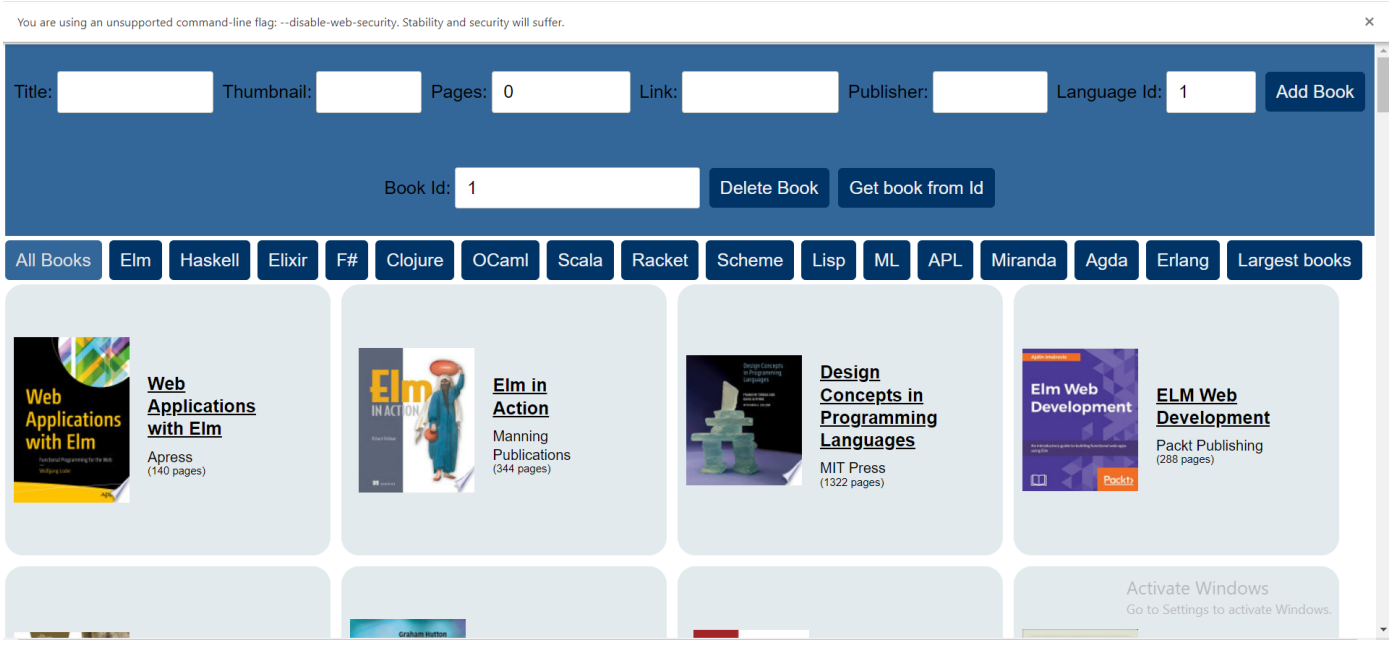
Autor: Milica Babić

link ka repozitorijumu sa kodom aplikacije na github-u

Ovo je prikaz i opis web-aplikacije koju smo radili za projekat iz Funkcionalnog programiranja. Zadatak nam je bio realizovati web-aplikaciju u čisto funkcionalnim jezicima, a prvenstveno u Haskelu, koji smo radili i na nastavi. Ova aplikacija se sastoji iz dva dijela, klijent i server, koji komuniciraju preko http-zahtjeva i odgovora, a pri tom podaci su uglavnom prenošeni u obliku JSON-a, u body elementu http-zahtjeva ili odgovora. Klijent je realizovan u jeziku za frontend pod nazivom Elm, o kojem ce biti više riječi kasnije. Server je realizovan u Haskellu, koristeći PostgreSQL bazu podataka i Servant biblioteku za izradu servera i API-ja. Za konekciju sa bazom koristi se biblioteka Persistent, koja omogućava ORM (object-relational mapping), tj. ekvivalentan koncept u funkcionalnim jezicima. Ta biblioteka takođe omogućava jednostavnije upite u obliku funkcija kao što su get, insert, delete, čijom se upotrebom smanjuje potreba za pisanjem čistog SQL koda od strane programera, pa prema tome se smanjuje mogućnost napada kao što su SQL injection. Druga biblioteka koriććena za pisanje upita je Esqueleto. Ona omogućava pisanje složenijih upita kao što su JOIN-i dvije ili više tabela, WHERE upiti i slično.

