

# Razvoj programskog jezika Elixir

Seminarski rad u okviru kursa  
Dizajn programskih jezika  
Matematički fakultet

Ime i prezime  
kontakt email adresa

4. novembar 2019

## Sažetak

U ovom radu prikazuju se osnove razvoja programskog jezika Elixir. Ukratko su opisani jezici koji su najviše uticali na njegov razvoj: Erlang, Python, Haskell, Ruby i Clojure. Za svaki jezik date su osnovne informacije kao i način na koji je taj jezik uticao na osobine i razvoj jezika Elixir. Prikazano je razvojno stablo koje uključuje ove jezike.<sup>1</sup>

## Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
<b>2</b>	<b>Osnovno o Elixiru</b>	<b>2</b>
<b>3</b>	<b>Razvojno stablo</b>	<b>2</b>
3.1	Erlang . . . . .	3
3.2	Python . . . . .	4
3.3	Haskell . . . . .	4
3.4	Ruby . . . . .	4
3.5	Clojure . . . . .	5
<b>4</b>	<b>Zaključak</b>	<b>5</b>
	<b>Literatura</b>	<b>6</b>

---

<sup>1</sup>Tekst u ovom radu je najvećim delom preuzet iz master rada Milene Dukanac [4].

# 1 Uvod

U oblasti razvoja programskih jezika stalno se dešavaju promene. Velika pažnja i značaj se pridaje razvoju novih programskih jezika koji omogućavaju brže i lakše programiranje, bržu obradu velike količine podataka i podršku za konkurentne procese. Upravo sa tim ciljem, u toku prethodnih godina nastao je veliki broj novih programskih jezika i razvojnih okruženja za njih. Jedan od novih jezika koji je nastao 2011. godine je *Elixir*. Za veoma kratko vreme postao je izuzetno popularan i danas ga veliki broj kompanija koristi za svoje projekte i softverska rešenja. Neki od razloga za popularnost jezika Elixir su brza obrada velike količine podataka, skalabilnost, podrška za konkurentne procese i veoma visoka tolerancija na greške. Logo jezika Elixir prikazan je na slici 1.



Slika 1: Logo programskog jezika Elixir

## 2 Osnovno o Elixiru

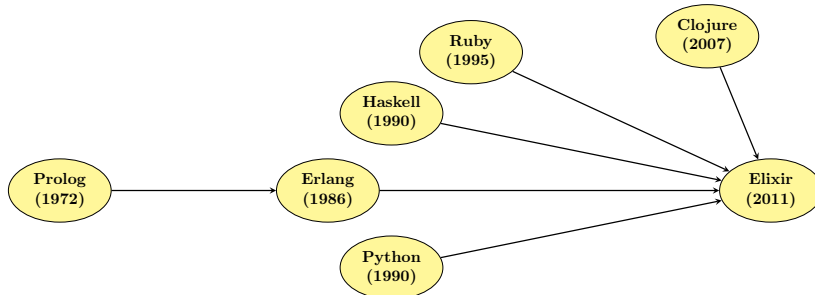
Elixir je funkcionalan programski jezik nastao 2011. godine [5, 13]. Njegovim tvorcem se smatra José Valim (engl. *José Valim*). Elixir je dizajniran za izgradnju skalabilnih i lako održivih aplikacija. Posедуje jednostavnu i modernu sintaksu. Zbog svoje funkcionalne prirode, izuzetno dobre podrške za rad u distribuiranim sistemima i tolerancije na greške koja je na jako visokom nivou, u Elixir-u je rađeno mnogo zanimljivih projekata iz oblasti robotike. Takođe se uspešno koristi u razvoju veba i u domenu softvera za uređaje sa ugrađenim računarom (engl. *embedded software*).

Hosé Valim je tokom 2010. godine bio zaposlen u kompaniji *Plataformatec* [9] i radio je na poboljšanju performansi okruženja *Ruby on Rails* na višezgarnim sistemima. Shvatio je da Ruby nije bio dovoljno dobro dizajniran da reši problem konkurentnosti, pa je započeo istraživanje drugih tehnologija koje bi bile prihvatljivije. Tako je otkrio Erlang i upravo ga je interesovanje prema virtuelnoj mašini jezika Erlang podstaklo da započne pisanje jezika Elixir. Uticaj projekta na kome je do tada radio odrazio se na to da Elixir ima sintaksu koja je nalik na sintaksu jezika Ruby. Ovaj jezik se pokazao veoma dobro pri upravljanju milionima simultanih konekcija: u 2015. je zabeleženo upravljanje nad dva miliona *WebSocket* konekcija, dok je u 2017. za skalirani Elixir zabeležena obrada pet miliona istovremenih korisnika. Elixir se danas koristi u velikim kompanijama, kao što su *Discord*, *Motorola Solutions* i *Pinterest* [8].

