01 UPORABNA INFORMATIKA

2023 ‹ ŠTEVILKA 1 ‹ LETNIK XXXI ‹ ISSN 1318-1882

U P O R A B N A
I N F O R M A T I K A
2023 ŠTEVILKA 1 JAN/FEB/MAR LETNIK XXXI ISSN 1318-1882

V spomin

Strokovni prispevki

Informacije

U P O R A B N A
I N F O R M A T I K A

2023 ŠTEVILKA 1 JAN/FEB/MAR LETNIK XXXI ISSN 1318-1882

Ustanovitelj in izdajatelj
Slovensko društvo INFORMATIKA
Litostrojska cesta 54, 1000 Ljubljana

Predstavnik
Niko Schlamberger

Odgovorni urednik
Mirjana Kljajić Borštnar

Tehnični urednik
Slavko Žitnik

Lektoriranje angleških izvlečkov
Marvelingua (angl.)

Oblikovanje
KOFEIN DIZAJN, d. o. o.

Prelom in tisk
Boex DTP, d. o. o., Ljubljana

Naklada
110 izvodov

Naslov uredništva
Slovensko društvo INFORMATIKA
Uredništvo revije Uporabna informatika
Litostrojska cesta 54, 1000 Ljubljana
www.uporabna-informatika.si

Revija izhaja četrtletno. Cena posamezne številke je 20,00 EUR. Letna naročnina za podjetja 85,00 EUR, za vsak nadaljnji izvod 60,00 EUR, za posameznike 35,00 EUR, za študente in seniorje 15,00 EUR. V ceno je vključen DDV.

Revija Uporabna informatika je od številke 4/VII vključena v mednarodno bazo INSPEC.

Revija Uporabna informatika je pod zaporedno številko 666 vpisana v razvid medijev, ki ga vodi Ministrstvo za kulturo RS.

Revija Uporabna informatika je vključena v Digitalno knjižnico Slovenije (dLib.si).

Vabilo avtorjem

V reviji Uporabna informatika objavljamo kakovostne izvirne prispevke domačih in tujih avtorjev z najširšega področja informatike, ki se nanašajo tako na poslovanju podjetij, javno upravo, družbo in posameznika. Prispevki so lahko znanstvene, strokovne ali informativne narave, še posebno spodbujamo objavo interdisciplinarnih prispevkov. Zato vabimo avtorje, da prispevke, ki ustrezajo omenjenim usmeritvam, pošljejo uredništvu revije po elektronski pošti na naslov ui@drustvo-informatika.si.

Avtorje prosimo, da pri pripravi prispevka upoštevajo navodila, ki so objavljena na naslovu http://www.uporabna-informatika.si.

Za kakovost prispevkov skrbi mednarodni uredniški odbor. Prispevki so anonimno recenzirani, o objavi pa na podlagi recenzij samostojno odloča uredniški odbor. Recenzenti lahko zahtevajo, da avtorji besedilo spremenijo v skladu s priporočili in da popravljeni prispevek ponovno prejmejo v pregled. Sprejeti prispevki so pred izidom revije objavljeni na spletni strani revije (predobjava), še prej pa končno verzijo prispevka avtorji dobijo v pregled in potrditev. Uredništvo lahko še pred recenzijo zavrne objavo prispevka, če njegova vsebina ne ustreza vsebinski usmeritvi revije ali če prispevek ne ustreza kriterijem za objavo v reviji.

Pred objavo prispevka mora avtor podpisati izjavo o avtorstvu, s katero potrjuje originalnost prispevka in dovoljuje prenos materialnih avtorskih pravic. Avtorji prejmejo enoletno naročnino na revijo Uporabna informatika, ki vključuje avtorski izvod revije in še nadaljnje tri zaporedne številke. S svojim prispevkom v reviji Uporabna informatika boste pomagali k širjenju znanja na področju informatike. Želimo si čim več prispevkov z raznoliko in zanimivo tematiko in se jih že vnaprej veselimo

Uredništvo revije

Navodila avtorjem člankov

Članke objavljamo praviloma v slovenščini, članke tujih avtorjev pa v angleščini. Besedilo naj bo jezikovno skrbno pripravljeno. Priporočamo zmernost pri uporabi tujk in, kjer je mogoče, njihovo zamenjavo s slovenskimi izrazi. V pomoč pri iskanju slovenskih ustreznic priporočamo uporabo spletnega terminološkega slovarja Slovenskega društva Informatika, Islovar (www.islovar.org).