Ove tehnologije i struktura aplikacije su izabrani zbog iskustva u izradi web-aplikacija kod kojih je React korišćen za frontend, a Node.js za backend; frontend je prema tome SPA (single-page application), a backend je RESTful API. Izradom ovog projekta se uočavaju prednosti i mane korišćenja funkcionalnih jezika, dok je sama arhitektura aplikacije ista kao u standardnim PERN aplikacijama (PostgreSQL+Express+React+Node.js).

Aplikacija se zove `Katalog knjiga o funkcionalnim jezicima` i pruža mogućnost prikaza velikog broja knjiga koje se bave funkcionalnim programskim jzicima. Klijent nakon pokretanja izgleda kao na narednoj slici, gdje se na stranici ispod forme prikaže red dugmadi za filtriranje knjiga po kategorijama. Prvo dugme u redu postiže isti efekat kao pokretanje aplikacije, a to je prikaz svih knjiga u nizu kartica, dok se na svaku karticu može kliknuti, što nas vodi na Google Books stranicu namijenjenu toj knjizi. Ostala dugmad filtriraju kartice, tako da se prikažu knjie po jezicima, dok posljednje dugme vraća prikaz 10 najobimnijih knjiga u bazi i postoji prvenstveno da bi se demonstrirala upotreba složenijih upita, kao što su JOIN dvije tabele i sotiranje. Forma omogućuje unos podataka o novoj knjizi koja se klikom na dugme Add Book dodaje u bazu, kao i brisanje knjige sa unijetim indeksom, ako ona postoji u bazi, ili pak selektovanje i prikaz jedne knjige koja odgovara unijetom indeksu.



Naredni dio ovog teksta, za server koji je rađen u Haskelu, je preuzet i prilagodjen sa: https://github.com/MondayMorningHaskell/RealWorldHaskell#readme. Ostavljeni su dijelovi koji su korišćeni kao primjer i osnova, a to su prvi, drugi i peti dio, a proširen je tekst u nekim dijelovima, jer je i kod aplikacije opširniji od ovog koji je služio kao polazna tačka i referenca. Prema tome, preporuke za blog autora, gdje je sve još detaljnije objašnjeno Real World Haskell

Instalacija neophodnog

Postgresql

Pokretanje aplikacije zahtijeva instalaciju Postgres-a. Ako se nakon lociranja u folder sa serverom pomoću komandi `cd serverHaskell`, `cd Books-Server` i izvršavanja naredbe `stack build` vidi sljedeća poruka o grešci, to je indikacija da Postgres nije instaliran ili nije dostupan:

```
>> stack build
setup: The program 'pg_config' is required but it could not be found
```

Na Linux operativnom sistemu su neophodni bar sljedeći paketi:

```
>> sudo apt install postgresql postgresql-contrib libpq-dev
```

Na Windows i MacOS operativnim sistemima - instalacija.

U fajlu `Database.hs` se vide podaci za uspostavljanje konekcije s bazom; u našem slučaju je difoltni port za Postgres `5432`, naziv korisnika je `postgres`, naziv baze `probnaBaza`, a šifa je `postgres`, pa je neophodno podesiti ove podatke:

```
localConnString :: PGInfo
localConnString = "host=127.0.0.1 port=5432 user=postgres dbname=probnaBaza password=postgres"
```

Pokretanje servera

Nakon što je Postgres instaliran i podešeni potrebni podaci, locirati se u folder sa serverom i kompajlirati:

```
>> stack build
```

