

RAZVOJ SPLETNIH APLIKACIJ V OGRODJU YESOD

DUŠAN FISTER, IZTOK FISTER JR.

Povzetek: Razvita ogrodja za razvoj spletnih aplikacij lahko domala najdemo za vsak programski jezik. Dandanes prevladujejo ogrodja v programskih jezikih Python, Java, Go, PHP, Ruby. Yesod je spletno ogrodje, razvito v programskem jeziku Haskell, namenjeno za razvoj tako enostavnih kot tudi zahtevnih spletnih aplikacij. Glavni značilnosti, ki opisujeta to ogrodje sta varnost in zmogljivost delovanja. V tem prispevku kratko opisujemo programski jezik Haskell in z nekaterimi primerki programske kode predstavljamo ogrodje Yesod.

Ključne besede: Haskell, Yesod, funkcijski programski jeziki, spletno programiranje, varnost

NASLOVA AVTORJEV: Dušan Fister, mladi raziskovalec, Univerza v Mariboru, Ekonomsko-poslovna fakulteta, Maribor, Slovenija, e-pošta: dusan.fister1@um.si. doc. dr. Iztok Fister Jr., asistent, Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, Slovenija, e-pošta: iztok.fister1@um.si.

<https://doi.org/10.18690/978-961-286-282-4.25>
Dostopno na: <http://press.um.si>

ISBN 978-961-286-282-4

1 UVOD

Zgodovina računalništva prinaša mnogo različnih načinov programiranja, s tem pa tudi kopico najrazličnejših programskih jezikov. Programiranje je eden izmed najosnovnejših načinov komunikacije med človekom in računalnikom, kjer človek z vpisovanjem ukazov v različnih oblikah neposredno usmerja delovanje računalnika. Obstaja veliko načinov programiranja, med katerimi so najbolj pogosti imperativni, objektni in funkcijski.

Eden izmed razširjenih funkcijskih jezikov je programski jezik Haskell. Tega pogosto srečamo ne le v akademskih krogih, temveč tudi v industriji, kjer se njegova priljubljenost, sodeč po zadnjem trendu, močno povečuje. Zaradi zagotavljanja visoke stopnje varnosti in hitrosti izvajanja aplikacij, je Haskell priljubljen tudi pri razvijalcih spletnih aplikacij. Njegov prevajalnik namreč zagotavlja zavidljivo zmogljivost izvajanja programske kode, podobno kot programski jezik C, kakor tudi lastnost, da uporabnik morebitne napake v programski kodi opazi takoj med postopkom prevajanja, namesto da na njih naleti pri kasnejšem delovanju (angl. runtime bugs). Vse kakršne tovrstne napake utegnejo povzročati neželene motnje v delovanju ter tako doveljavati varnostne luknje [4].

Programski jezik Haskell praktično srečamo pri razvoju visoko-zmogljivih aplikacij v finančni industriji [10], računalništvu v oblaku [8] ter uporabnih aplikacijah umetne inteligence [9], ipd. Nenazadnje srečamo Haskell tudi pri razvoju vgrajenih domensko specifičnih jezikov (angl. embedded domain-specific languages, krajše EDSL [5]).

Obstaja več spletnih ogrodij, razvitih v programskem jeziku Haskell, kot npr. Happstack, Snap, Scotty, Servant in Yesod. Izmed naštetih je v komercialnih aplikacijah najširše zastopan ravno Yesod. Za tega namreč obstaja obilo programskih knjižnic, ki poenostavljajo delo s programiranjem, dodatno pa je slednji že dlje časa izpostavljen ustaljenemu razvoju. K temu priča tudi mnogo zastavljenih vprašanj in odgovorov uporabnikov tega ogrodja na platformi StackOverflow.

V tem prispevku na kratko predstavljamo programski jezik Haskell, navajamo njegove prednosti glede na ostale programske jezike ter prikazujemo njegovo uporabno vrednost. Izpostavljam bistvene značilnosti programskega jezika Haskell ter demonstriramo delovanje. Podrobneje predstavljamo in opisujemo spletno ogrodje Yesod in nekaj njegovih osnovnih primerov.