Znanstveni prispevek naj obsega največ 40.000 znakov, kratki znanstveni prispevek do 10.000 znakov, strokovni članki do 30.000 znakov, obvestila in poročila pa do 8.000 znakov.

Prispevek naj bo predložen v urejevalniku besedil Word (*.doc ali *.docx) v enojnem razmaku, brez posebnih znakov ali poudarjenih črk. Za ločilom na koncu stavka napravite samo en presledek, pri odstavkih ne uporabljajte zamika.

Naslovu prispevka naj sledi polno ime vsakega avtorja, ustanova, v kateri je zaposlen, naslov in elektronski naslov. Sledi naj povzetek v slovenščini v obsegu 8 do 10 vrstic in seznam od 5 do 8 ključnih besed, ki najbolje opredeljujejo vsebinski okvir prispevka. Sledi naj prevod naslova povzetka in ključnih besed v angleškem jeziku. V primeru, da oddajate prispevek v angleškem jeziku, velja obratno. Razdelki naj bodo naslovljeni in oštevilčeni z arabskimi številkami.

Slike in tabele vključite v besedilo. Opremite jih z naslovom in oštevilčite z arabskimi številkami. Na vsako sliko in tabelo se morate v besedilu prispevka sklicevati in jo pojasniti. Če v prispevku uporabljate slike ali tabele drugih avtorjev, navedite vir pod sliko oz. tabelo. Revijo tiskamo v črno-beli tehniki, zato barvne slike ali fotografije kot original niso primerne. Slikam zaslonov se v prispevku izogibajte, razen če so nujno potrebne za razumevanje besedila. Slike, grafikoni, organizacijske sheme ipd. naj imajo belo podlago. Enačbe oštevilčite v oklepajih desno od enačbe.

V besedilu se sklicujte na navedeno literaturo skladno s pravili sistema IEEE navajanja bibliografskih referenc, v besedilu to pomeni zaporedna številka navajenega vira v oglatem oklepaju (npr. [1]). Na koncu prispevka navedite samo v prispevku uporabljeno literaturo in vire v enotnem seznamu, urejeno po zaporedni številki vira, prav tako v skladu s pravili IEEE. Več o sistemu IEEE, katerega uporabo omogoča tudi urejevalnik besedil Word 2007, najdete na strani https://owl.purdue.edu/owl/research_and_citation/ieee_style/ieee_general_format.html.

Prispevku dodajte kratek življenjepis vsakega avtorja v obsegu do 8 vrstic, v katerem poudarite predvsem strokovne dosežke.

In memoriam:
Iztok Lajovic, univ. dipl. ing.

Spoštovani kolega se je rodil 4. julija 1944 v Ljubljani. Od nas se je poslovil 6. oktobra 2022.

Z informatiko in računalništvom se je začel ukvarjati že med študijem elektrotehnike. Njegovo delo na tem področju lahko razdelimo na razvojno-raziskovalno, strokovno, vodstveno in pedagoško. V času svoje zaposlitve na Institutu »Jožef Stefan« (od leta 1970 do leta 1975) je v skupini prof. dr. Antona P. Železnikarja raziskoval programske jezike in računalniška omrežja. V okviru Evropske gospodarske skupnosti je vzpostavil računalniške omrežne povezave med raziskovalnimi institucijami v Evropi. V tem obdobju je bil tudi med prvimi predavatelji predmeta računalništvo na slovenskih srednjih šolah.