## 3 Razvojno stablo

Na nastanak jezika Elixir je najviše uticao programski jezik **Erlang**. Pri njegovom kreiranju značajnu ulogu u smislu sintakse imao je programski jezik **Ruby**, a iz jezika kao što su **Python**, **Haskell** i **Clojure**

je preuzeo mnoge koncepte. Razvojno stablo jezika Elixir može se videti na slici 2.



Slika 2: Razvojno stablo jezika Elixir

### 3.1 Erlang

Firma Erikson je 1981. godine oformila novu laboratoriju **Erikson CSLab** (engl. *The Ericsson CSLab*) sa ciljem da predlaže i stvara nove arhitekture, koncepte i strukture za buduće softverske sisteme [1]. Jedan od zadataka novonastale laboratorije bio je dodavanje konkurentnih procesa u programski jezik **Prolog**<sup>2</sup> i njegovo unapređivanje. Prolog predstavlja začetak novog programskog jezika koji je 1987. godine nazvan **Erlang**. Ime je nastalo zahvaljujući inicijativi zaposlenih koji su radili na telefonskim prekidačima, a za koje je jezik dizajniran. Naime, oni su predložili da jezik nosi ime Erlang u čast danskom matematičaru i inženjeru Agneru Krarupu Erlangu (engl. *Agner Krarup Erlang*), a što je ujedno odgovaralo i skraćenici od "**E**ricsson **L**anguage". Erlang se smatrao dijalektom Prologa sve do 1990. godine, kada je postao potpuno samostalan programski jezik sa sopstvenom sintaksom. Međutim, zadržao je neke delove sintakse i koncepte iz Prologa (promenljive počinju velikim slovom, svaka funkcionalna celina se završava tačkom, poklapanje obrazaca (engl. *pattern matching*)).

Nakon mnogo godina rada nastajale su sve brže, bolje i stabilnije verzije jezika, kao i **standardna biblioteka OTP** (engl. *The Open Telecom Platform*) [7]. Od decembra 1998. godine, kada su postali deo slobodnog softvera (engl. *open source software*), Erlang i OTP se mogu slobodno preuzeti sa zvaničnog sajta jezika Erlang [7]. Erlang dobija široko prihvatanje pojavom višejezgaranih procesora i njihovog novog skalabilnog pristupa konkurentnosti. Erlang je funkcionalan jezik idealan za svaku situaciju u kojoj su paralelnost, tolerancija na greške i brz odziv neophodni [2], te se koristi u velikom broju kompanija za razvoj njihovih glavnih softverskih rešenja (npr. Erikson (engl. *Ericsson*), Motorola, Votsap (engl. *WhatsApp*), Jahu (engl. *Yahoo!*), Fejsbuk (engl. *Facebook*)).

Elixir je preuzeo izmenjenu Erlangovu sintaksu i dopunjenu Erlangovu standardnu biblioteku. Pokreće se na vituelnoj mašini jezika Erlang, što

<sup>2</sup>Prolog (engl. *PROgramming in LOGic*) je deklarativan programski jezik namenjen rešavanju zadataka simboličke prirode. Prolog se temelji na teorijskom modelu logike prvog reda. Početkom 1970-ih godina **Alen Kolmerur** (engl. *Alain Colmerauer*) i **Filip Rusel** (engl. *Philippe Roussel*) na Univerzitetu u Marselju, zajedno sa **Robertom Kovalskim** (engl. *Robert Kowalski*) sa Departmana veštačke inteligencije na Univerzitetu u Edinburgu, razvili su osnovni dizajn jezika Prolog.

znači da je nasledio i sve karakteristike Erlang platforme koja postoji već godinama i koja se pokazala pouzdanim rešenjem za skalabilne aplikacije, kao što su **konkurentnost** i **tolerisanje grešaka** [12]. Elixir je nadomestio mnoge koncepte koji su nedostajali jeziku Erlang. Neki od njih su **metaprogramiranje**<sup>3</sup>, **polimorfizam**, **makroi** i **podrška za alate**. Elixir poseduje podrazumevano okruženje, takozvani **Kernel**, koji obezbeđuje podršku za osnovne tipove i funkcionalnosti jezika.