Struktura tega prispevka je sledeča: v poglavju 2 je na kratko predstavljen programski jezik Haskell, medtem ko je v poglavju 3 opisano ogrodje Yesod za razvoj spletnih strani. Navedenih je nekaj praktičnih primerov uporabe ogrodja Yesod. Postavitve (angl. deployment) aplikacije je prikazana v poglavju 4, v poglavju 5 pa podan povzetek prispevka.

2 PROGRAMSKI JEZIK HASKELL

Programski jezik Haskell spada med čiste funkcijske jezike (angl. purely functional languages), katerega začetek razvoja sega v osemdeseta leta prejšnjega stoletja. Tehnično gledano je programski jezik Haskell [1]:

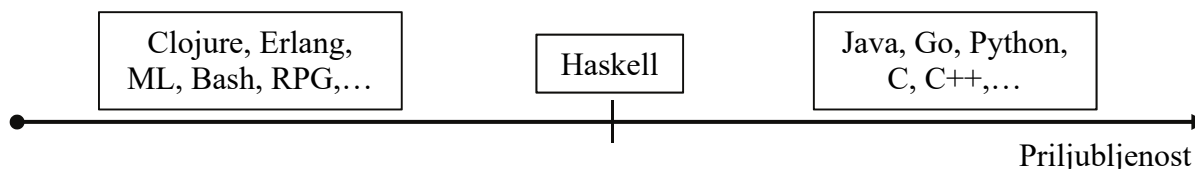
- močno in statično tipiziran jezik (angl. statically typed), kar pomeni da prevajalnik samodejno zaznava tip spremenljivk, npr. celo število, realno število, znak [16],
- uporablja zakasneni izračun (angl. lazy evaluation), kar pomeni da Haskell ne izračunava rezultatov, dokler ni prisiljen. Praktično to pomeni da se z definicijo funkcije ta ne izvede takoj, ampak šele dokler aplikacija ne potrebuje rezultatov te funkcije,
- eleganten in zgoščen jezik (angl. elegant and concise), ker visoko-nivojske koncepte opisuje smiselno, kratko ter jedrnat.

Funkcijski jeziki v splošnem zahtevajo popolnoma drugačno razmišljanje, kot npr. pri uporabi objektno orientiranih jezikov. Kakopak je zaradi tega drugačna tudi sintaksa jezika. V tabeli 1 je predstavljena osebna izkaznica programskega jezika Haskell [7,13].

Tabela 1. Osebna izkaznica programskega jezika Haskell [7,13]

Nastanek jezika	1990
Prevajalniki	GHC (Glasgow Haskell Compiler), Hugs
Nastal po vzoru sledečih programskih jezikov	Miranda, Lisp, Orwell (med drugimi)
Vplival na razvoj sledečih programskih jezikov	C++ 11, Python, Clojure, Elm, Rust, Scala, Swift (med drugimi)
Operacijski sistem	Deluje na vseh platformah
Končnice datotek	.hs, .hls
Github repozitorij	https://github.com/haskell
Aplikacije, ki uporabljajo Haskell [12]	xmonad, Monadius, Flippi, AMuZeD, Paradox, LPS, Knit, Truth, HWS-WP, Yi, Hoogle, EDSL

Priljubljenost programskega jezika dokazuje lestvica »Tiobe Index«, ki trenutno Haskell uvršča na 45. mesto med vsemi programskimi jeziki. Slika 1 prikazuje izvleček Tiobe lestvice, kamor umeščamo programski jezik Haskell.



Slika 1. Delna Tiobe lestvica [15]

Programski jeziki kot npr. Java, Python in C, so sicer prepričljivo pred Haskellom glede na popularnost, a se ta uvršča pred Clojure, Erlang in RPG. Glavni razlog za manjšo popularnost leži v dejstvu, da je Haskell čisti funkcijski jezik in nima veliko skupnega z objektno orientiranim programiranjem (angl. object oriented programming), ki dandanes predstavlja glavni standard razvoja aplikacij. Nekaj dodatnih razlogov za manjšo popularnost programskega jezika Haskell navaja spletni vir [17].

V nadaljevanju poglavja podajamo nekaj enostavnih in kratkih izsekov programske kode Haskell.