Med leti 1976 in 1980 je bil direktor Elektronskega računskega centra v podjetju Tekstil Commerce. Med leti 1981 in 1990 je bil sprva pri Novi Ljubljanski banki Gospodarski banki vodja razvoja računalniške opreme, za tem pa pri NLB d.d. direktor sektorja razvoja programske opreme. Od leta 1991 je bil ustanovitelj in direktor podjetja KreS – Kreativni sistemi. Ves čas se je aktivno udeleževal konferenc iz področja informatike in računalništva doma in na tujem ter bil vabljeni predavatelj na številnih fakultetah. Je ustanovni član in nosilec Zlatega častnega znaka Slovenskega društva INFORMATIKA.

Številne originalne rešitve na področju informatike in računalništva nosijo njegov inovativni pečat. V tej zvezi naj posebej izpostavim programski sistem PANORAMA, ki udejanja izviren pristop k obravnavi in preiskovanju obsežnih podatkovnih zbirk. Koncipiran je kot nadgradnja relacijskih datotek, ki mrežno povezuje poljubno število datotek. Vsebino datotek in povezave med njimi upravlja podatkovni slovar. Je odprt sistem, ki omogoča uporabniku poljubno dopolnjevanje in nedestruktivno preoblikovanje datotek, ne da bi za te akcije potreboval znanja s področja podatkovnih baz. Uporabniški vmesnik sledi principom asociativnega razmišljanja in omogoča neomejeno potovanje po informacijskem prostoru.

Poleg njegove strokovne ustvarjalnosti naj omenim še njegovo umetniško ustvarjalnost. Kot srednješolec, je poleg klasične gimnazije, obiskoval tudi srednjo glasbeno šolo. Njegov instrument je bil klarinet. Lepo pa ga je bilo poslušati tudi za klavirjem. Uglasbil je več pesmi Josipa Murna Aleksandrova. Je avtor tudi drugih krajših glasbenih del, ki so bila javno izvajana. Pel je v zborih, tako v Akademskem pevskem zboru, Študentskem oktetu in pevskem zboru Lipa. Neredko je nastopal kot korepetitor in dirigent. Ustvarjalen je bil tudi na likovnem področju, kjer se je posvečal upodabljanju krajine v tuš tehniki.

Kolega Iztok Lajovic je zgled pokončnega človeka, ki je znan po svoji predanosti stvarem, za katere je sodil, da so pomembe za ožje ali širše okolje, čeprav niso bile vedno njemu v prid.

Prof. dr. Vladislav Rajkovič, zasl. prof.

Velepodatki – 5 V-jev v praktičnih primerih

Jure Jeraj[1], Urška Nered[1], Stevanče Nikoloski[1,2]
[1] Result, d. o. o., Celovška 182, Ljubljana
[2] Fakulteta za ekonomijo in informatiko, Univerza v Novem mestu, Na Loko 2, 8000, Novo mesto
jure.jeraj@result.si, urska.nered@result.si, stevance.nikoloski@result.si

Izvleček

Vsak posameznik in podjetje dimenzijo velepodatkov dojema malce drugače. A velepodatki se ne merijo kot 5 diskov podatkov, niti kot 5 sodov podatkov. Tudi pretočnih podatkov ne merimo z litri. Velepodatki so miselnost, ki predstavlja temelje za podatkovno gnano poslovanje. Da bi velepodatke resnično uvedli in izkoristili v poslovanju, jih moramo najprej razumeti. Le tako si lahko od njih obetamo sobivanje posla in tehnologije ter ustvarjanje dodane vrednosti.
V tem članku predstavljamo vse dimenzije velepodatkov (t. i. 5 V) in njihovo rabo osvetljujemo na praktičnih primerih. Predstavljamo širši ekosistem velepodatkov ter temelje za gradnjo okolja za velepodatke.

Ključne besede: velepodatki, obdelava v realnem času, podatkovne platforme, pretočni podatki, računalništvo v oblaku

Big Data – 5 V's in practical cases

Abstract

Every individual and company perceives the dimensions of big data a bit differently. Big data is not measured as 5 hard drives nor as 5 data barrels, neither is flow of data measured in litres. Big data is a mindset that forms the foundation for data-driven business. For mass data to be truly introduced to and used in business, one must first understand it. Only then can we expect business and technology to co-exist and deliver added value. In this article, I present all dimensions of big data (i.e., 5 V's) and shed light on its use in practical examples. I also present the broader ecosystem of big data and the foundations for building an environment for big data.