### 3.2 Python

**Python** je interpretirani jezik opšte namene čiji tvorac je Guido van Rossum (engl. *Guido van Rossum*) [10]. Krajem 1980-ih je koncipiran kao naslednik jezika **ABC** [14], a prvi put je objavljen 1991. godine. Fiziološka dizajna jezika Python naglašava čitljivost koda. Njegove jezičke konstrukcije i objektno-orijentisani pristup imaju za cilj da pomognu programerima da napišu jasan i logičan kôd za male i velike projekte. Python je dinamički tipiziran jezik i poseduje sistem za prikupljanje smeća (engl. *garbage collector*). Podržava više paradigmi programiranja uključujući proceduralno, objektno-orijentisano i funkcionalno programiranje. Python interpreteri su dostupni za mnoge operativne sisteme. Globalna zajednica programera razvija i održava referentnu implementaciju otvorenog koda **CPython**. Neprofitna organizacija *The Python Software Foundation* upravlja resursima i usmerava ih za razvoj jezika Python i CPython. Jedna od osobina koje je Elixir nasledio od Python-a je podrška za dokumentaciju u vidu dokumentacionih stringova (engl. *docstrings*) koji omogućavaju povezivanje dokumentacije sa modulima, funkcijama, klasama, metodama.

### 3.3 Haskell

**Haskell** je čisto funkcionalni jezik koji je statički tipiziran [6]. Nazvan je po Haskellu Bruks Kariju (engl. *Haskell Brooks Curry*), čiji rad u oblasti matematičke logike služi kao osnova za sve funkcionalne jezike. Haskell je zasnovan na lambda računu, pa se stoga lambda koristi kao logo jezika. Nudi kratak, jasan i održiv kôd, mali procenat grešaka i veliku pouzdanost. Stoga je pogodan za pisanje velikih softverskih sistema, jer njihovo održavanje čini lakšim i jeftinijim. Jedna od karakteristika koje je Elixir preuzeo od ovog jezika je lenjo izračunavanje.

### 3.4 Ruby

**Ruby** je dinamički tipiziran programski jezik otvorenog koda nastao 1995. godine [11]. Kod ovog programskog jezika fokus je na jednostavnosti i produktivnosti. Ruby ima elegantnu sintaksu koja je prirodna za čitanje i lako pisanje. Ruby je interpretirani programski jezik, što znači da se izvorni kôd prevodi u kôd razumljiv računaru prilikom svakog izvršavanja programa. Interpretirani programski jezici su sporiji od kompajliranih, ali su fleksibilniji i potrebno je kraće vreme za izradu programa. Međutim, sve više iskusnih Ruby programera se okreće Elixir-u. Zapravo, Elixir je prvi jezik nakon Ruby-ja koji zaista brine o lepoti koda i korisničkom iskustvu vezanom za jezik, biblioteke i ekosistem.

---

<sup>3</sup>Tehnika koja omogućava da programi posmatraju druge programe kao svoje podatke i na taj način čitaju i modifikuju i njihov i svoj kôd u vreme izvršavanja.

```

1000 class Concat
1001   attr_reader :value
1002
1003   def initialize(value)
1004     @value = value
1005   end
1006
1007   def join(another_value)
1008     value + another_value
1009   end
1010 end
1011
1012 irb> Concat.new("Hello ").join("
    world!")
irb> => "Hello world!"

```

Listing 1: Ruby

```

1000 defmodule Concat do
1001   def join(value, another_value)
1002     do
1003       value <> another_value
1004     end
1005 end
1006
1007 iex> Concat.new("Hello ", "world!")
iex> "Hello world!"