2.1 Uvodni program »Hello World«

Tipični uvodni primer programa je t.i. »Hello World«, katerega v programskem jeziku Haskell zapišemo kot prikazuje levi stolpec tabele 2 [3]:

Tabela 2. Hello World

<pre>'hello.hs' main = putStrLn 'Hello World'</pre>	<pre>>> ghc -o hello hello.hs [1 of 1] Compiling Main (hello.hs, hello.o) Linking hello ... >> ./hello >> Hello World</pre>
---	---

Programsko kodo v tem prispevku preizkušamo na operacijskem sistemu Linux Ubuntu, kamor namestimo prevajalnik GHC, npr. najlažje z »apt-get install ghc«. Programsko kodo prevedemo s sledečim ukazom v terminalnem oknu »ghc -o hello hello.hs«. S tem ustvarimo novo izvršljivo datoteko z imenom »hello«, katero nato poženemo z ukazom »./hello«. Ta nam v novi vrstici terminalnega okna izpiše »Hello World«.

2.2 Delo s funkcijami

V naslednjem primeru definiramo našo prvo funkcijo, poimenovano »seštevanje«. Funkcijo uredimo v urejevalniku besedil ter datoteko shranimo kot »sesteh.hs«. Tabela 3 prikazuje primer funkcije seštevanja, kjer število, ki ga želimo sešteti s samim seboj, vnesemo kot vhodni parameter. Uporabnikovo posredovanje je, tako kot pri prejšnjem primeru, označeno z modro barvo v desnem stolpcu tabele 3.

Tabela 3. Funkcija seštevanja

'sestej.hs' sestevanje x = x + x	>> ghci GHCi, version 8.0.2: http://www.haskell.org/ghc/ :? for help Prelude> :l sestej [1 of 1] Compiling Main (sestej.hs, interpreted) Ok, modules loaded: Main. *Main> sestevanje 2 4 *Main>
-------------------------------------	--

V tabeli 3 upodabljam o interaktivno uporabo prevajalnika GHC, katerega zaženemo z ukazom »ghci« ali »ghc --interactive«. V interaktivnem prevajalniku skripto s sledečim ukazom najprej naložimo »:l sestej«. Ta nam jo sprva prevede, oz. definira, šele nato pa jo lahko uporabimo. H klicu funkcije dodamo še vhodni parameter, npr. 2, kot to prikazuje ukaz »sestevanje 2«. Vhodni parameter se v programskem jeziku Haskell zapiše pod spremenljivko »x«, katerega nato seštejemo s samim seboj, prevajalnik pa samodejno izpiše rezultat 4, tj. $2 + 2 = 4$. Vidimo, da funkcijo in vhodne parametre v programskem jeziku ustvarimo zelo kompaktno, tj. v eni vrstici (angl. one-liner). Podobno lahko sklepamo za nekoliko kompleksnejšo funkcijo, ki zraven matematičnih operacij množenja in seštevanja prinaša še pogojni (angl. »if«) stavek (tabela 4) [3].

Tabela 4. Matematične operacije in pogojni stavek

'kvadriraj.hs' kvadrirajMaloStevilo x = (if x > 100 then x else x*x) + 5	Prelude> :l kvadriraj *Main> kvadrirajMaloStevilo 3 14 *Main> kvadrirajMaloStevilo 101 106
---	--

Funkcija »kvadrirajMaloStevilo« kvadrira vhodni parameter, če je ta manjši ali enak 100. Ne glede na velikost vhodnega parametra (saj uporabljamo oklepaj), funkcija delnemu rezultatu prišteje število 5 in končni rezultat izpiše. Brez uporabljenega oklepaja bi funkcija število 5 prištela le, če bi bil vhodni parameter manjši ali enak 100. Kot zanimivost naj povemo, da je »else« stavek v programskem jeziku Haskell obvezen [3].

2.3 Delo s seznammi

Nadalje prikazujemo tudi delo s seznammi (angl. lists), ki so sodeč po [3] najbolj uporabljana podatkovna struktura v programskem jeziku Haskell. Seznami so v programskem jeziku Haskell homogeni, kar pomeni da morajo biti vsi vsebovani elementi seznama enakega tipa, npr. cela števila, karakterji, ipd. [3]. Seznam v programskem jeziku Haskell ustvarimo zelo preprosto, kot to prikazuje uvodna vrstica tabele 5.