Nakon toga komanda za migraciju baze:

```
>> stack exec migrate-db
```

Pa onda komanda za pokretanje servera na portu 5000 (Promijeniti port u fajlu BasicServer.hs ako ne odgovara):

```
>> stack exec run-server
```

U fajlu `BasicSchema.hs` se vidi definicija šeme baze:

```
PTH.share [PTH.mkPersist PTH.sqlSettings, PTH.mkMigrate "migrateAll"] [PTH.persistLowerCase|
  Language sql=languages
    name Text
    UniqueTitle name
    deriving Show Read Eq

  Book sql=books
    title Text
    thumbnail Text
    pages Int
    link Text
    publisher Text
    languageId LanguageId
    UniqueText title
    deriving Show Read Eq
|]
```

Postoje dvije tabele, languages i books; u prvoj je naziv jezika jedinstven, a id iz prve je strani ključ u drugoj, pod nazivom languageId. U drugoj tabeli je kolona title jedinstvena, a ostale kolone su thumbnail, pages, link i publisher. Tu se vidi definicija tipa Book i Language. Prikaz nekoliko relacija u tabelama preko `json` notacije i tabelarno:

```
{
  "id":1,
  "name":"Elm"
}
```

```
{
  "id":2,
  "name":"Haskell"
}
```

id	name

id	name
1	Elm
2	Haskell

Ovo je za tabelu languages, a za tabelu books sljedecih nekoliko primjera relacija predstavljenih `json` objektima i odgovarajući prikaz u tabeli:

```
{"title":"Web Applications with Elm",
"thumbnail":"http://books.google.com/books/content?
id=KnhqDwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
"pages":140,
"link":"https://books.google.com/books/about/Web_Applications_with_Elm.html?hl=&id=KnhqDwAAQBAJ",
"publisher":"Apress",
"languageId":1
}
```

```
{"title":"Programming in Haskell",
"thumbnail":"http://books.google.com/books/content?
id=75C5DAAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
"pages":320,
"link":"https://books.google.com/books/about/Programming_in_Haskell.html?hl=&id=75C5DAAAQBAJ",
"publisher":"Cambridge University Press",
"languageId":2
}
```

```
{"title":"Haskell in Depth",
"thumbnail":"http://books.google.com/books/content?
id=r4UxEAAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
"pages":664,
"link":"https://books.google.com/books/about/Haskell_in_Depth.html?hl=&id=r4UxEAAAQBAJ",
"publisher":"Simon and Schuster",
"languageId":2
}
```

id	title	thumbnail	pages	link	publisher	languageId
2	Web Applications with Elm	http://books.google.com/books/content?id=Knh...	140	https://books.google.com/books/about/We_...	Apress	1
8	Programming in Haskell	http://books.google.com/books/content?id=7...	320	https://books.google.com/books/about/Program...	Cambridge University Press	2
9	Haskell in Depth	http://books.google.com/books/content?id=r4Ux...	664	https://books.google.com/books/about/Haskell_in_D...	Simon and Schuster	2

Konekcija sa bazom: Persistent

Biblioteka za konekciju backend-a sa bazom je Persistent. Izabrana je zbog mnogih pozitivnih karakteristika, kao i broja preuzimanja paketa sa Hackage repozitorijuma. Nije neophodno da se koristi PostgreSQL baza uz biblioteku Persistent, već ona omogućava podršku za nekoliko najpoznatijih baza. Glavne osobine biblioteke prema autorima iste, preuzeto sa - Persistent :: Yesod Web Framework Book :

> ... Persistent is Yesod's answer to data storage- a type-safe, universal data store interface for Haskell.
>
> Haskell has many different database bindings available. However, most of these have little knowledge of a schema and therefore do not provide useful static guarantees. They also force database-dependent APIs and data types on the programmer.
>
> ...
>
> In contrast, Persistent allows us to choose among existing databases that are highly tuned for different data storage use cases, interoperate with other programming languages, and to use a safe and productive query interface, while still keeping the type safety of Haskell datatypes.
>
> Persistent follows the guiding principles of type safety and concise, declarative syntax. Some other nice features are:
>
> - Database-agnostic. There is first class support for PostgreSQL, SQLite, MySQL and MongoDB, with experimental Redis support.
> - Convenient data modeling. Persistent lets you model relationships and use them in type-safe ways. The default type-safe persistent API does not support joins, allowing support for a wider number of storage layers. Joins and other SQL specific functionality can be achieved through using a