```

Listing 2: Elixir

Slika 3: Kôd za nadovezivanje dva stringa u jezicima Ruby i Elixir

Ruby je imao veliki uticaj na sintaksu programskog jezika Elixir. Na slici 3 se nalaze delovi koda napisani u jeziku Ruby i jeziku Elixir, koji imaju dosta sličnosti, a čiji je rezultat izvršavanja isti - dva nadovezana stringa. U jeziku Ruby se definiše klasa *Concat* koja ima polje *value*, funkciju *initialize* koja se poziva pri kreiranju objekta klase radi inicijalizacije polja *value* i funkciju *join* koja vrši nadovezivanje dva stringa. U Elixir-u se umesto klase definiše modul *Concat* koji sadrži samo funkciju *join* koja vrši nadovezivanje dva stringa. Uočavaju se sličnosti u sintaksi koje su ilustrovane ovim primerom pri definisanju klase/modula, funkcija (ključne reči *def* i *end*), zatim pri nadovezivanju stringova (operatori *+* i *<>*) i pozivanju funkcija (ime klase/modula za kojim sledi tačka).

### 3.5 Clojure

**Clojure** je dinamički tipiziran programski jezik opšte namene nastao 2007. godine [3]. Njegov tvorac je Rič Hiki (engl. *Rich Hickey*). Clojure kombinuje pristupačnost i interaktivni razvoj skriptnog jezika sa efikasnom i robusnom infrastrukturom za višenitno programiranje. On je kompajlirani jezik, ali je i dalje potpuno dinamički tipiziran - svaka funkcija koju podržava Clojure je podržana u toku izvršavanja. Predstavlja dijalekt Lisp-a<sup>4</sup> i deli njegovu filozofiju *code-as-data* (program je funkcija koja se izvršava nad podacima) i moćan makro sistem. Clojure je pretežno funkcionalni programski jezik i sadrži bogat skup nepromenljivih i postojećih struktura podataka. Elixir je preuzeo neke od najboljih Clojure karakteristika - efikasne, nepromenljive strukture podataka, opcionalno lenjo izračunavanje i protokole.

## 4 Zaključak

U ovom tekstu ukratko su predstavljene osnove razvoja programskog jezika Elixir. Prikazani su jezici koji su najviše uticali na njegov nastanak

<sup>4</sup>Lisp je programski jezik zasnovan na matematičkoj teoriji rekurzivnih funkcija (u kojoj se funkcija pojavljuje u sopstvenoj definiciji), a Lisp program je funkcija koja se primenjuje na podatke. Ime LISP je nastalo od „LISt Processor“, a povezane liste su jedan od glavnih tipova podataka. Osnova Lisp-a je funkcionalno programiranje, ali se Lisp zbog raznih drugih svojstava smatra multiparadigmatiskim programskim jezikom.

i razvoj, i prikazano je njegovo elementarno razvojno stablo. Za dodatne informacije i praćenje daljeg razvoja, kao početnu odrednicu najbolje je koristiti zvaničnu stranu programskog jezika Elixir [5].

## Literatura

- [1] Joe Armstrong. *Making reliable distributed systems in the presence of software errors*. PhD thesis, The Royal Institute of Technology, 2003.
- [2] Francesco Cesarini and Simon Thompson. *Erlang Programming*. O'Reilly Media, June 2009.
- [3] Clojure. Zvanična stranica programskog jezika Clojure. <https://clojure.org/>.
- [4] Milena Dukanac. Jezik Elixir sa primenom u sekvencioniranju genoma, 2019. Master rad.
- [5] Elixir. Zvanična stranica programskog jezika Elixir. <https://elixir-lang.org/>.
- [6] Haskell. Zvanična stranica programskog jezika Haskell. <https://www.haskell.org/>.
- [7] Erlang OTP. Zvanična stranica programskog jezika Erlang. <http://erlang.org>.
- [8] Charles Petit. A brief history of erlang and elixir. *Coder Stories*, January 2019.
- [9] Platformatec. Zvanična stranica kompanije Platformatec. <http://plataformatec.com.br/>.
- [10] Python. Zvanična stranica programskog jezika Python. <https://www.python.org/>.
- [11] Ruby. Zvanična stranica programskog jezika Ruby. <https://www.ruby-lang.org/en/>.
- [12] Dave Thomas. *Programming Elixir 1.3*. The Pragmatic Bookshelf, 2016.
- [13] Dave Thomas. *Programming Elixir ≥ 1.6: Functional />Concurrent />Pragmatic />Fun*. The Pragmatic Bookshelf, 2018.
- [14] Rolf Zwart, April 2018. <https://reinout.vanrees.org/weblog/2018/04/25/origin-of-python-abc.html>.