Tabela 5. Seznami v programskem jeziku Haskell

Prelude> seznam=[1,2,4,5,6,8] Prelude> seznam [1,2,4,5,6,8] Prelude> seznam !! 3 5 Prelude> head(seznam) 1 Prelude> tail(seznam) [2,4,5,6,8] Prelude> maximum(seznam) 8 Prelude> [x * 2 x <- [1..10]] [2,4,6,8,10,12,14,16,18,20]

Seznam ustvarimo z ukazom »seznam=[1,2,4,5,6,8]«, pri čemer je »seznam« ime seznama, 1,2,4,5,6,8 pa so njegovi elementi. Slednje lahko v terminalnem oknu z ukazom »seznam« izpišemo. Izpišemo lahko tudi posamezni element seznama, npr. četrti element izpišemo z ukazom »seznam !! 3«, saj Haskell indeksiranje elementov začneja z ničlo. Nadalje lahko izpišemo prvi element seznama, tj. »head(seznam)«, preostali del seznama (razen prvega elementa), tj. »tail(seznam)« ter maksimalno vrednost seznama, tj. »maximum(seznam)«. Haskell omogoča tudi zahtevnejše operacije s seznamami, tj. »pipe«. Zadnji primer prikazuje funkcijo, ki generira prvih deset števil ter jih pomnoži z 2, vse v eni vrstici.

Ker Haskell samostojno zaznava tip spremenljivk, uporabniku teh ni treba eksplicitno navajati. Kljub temu pa uporabnik tip spremenljivk lahko določi, npr. kot to prikazuje tabela 6. V levem stolpcu je prikazan primer za ročno določitev celega števila (angl. integer, krajše int) vhodni in izhodni spremenljivki. Prav tako lahko po vzoru desnega stolpca tip določimo tudi več vhodnim spremenljivkam.

Tabela 6. Ročno določanje tipa funkcije

<pre>'kvadriranje.hs' kvadriraj :: Integer -> Integer kvadriraj x = x * x *Main> kvadriraj 3 9</pre>	<pre>'sestejStevila.hs' sestejStevila :: Int -> Int -> Int -> Int sestejStevila x y z = x + y + z *Main> sestejStevila 1 2 3 6</pre>
---	---

3 OGRODJE YESOD

Yesod je prosto dostopno spletno ogrodje, namenjeno za razvoj varnih in visoko-zmogljivih spletnih aplikacij [6]. Spletno ogrodje Yesod izvira iz programskega jezika Haskell, kar pomeni da vključuje vse ugodne značilnosti, našteje v drugem poglavju [7]. Avtor ogrodja Yesod Michael Snoyman navaja glavne posebnosti programskega ogrodja Yesod [1]:

- varnost tipov (angl. type safety): točno lahko določimo pričakovane vhodne in izhodne tipe,
- zgoščenost (angl. concise code): Yesod omogoča uporabo pred-pripravljenih orodij (angl. scaffolding tools) in knjižnic (npr. iz repozitorija Hackage [11]) za zmanjšanje količine programske kode in pohitritev razvoja,
- učinkovitost (angl. performance): Yesod zagotavlja prepričljivo učinkovitost, tako prevajalnika GHC, ki je podvržen neprenehnemu izboljševanju, kakor tudi same arhitekture. Slednja z uporabo voda (angl. conduit) in graditelja (angl. builder) dovoljuje, da se programska koda izvaja v pomnilniku in s tem izogiba dolgotrajnemu dostopu do trdega diska. K temu priča dejstvo, da je eden izmed Haskell-ovih spletnih strežnikov, t.i. Warp eden najhitrejših kar jih trenutno obstaja [2],
- modularnost (angl. modularity): Yesod z domiselnimi idejami in razvojem Yesod-a spodbuja razvoj novih programskih, medsebojno združljivih in konsistentnih paketov, ki so velikokrat uporabni tudi izven okvirjev Haskell-a,
- trdni temelji (angl. a solid foundation): Yesod je deležen vzporednega razvoja več različnih programskih paketov za enako stvar, kar mu omogoča izbiro najučinkovitejše rešitve za dan problem. Dodatno razvijalci Yesod-a in Haskell-a stremijo k nenehnemu izboljševanju in poenostavljanju programskih paketov,
- arhitektura (angl. architecture): posebnost Yesoda je v tem, da za ustvarjanje HTML, CSS ali JavaScript programske kode uporablja posebne domensko specifične jezike, ki so kot celotna družina znani pod skupnim imenom Shakespearske šablone (angl. Shakespearean Templates). Vsaka šablona je poimenovana po Shakespearskih akterjih, npr. šablona za HTML se imenuje »Hamlet«, šablona za CSS sta »Lucius« in »Cassius«, šablona za JavaScript pa je poimenovana »Julius«. Družina Shakespearskih šablon bo v nadaljevanju prispevka predstavljena podrobneje.