Keywords: Big data, real-time computing, data platforms, data streaming, cloud computing

1 UVOD

Izraz velepodatki (ang. Big Data) je v današnjem času zelo zanimiv in privlačen, a hkrati izredno zavajajoč. Nakazuje namreč na nekaj modernega (»Big« in še »Data«), hkrati pa namiguje, da imamo opravka zgolj z izrazito veliko količino podatkov. Vendar se velepodatki ne merijo zgolj s predponami tera-, peta-, eksa- in podobnimi, velika količina podatkov je le ena izmed njihovih lastnosti. So namreč še mnogo več.

V obstoječi literaturi lahko najdemo veliko različnih definicij koncepta »velepodatki«, ki jih raziskovalci in praktiki (inženirji, razvijalci) uporabljajo v svojih razlagah. Na primer, Cox & Ellsworth (1997) sta velepodatke omenila kot obseg oziroma količino znanstvenih podatkov za vizualizacije. Manyika in drugi (2011) definirajo velepodatke kot »količino podatkov, ki presega zmogljivost tehnologije za učinkovito hrambo, vodenje in procesiranje«. Hashem in drugi (2015) pišejo, da so velepodatki nabor tehnik in tehnologij, ki zahtevajo nove oblike integracije za odkrivanje velikih skritih vrednosti iz raznolikih in kompleksnih obsežnih podatkov.

Največkrat izpostavljena osnovna definicija velepodatkov pravi, da gre za kompleksne podatkovne strukture velikega obsega, ki jih ni moč obdelati na klasičen in tradicionalen način (Awanish, 2021). Po-

manjkljivost te definicije je že v načinu primerjave. Kaj posameznik razume kot klasičen in tradicionalen način? Je to nekaj, kar je bilo vrhunec tehnologije pred 10 leti? Pred 5 leti? Včeraj? Tehnologija se namreč izredno hitro razvija, zato preveč posplošena razlaga ni dobrodošla.

Napačno ali pomanjkljivo razumevanje velepodatkov ne spravlja v zadrego le strokovnjakov ob tehničnih razpravah. Precej večja težava je, kadar zaradi napačnega razumevanja podjetje prehitro zavrne implementacijo orodji in tehnik za izrabo velepodatkov v svoj poslovni model ali pa se prehitro zadovolji z zgolj vpeljavo podatkovne analitike na veliki količini podatkov, npr. z obdelavo 100 milijonov zapisov v podatkovni bazi. Kaj pa dejansko delamo z velepodatki? Že od nekdaj podatke obdelujemo in velepodatki v tem pogledu niso izjema. Razlika je v razvoju sodobnih, ekstremnih tehnik, tehnologij, orodij in konceptov za delo z njimi.

Eden izmed glavnih pogojev, da podjetje postane podatkovno gnano podjetje, je, da je uporaba in izraba velepodatkov poslovna odločitev, za katero trdno stoji miselnost organizacije. Brez podatkovno razvojne miselnosti in dobre podatkovne pismenosti bo imelo podjetje oziroma organizacija ali skupnost velike težave pri uspešni izrabi velepodatkov. Odločitev o uvedbi tehnologij za obdelavo velepodatkov ni tehnična, temveč predvsem poslovna odločitev.