> raw SQL layer (with very little type safety). An additional library, `Esqueleto`, builds on top of the Persistent data model, adding type-safe joins and SQL functionality.
>
> - Automatic database migrations in non-production environments to speed up development.
>
> Persistent works well with Yesod, but it is quite usable on its own as a standalone library.

Pretvaranje Haskell tipova u JSON i obrnuto: Aeson

U fajlu `BasicSchema.hs` se nalaze funkcije za parsiranje JSON-a i prevodjenje odgovarajuce JSON objekta u Haskell-ove tipove Book i Language i obrnuto, koje koriste biblioteku Aeson.

Server i API: Servant

Za definisanje izlaznih tačaka ovog servera korišćena je biblioteka Servant. O samoj biblioteci i korišćenju detaljnije na zvaničnom tutorijalu.

Serveru se mogu slati HTTP-zahtjevi sa bilo kojeg klijentskog programa. Jedna od opcija je Postman. (Na ovaj način smo prvobitno popunili bazu, da bismo testirali ispravnost APIja.) Slanjem tih zahtjeva se lakše vidi koje funkcionalnosti obezbjeđuje naš server. Jedna od njih je dodavanje nove knjige slanjem POST zahtjeva na adresu http://localhost:5000/books :

```
POST /books
{"title":"Web Applications with Elm",
"thumbnail":"http://books.google.com/books/content?
id=KnhqDwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
"pages":140,
"link":"https://books.google.com/books/about/Web_Applications_with_Elm.html?hl=&id=KnhqDwAAQBAJ",
"publisher":"Apress",
"languageId":1
}

...

2
```

Dohvatanje dodate knjige:

```
GET /books/2

...

{"title":"Web Applications with Elm",
"thumbnail":"http://books.google.com/books/content?
id=KnhqDwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
"pages":140,
"link":"https://books.google.com/books/about/Web_Applications_with_Elm.html?hl=&id=KnhqDwAAQBAJ",
"publisher":"Apress",
"languageId":1,
"id": 2
}
```

Pokušaj dohvatanja nepostojeće knjige:

```
GET /books/110

...

Could not find user with that ID
```

Dohvatanje liste parova [jezik, knjiga]:

```
GET /books/joinJezik
[
  [
    {
      "name": "Elm",
      "id": 1
    },
    {"title":"Web Applications with Elm",
      "thumbnail":"http://books.google.com/books/content?
```

```
id=KnhqDwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
      "pages":140,
      "link":"https://books.google.com/books/about/Web_Applications_with_Elm.html?hl=&id=KnhqDwAAQBAJ",
      "publisher":"Apress",
      "languageId":1,
      "id": 2
    }
  ],
  [
    {
      "name": "Haskell",
      "id": 2
    },
    {"title":"Programming in Haskell",
      "thumbnail":"http://books.google.com/books/content?
id=75C5DAAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
      "pages":320,
      "link":"https://books.google.com/books/about/Programming_in_Haskell.html?hl=&id=75C5DAAAQBAJ",
      "publisher":"Cambridge University Press",
      "languageId":2
    }
  ]
]
```

Dohvatanje liste knjiga za jedan jezik:

```
GET /books/elm
[
  {"title":"Web Applications with Elm",
    "thumbnail":"http://books.google.com/books/content?
id=KnhqDwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
    "pages":140,
    "link":"https://books.google.com/books/about/Web_Applications_with_Elm.html?hl=&id=KnhqDwAAQBAJ",
    "publisher":"Apress",
    "languageId":1,
    "id":2
  },
  {"title":"Elm in Action",
    "thumbnail":"http://books.google.com/books/content?
id=rHbgDwAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api",
    "pages":344,
    "link":"https://books.google.com/books/about/Elm_in_Action.html?hl=&id=rHbgDwAAQBAJ",
    "publisher":"Manning Publications",
    "languageId":1,
    "id":3
  }
]
```