3.1 Shakespearske šablone

Shakespearske šablone nudijo samodejno ustvarjanje HTML, CSS in JavaScript [1]. Hamlet je domensko specifični jezik, namenjen za ustvarjanje HTML. Nudi kompaktno in enostavno sintakso, brez uporabe zavitih oklepajev (angl. braceless), a s praznimi prostori (angl. white-spaces) za poravnavo (angl. indentation). Poleg ustvarjanja HTML kode omogoča tudi osnovne nadzorne bloke, kot npr. vejitve (tabela 8) in zanke. Hamlet je

najkompleksnejši izmed štirih naštetih domensko specifičnih jezikov [1], njegova sintaksa pa je prikazana v tabeli 7.

Tabela 7. Primerjava med HTML in Hamlet

<pre> 1 <body> 2 <p>Primer nastevanja</p> 3 4 Prvi 5 Drugi 6 7 <p><h3>Vecji naslov</p> 8 </body> </pre> HTML	<pre> 1 <body> 2 <p>Primer nastevanja 3 4 Prvi 5 Drugi 6 <p> 7 <h3>Vecji naslov </pre> Hamlet
--	---

Tabela 8. Primer vejitve Hamlet

<pre> \$if delovniDan <p>Gremo v službo. \$else <p>Ostanemo doma. </pre>
--

Lucius in Cassius sta domensko specifična jezika namenjena ustvarjanju CSS-a. Funkcionalno se slednja ne razlikujeta, sta si pa sintaktično različna. Prvi izmed njiju uporablja zavite oklepaje, medtem ko drugi prazne prostore za poravnavo. Tabela 9 prikazuje njuno osnovno sintakso.

Tabela 9. Primerjava med Lucius in Cassius [1]

<pre> section.blog { padding: 1em; border: 1px solid #000; h1 { color: #{headingColor}; background-image: url(@{MyBackgroundR}) } } </pre> Lucius (z zavitimi oklepaji)	<pre> section.blog padding: 1em border: 1px solid #000 h1 color: #{headingColor} background-image: url(@ </pre> Cassius (s praznimi prostori)
--	--

Julius je zadnji izmed domensko specifičnih jezikov, namenjen ustvarjanju JavaScript programske kode. V splošnem je zelo podoben samemu JavaScript-u, zato [1] navaja, da je Julius najenostavnejši izmed domensko specifičnih jezikov. Tabela 10 prikazuje njegovo sintakso.

Tabela 10. Prikaz domensko specifičnega jezika Julius [1]

<pre> 1 \$(function() { 2 \$("section.#{rawJS sectionClass}").hide(); 3 \$("#mybutton").click(function() {document.location = "@{ 4 SomeRouterR"}"); 5 ^{addBling} 6 }); </pre>

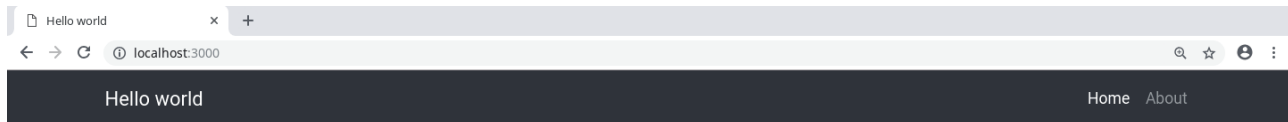
3.2 Uvodni program »Hello World« v ogrodju Yesod

Yesod je zelo obširno ogrodje, zato v prihajajočih podpoglavjih prikazujemo le njegove bistvene značilnosti. Sprva predstavljamo osnovni program »Hello World« (tabela 11), osnovan na čelnem ogrodju (angl. front-end framework) »Twitter Bootstrap« [19]. Za prevajanje programa uporabljamo prevajalnik GHC, lahko pa program zaženemo tudi z ukazom »runhaskell app.hs«, kjer »app.hs« predstavlja ime programa. S tem

zaženemo spletni strežnik Warp [2], čigar delovanje programa preizkusimo z obiskom na spletni strani <http://localhost:3000>, kot to prikazuje slika 2.

Tabela 11. Primer »Hello World« v spletnem ogrodju Yesod

<pre> 1 {-# LANGUAGE OverloadedStrings #-} 2 {-# LANGUAGE QuasiQuotes #-} 3 {-# LANGUAGE TemplateHaskell #-} 4 {-# LANGUAGE TypeFamilies #-} 5 {-# LANGUAGE MultiParamTypeClasses #-} 6 {-# LANGUAGE ViewPatterns #-} 7 8 import Yesod 9 import Yesod.Static 10 11 data App = App 12 { getStatic :: Static 13 } 14 15 mkYesod "App" [parseRoutes 16 / HomeR GET 17 /static StaticR Static getStatic 18] 19 20 instance Yesod App 21 22 getHomeR :: Handler Html 23 getHomeR = do 24 defaultLayout \$ do 25 setTitle "Hello world" 26 \$(whamletFile "hamlet/hello.hamlet") 27 28 main :: IO () 29 main = do 30 static@(Static settings) <- static "static" 31 warp 3000 \$ App static </pre>	Haskell language pragmas Vključitev Yesod knjižnic Delo s statičnimi datotekami Usmerjanje (angl. routing) Rutina (angl. handler function) Glavni program (zagon spletnega strežnika Warp)
--	--



Slika 2. Pogled v spletnem brskalniku

3.3 Gradniki

Nadaljujemo s pregledom Yesod gradnikov. Med razvojem spletne strani in aplikacij navadno usklajujemo tri različne tehnologije, tj. HTML, CSS in Javascript. Naštete se na spletni strani nahajajo na različnih lokacijah, npr. CSS v glavi in HTML v jedru spletne strani, kar dodatno otežuje organizacijo. Yesod za ta problem uvaja gradnike (angl. widgets). Ti omogočajo vključevanje različnih knjižnic, morebiti shranjenih lokalno, ali pa tudi oddaljeno. Primeri nekaterih elementov gradnikov so predstavljeni v tabeli 12.

Tabela 12. Primeri gradnikov

setTitle	Nastavi naslov spletne strani
addStylesheet	Poveži do lokalno shranjene slogovne predloge CSS
addStylesheetRemote	Poveži do oddaljeno shranjene slogovne predloge CSS
addCassius	Dodaj nekaj CSS značk
addHamlet	Dodaj Hamlet v značko body
addScript	Poveži do lokalno shranjene skripte
addScriptRemote	Poveži do oddaljeno shranjene skripte
addHtmlHead	Dodaj HTML v značko head

Tabela 13 prikazuje osnovni primer programske kode gradnikov, za več gradnikov pa je bralec povabljen k ogledu uradne dokumentacije [18]. Slika 3 predstavlja prikaz gradnikov v spletnem brskalniku.

Tabela 13. Prikaz programske kode za razvoj gradnika

<pre> 1 {-# LANGUAGE OverloadedStrings #-} 2 {-# LANGUAGE QuasiQuotes #-} 3 {-# LANGUAGE TemplateHaskell #-} 4 {-# LANGUAGE TypeFamilies #-} 5 6 import Yesod 7 8 data App = App 9 mkYesod "App" [parseRoutes 10 / DomovR GET 11] 12 instance Yesod App 13 14 getDomovR = defaultLayout \$ do 15 setTitle "OTS 2019" 16 toWidget [lucius h3 { color: blue; }] 17 addScriptRemote "https://ajax.googleapis.com/ajax/libs/jquery 18 /1.6.2/jquery.min.js" 19 toWidget 20 [julius 21 \$(function() { 22 \$("h3").click(function(){ 23 alert("Kliknili ste na naslov!"); 24 }); 25] 26 toWidgetHead 27 [hamlet 28 <meta name=keywords content="haskell, yesod, spletna 29 stran"> 30] 31 toWidget 32 [hamlet 33 <h3>Na tak nacin lahko vkljucimo vsebino 34] 35 [whamlet <h4>In se en nacin] 36 toWidgetBody 37 [julius 38 alert("JavaScript alert vkljucen v body"); 39] 40 main = warp 3000 App </pre>	Nastavi naslov spletne strani Poveži do oddaljene skripte Vpelji pojavno okno Določi meta-podatke spletne strani Ustvari odstavek Ustvari dodaten odstavek
--	--