Brisanje knjige sa izabranim id-om:

```
POST /books/2

...

[]
```

Ovakav način brisanja nije najbolji i trebalo bi da se promijeni u budućnosti, da se realizuje pomoću DELETE zahtjeva, i da vraća indeks knjige ili poruku ako je uspješno obrisana, a ako nije, onda poruku o tome takođe. Implementacija ovih funkcionalnosti je u fajlu `BasicServer.hs`.

Složeniji upiti i realizacija JOIN-a: Esqueleto

Kako Persistent ne omogućuje JOIN-e, dodana je i ova biblioteka. Primjer funkcije iz fajla `Database.hs`:

```
fetchRecentBooksPG :: PGInfo -> IO [(Entity Language , Entity Book)]
fetchRecentBooksPG connString = runAction connString fetchAction
  where
    fetchAction :: SqlPersistT (LoggingT IO) [(Entity Language, Entity Book)]
    fetchAction = select . from $ \(languages `InnerJoin` books) -> do
      on (languages ^. LanguageId  ==. books ^. BookLanguageId)
```

```
        orderBy [desc (books ^. BookPages)]
        limit 10
        return (languages, books)
```

Ova funkcija realizuje prethodno navedeni upit GET /books/joinJezik. Detaljnije o biblioteci na link.

Logovanje: monad-logger

Za logovanje informacija korišćena je biblioteka `monad-loger` u fajlu `Database.hs`.

To je to što se tiče serverske strane aplikacije, naredni dio je posvećen klijentskom dijelu. Klijent je pisan u jeziku Elm, namijenjenom za frontend. Hvale ga zbog dizajna kompajlera i vrlo korisnih poruka o greškama. Takođe, autori JS biblioteke Redux kažu da im je inspiracija bila Elm arhitektura. Ova biblioteka se inače koristi u paru sa bibliotekom React, jednom od najpopularnijih frontend JS biblioteka. Detaljnije o jeziku na zvaničnom tutorijalu - An Introduction to Elm. Autor jezika Elm o prednosti korišćenja funkcionalnog jezika za frontend:

> Why a functional language?
>
> You can get some benefits from programming in a functional style, but there are some things you can only get from a functional language like Elm:
>
> - No runtime errors in practice.
> - Friendly error messages.
> - Reliable refactoring.
> - Automatically enforced semantic versioning for all Elm packages. No combination of JS libraries can give you all of these guarantees. They come from the design of the language itself! And thanks to these guarantees, it is quite common for Elm programmers to say they never felt so confident while programming. Confident to add features quickly. Confident to refactor thousands of lines. But without the background anxiety that you missed something important!
>
> I have put a huge emphasis on making Elm easy to learn and use, so all I ask is that you give Elm a shot and see what you think. I hope you will be pleasantly surprised!

Instalacija neophodnog

Node.js

Ovaj jezik se kompajlira u JavaScript, pa je potrebno imati Node.js instaliran - instalacija.

Elm jezik i paketi

Nakon instalcije Node.js, zbog korišenja npm menadžera JS paketa, intalirati Elm prateći uputstvo. Ekstenzija za VS Code za podrsku jeziku Elm: Elm. Nakon toga instalirati pakete koji se koriste u projektu:

```
>> elm install elm/http
>> elm install elm/json
>> elm install elm/svg
>> elm install mdgriffith/elm-ui
```

Paketi za sinhronizaciju prikaza i koda za vrijeme razvoja, kao i za formatiranje elm koda:

```
>> npm install -g elm-test elm-format elm-review elm-live
```

Pokretanje klijenta

```
>> npx elm-live src/Main.elm
```

Klijent se nalazi na http://localhost:8000/, ali ako bi se samo ovako otvorio browser, javljala bi se CORS greška, zato što server ne sadrži funkcije za njeno otklanjanje, što bi trebalo definitivno dodati u budućnosti. Zbog toga zatvoriti Google Chrome i ponovo ga otvoriti iz komandne linije komandom (na Windows operativnom sistemu):

```
>> "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --args --user-data-dir="/tmp/chrome_dev_test" --disable-
web-security
```

Sad u tako otvorenom browseru sa onemogućenim sigurnosnim flag-ovima, otvoriti http://localhost:8000/.