Slika 3. Prikaz gradnika v spletnem brskalniku

3.4 Obrazci

Poslednja zanimivost, ki jo predstavljamo v tem prispevku, so Yesod obrazci (angl. forms). Obrazci so interaktivni deli spletne strani, v katere uporabnik lahko vpisuje podatke. Tabela 14 prikazuje primer programske kode za razvoj Yesod obrazca, medtem ko slika 4 rezultat, ki se pri tem pojavi. Levi stolpec tabele 14 prikazuje določevanje lastnega podatkovnega tipa spremenljivk, s katerimi bo uporabnik spletne strani operiral. Desni stolpec tabele 14 prikazuje dodane gradnike, ki bodo določili izgled spletne strani.

Tabela 14. Prikaz programske kode za razvoj obrazcev

<pre>data Oseba = Oseba { osebaIme :: Text , osebaPriimek :: Text , osebaKraj :: Text } deriving Show</pre>	<pre>osebaForm :: Html -> MForm Handler (FormResult Oseba, Widget) osebaForm = renderDivs \$ Oseba <\$> areq textField "Ime" Nothing <*> areq textField "Priimek" Nothing <*> areq textField "Kraj" Nothing</pre>
---	--

← → ↻ ⓘ localhost:3000

Prosimo izpolnite obrazec

Ime

Priimek

Kraj

Slika 4. Prikaz enostavnega obrazca Yesod v spletnem brskalniku

Z obrazci zaključujemo kratek pregled bistvenih značilnosti Yesod-a in v prihodnjem poglavju nadaljujemo na postavitvev aplikacije.

4 POSTAVITEV APLIKACIJE

Postavitev končne aplikacije je navadno zadnji, a nadvse kompleksen korak razvoja spletnih aplikacij. Pri tem ni izjema niti spletno ogrodje Yesod, čeravno se celotna skupnost raziskovalcev zavzema za poenostavitev postopka postavitve [14]. V Yesod-u postavitev poteka večplastno. Sprva s prevajalnikom prevedemo vso programsko kodo in pridobimo izvedljivo datoteko. To nato prenesemo na spletni strežnik, kamor obvezno priložimo tudi statične datoteke. V drugem koraku namestimo enega izmed postavitvenih ogrodij (angl. deployment engine) ali spletni strežnik. V tabeli 15 navajamo nekaj najbolj pogostih postavitvenih ogrodij, njihove prednosti ter slabosti.

Tabela 15. Yesod postavitvena ogrodja

Ogrodje	Glavne značilnosti	Prednosti	Pomankljivosti
Keter	Sistem za postavitvev spletnih aplikacij Yesod	Zagotavlja podporo SSL, samodejno zažene aplikacijo ob zagonu, spremlja procese kakor tudi ponovno zažene aplikacijo v primeru sesutja	Ni primerno ogrodje za manjše aplikacije
Warp	Visoko-zmogljiv spletni strežnik, v celoti napisan v Haskellu	Ni potrebe po namestniškem strežniku (angl. proxy server)	V primeru sesutja je potreben mehanizem za ponovni zagon aplikacije
Nginx	Vzvrtni namestniški strežnik (angl. reverse proxy)	Lažja migracija spletnih aplikacij med več strežniki	Zahteva obilo konfiguriranja
Kombinacija	Kombinacija več postavitvenih ogrodij in tehnik, npr. Warp in Keter	Možnosti prilagajanja	Zahtevano napredno znanje sistemske administracije v Linuxu, oteženo vzdrževanje

5 ZAKLJUČEK

V tem prispevku smo na kratko predstavili programski jezik Haskell in spletno ogrodje Yesod. Yesod je ogrodje, ki omogoča hiter razvoj spletnih aplikacijah, katere odlikujeta varnost in zmogljivost. Varnost je zagotovljena z notranjo zasnovo programskega jezika Haskell, saj tovrstne aplikacije po uspešnem prevajanju ne povzročajo napak med sprotnim izvajanjem programske kode, zmogljivost izvajanja programske kode pa dosega približno raven programskega jezika C. Ocenjujemo, da je Yesod zato zelo primeren za časovno kritične spletne aplikacije in spletne aplikacije s poudarjenimi varnostnimi zahtevami.