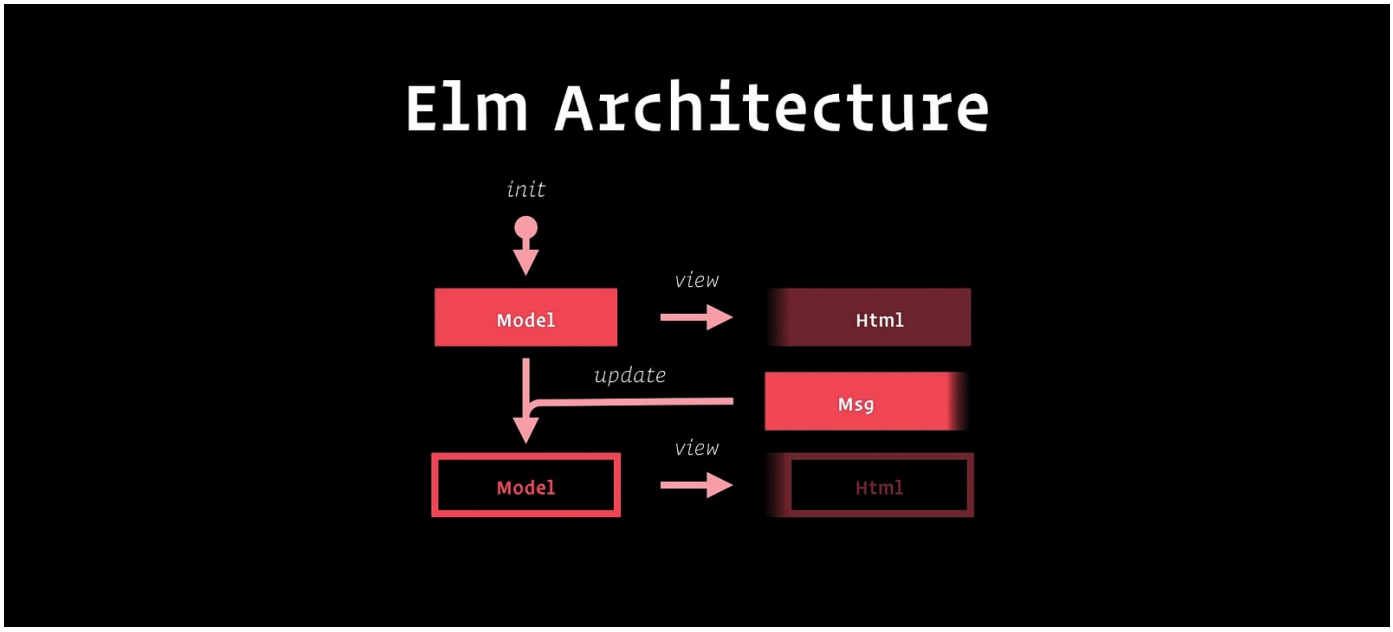
Elm arhitektura je varijacija na MVC stil, tako da moramo imati definisan Model:

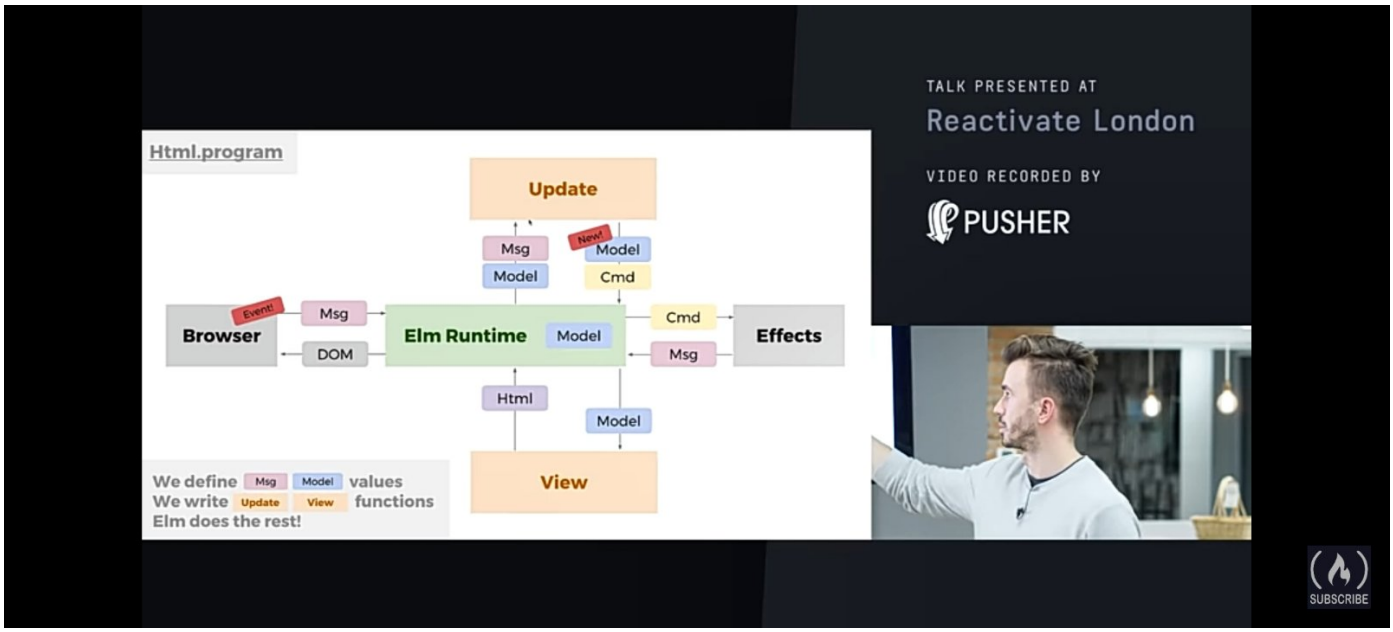
```
type alias Model =
    { bookTitle : String
    , bookThumbnail : String
    , bookPages : Int
    , bookLink : String
    , bookPublisher : String
    , bookLanguageId : Int
    , bookId : Int
    , poruka : String
    , results : List Book
    , resultBook : Maybe Book
    , errorMessage : Maybe String
    , loading : Bool
    }
```

Početni model:

```
initModel : Model
initModel =
    { bookTitle = ""
    , bookThumbnail = ""
    , bookPages = 0
    , bookLink = ""
    , bookPublisher = ""
    , bookLanguageId = 1
    , bookId = 1
    , poruka = ""
    , results = []
    , resultBook = Nothing
    , errorMessage = Nothing
    , loading = False
    }
```

On služi za generisanje View-a; razne interakcije klijenta sa view-om uzrokuju slanje poruka, koje su prvobitno definisane kao tip Msg. Te poruke se prosljeđuju funkciji update koja u zavisnosti od poruka update-uje polazni model. Takođe, poruke tipa Msg može generisati dobijanje odgovora od servera, pa i na njih adekvatno reaguje update funkcija.