Ugotavljamo, da je postavitve osnovne spletne aplikacije v spletnem ogrodju Yesod zelo kompleksna. Potrebne so veščine systemske administracije v operacijskem sistemu Linux in poglobljeno znanje programskega jezika Haskell. Za vizualno privlačne spletne aplikacije je potrebno tudi dobro poznavanje Yesod gradnikov in obrazcev, nenazadnje pa tudi Shakespearskih šablon. Kljub temu pri zagonih osnovnih spletnih aplikacij opazamo nazorno pohitritev delovanja spletne strani.

6 LITERATURA

- [1] SNOYMAN Michael. Developing web applications with Haskell and Yesod. O'Reilly Media, Inc., 2012.
- [2] SNOYMAN Michael. "Warp: A Haskell web server." *IEEE Internet Computing* 15.3 (2011): 81-85.
- [3] LIPOVAČA Miran. *Learn you a haskell for great good!: a beginner's guide*. No Starch Press, 2011.
- [4] HUDAK Paul, FASEL H. Joseph, PETERSON John. "A gentle introduction to Haskell." *Sigplan Notices* 27.5 (1992): T1-T53.
- [5] HUDAK Paul. "Building domain-specific embedded languages." *ACM Computing Surveys* 28.4es (1996): 196.
- [6] DVOŘÁK Pavel. Web applications in Haskell. Diss. Masarykova univerzita, Fakulta informatiky, 2012.
- [7] SNOYMAN Michael. Developing Web Apps with Haskell and Yesod: Safety-driven Web Development. "O'Reilly Media, Inc.", 2015.
- [8] EPSTEIN Jeff, BLACK P. Andrew, PEYTON-JONES Simon. "Towards Haskell in the Cloud." *ACM SIGPLAN Notices*. Vol. 46. No. 12. ACM, 2011.
- [9] IZBICKI Michael. "Hlearn: A Machine Learning Library for Haskell." Proceedings of The Fourteenth Symposium on Trends in Functional Programming, Brigham Young University, Utah. 2013.
- [10] ANNENKOV Danil, ELSMAN Martin. "Certified Compilation of Financial Contracts." *Proceedings of the 20th International Symposium on Principles and Practice of Declarative Programming*. ACM, 2018.
- [11] <https://hackage.haskell.org/>, Welcome to Hackage!, obiskano 13. 5. 2019.
- [12] https://wiki.haskell.org/Haskell_in_practice, Haskell in practice, obiskano 10. 5. 2019.
- [13] [https://en.wikipedia.org/wiki/Haskell_\(programming_language\)](https://en.wikipedia.org/wiki/Haskell_(programming_language)), Haskell (programming language), obiskano 10. 5. 2019.
- [14] <https://www.yesodweb.com/book>, Yesod Web Framework, obiskano 9. 5. 2019.
- [15] <https://www.tiobe.com/tiobe-index/>, TIOBE Index for May 2019, obiskano 13. 5. 2019.
- [16] [https://en.wikipedia.org/wiki/Yesod_\(web_framework\)](https://en.wikipedia.org/wiki/Yesod_(web_framework)), Yesod (web framework), obiskano 10. 5. 2019.
- [17] <https://www.forbes.com/sites/quora/2017/07/11/why-arent-more-programmers-using-haskell/>, Why Arent't More Programmers Using Haskell?, obiskano 10. 5. 2019.
- [18] <https://hackage.haskell.org/package/yesod-core-1.1.1/docs/Yesod-Widget.html>, Yesod Widget, obiskano 14. 5. 2019.
- [19] <https://github.com/firefly-cpp/Yesod-bootstrap-skeleton>, GitHub Yesod-bootstrap-skeleton, obiskano 14. 5. 2019.