Primjer jedne od Update funkcija:

```
updateAddBook : Model -> (Model, Cmd Msg)
updateAddBook model =
    ( { model | loading = True }, postBooks model )
```

Paket za UI: elm-ui

Ovaj paket omogućava jednostavnije stilizovanje, planiranje i raspoređivanje elemenata u prikazu. Neke od osobina, prema autorima:

> The high level goal of this library is to be a **design toolkit** that draws inspiration from the domains of design, layout, and typography, as opposed to drawing from the ideas as implemented in CSS and HTML.
>
> This means:
>
> - Writing and designing your layout and view should be as **simple and as fun** as possible.
> - Many layout errors (like you'd run into using CSS) **are just not possible to write** in the first place!
> - Everything should just **run fast**.
> - **Layout and style are explicit and easy to modify.** CSS and HTML as tools for a layout language are hard to modify because there's no central place that represents your layout. You're generally forced to bounce back and forth between multiple definitions in multiple files in order to adjust layout, even though it's probably the most common thing you'll do.

Paket za slanje zahtjeva: http

Primjer funkcije u našem kodu gdje se šalje GET zahtjev:

```
cmdSearchAll : Cmd Msg
cmdSearchAll=
    Http.get
        {
          url = "http://localhost:8080/books/sve"
        , expect = Http.expectJson MsgGotResults decodeItems
        }
```

Primjer funkcije za slanje POST zahtjeva:

```
postBooks : Model -> Cmd Msg
postBooks model =
  Http.post
    { url = "http://localhost:8080/books"
    , body = jsonBody (encode model)
    , expect = Http.expectWhatever MsgSuccesfulPost
    }
```

Paket za enkodiranje i dekodiranje JSONa: json

Ovaj paket omogućava pretvaranje JSON objekata koji se dobiju u http odgovoru u odgovarajuće Elm-tipove, i obrnuto. Tip knjiga iz aplikacije:

```
type alias Book =
    { title : String
    , thumbnail : Maybe String
    , link : String
    , pages : Maybe Int
    , publisher : Maybe String
    }
```

Primjer funkcije za dekodiranje JSON-a u tip Book:

```
decodeItem : JD.Decoder Book
decodeItem =
    JD.map5 Book
        (JD.field "title" JD.string)
        (JD.maybe (JD.field "thumbnail" JD.string))
        (JD.field "link" JD.string)
        (JD.maybe (JD.field "pages" JD.int))
        (JD.maybe (JD.field "publisher" JD.string))
```

Primjer funkcije za enkodiranje tipa Book u JSON:

```
encode : Model -> JE.Value
encode model =
  JE.object
    [ ("title", JE.string model.bookTitle)
    , ("thumbnail", JE.string model.bookThumbnail)
    , ("pages", JE.int model.bookPages)
    , ("link", JE.string model.bookLink)
    , ("publisher", JE.string model.bookPublisher)
    , ("languageId", JE.int model.bookLanguageId)
    ]
```

Paket za crtanje i prikaz Scalable vector graphics (SVG): svg

Pisanje dokumentacije

Ova dokumentacija je pisana korišćenjem jezika za formatiranje pod nazivom Markdown. Sintaksa i detaljnije na Markdown Guide. Za pretvaranje .md fajla u PDF je korišćena ekstenzija u VSCode editoru pod nazivom Markdown PDF. Nakon dodavanja u VSCode editor, vrlo jednostavno se koristi:

> 1. Open the Markdown file
> 2. Press F1 or Ctrl+Shift+P
> 3. Type export and select below
>   - markdown-pdf: Export (settings.json)
>   - markdown-pdf: Export (pdf)
>   - markdown-pdf: Export (html)
>   - markdown-pdf: Export (png)
>   - markdown-pdf: Export (jpeg)
>   - markdown-pdf: Export (all: pdf, html, png, jpeg)

Dodatne reference

I za kraj, dodatne reference:

- Za izradu klijenta veoma mi je koristio kurs: Elm - The Complete Guide (a web development video tutorial).
- Za slike vezane za Elm arhitekturu videi: Developer Happiness on the Front End with Elm i Elm crash course - Building unbreakable webapps fast.
- Primjeri u elmu: Examples i github repozitorijum sa kodom sa kursa.
- Repozitorijumom sa paketima za jezik Elm: Elm Packages.
- Blog sa primjerom TODO aplikacije: Elm: Functional Frontend.