Poučen primer take miselnosti je projekt LoginEko fundacije Login (Login5 Foundation, 2022). Glavni cilj fundacije je »oblikovati prihodnost kmetijstva nove generacije«. Poslovni model predvideva »vzpostavitev novega modela trajnostnega eko kmetijstva velikih razsežnosti«. Temelji na konceptu podatkovno gnanega poslovanja. Snovalci pojasnjujejo, da »podatkovno gnano kmetijstvo« zanje pomeni:

- zajem podatkov iz vseh možnih senzorjev,
- aktualne posnetke vseh polj s pomočjo brezpilotnih letalnikov (dronov),
- zbiranje podatkov mehanizacije v realnem času,
- napredne vremenske postaje na vseh mikrolokacijah,
- sledenje mikroflore tal,
- sledenje vsem kmetijskim dejavnostim,
- centralni informacijski sistem,
- odločanje in učenje na podlagi zbranih podatkov.

To je primer pristopa, ki v celoti podpira uvajanje tehnologij za obdelavo velepodatkov. Pri tem se tehnologije ne gleda zgolj kot strošek, temveč predvsem kot rešitev, ki ustvarja dodano vrednost.

2 VELEPODATKI, OPREDELJENI KOT SKUPEK 5 V-JEV

Podjetja z visoko podatkovno zrelostjo in razvito podatkovno kulturo so se ob prelomu stoletja začela aktivno ukvarjati s snovanjem poenostavljenega razumevanja koncepta »velepodatki«, ki bi ga lahko vpeljali v korporativno podatkovno upravljanje. Eden prvih, Doug Laney iz Meta Group, ki je del Gartnerjeve korporacije, je leta 2001 predstavil koncept velepodatkov kot arhitekturo 3 V-jev (Laney, 2001), in sicer:

- količina oziroma obseg podatkov (ang. volume),
- hitrost prevzemanja, obdelave in posredovanja podatkov (ang. velocity),
- raznolikost v smislu različne strukturiranosti, frekvence in formata podatkov (ang. variety).

A razvoj je pokazal, da ti trije V-ji niso dovolj, saj se je začel pojavljati dvom v zanesljivost zajetih podatkov. V podjetju IBM so naredili raziskavo več različnih podatkovno gnanih podjetjih. Raziskava je pokazala, da 27 % anketirancev vlaga 3,1 trilijon ameriških dolarjev na leto v zagotavljanje ustrezne kakovosti zbranih podatkov ter zanesljivost podatkovnih analiz (IBM, 2014). Glavna ugotovitev je torej bila, da je treba definicijo 3 V-jev nadgraditi z novim, četrtim V-jem (Van Rijmenam, 2014; Allen, 2016):

- verodostojnost podatkov (ang. veracity)
  Z izboljšano podatkovno infrastrukturo ter večjo kakovostjo podatkov in podatkovnih cevovodov so v zadnjem desetletju podatkovno gnana podjetja veliko pridobila z uvedbo koncepta velepodatkov. Tako paradigma velepodatkov pridobiva novo dimenzijo. Sestavni del koncepta zgoraj opisanih 4 V-jev je tako postala tudi:
- vrednost (ang. value).
  V strokovni literaturi se poleg vrednosti pojavljajo še dodatne dimenzije, kot sta na primer variabilnost (ang. variability) in vizualizacija (ang. visualization) (Van Rijmenam, 2013). Ne glede na to, koncept 5 V-jev trenutno predstavlja pot do razumevanja maksimalne in pravilne izrabe velepodatkov (Alabi, 2020).

2.1 Ekosistem velepodatkov

Vsak V zase je preprosto razumeti in o njem lahko razpredamo že z osnovnimi izkušnjami iz sveta in-

formacijske tehnologije. Marsikateri klasični informacijski sistem je možno prikazati kot popoln sistem velepodatkov. A bistvo velepodatkov se skriva v poskusih implementacije s sodobnimi informacijskimi sistemi (sistem v več oblačnih okoljih, dogodkovno orientirane arhitekture …) in z njimi povezanim doseganjem tehnoloških in konkurenčnih prednosti.

Velepodatki gredo z roko v roki z dvema sodobnima tehnikama:

- internetom stvari (ang. IoT – Internet of Things) in
- umetno inteligenco/strojnim učenjem (ang. AI – Artificial Intelligence / ML – Machine Learning).

Oboje pa obkroža še ena ključna dimenzija:

- realni čas (ang. Real Time).

Za lažje razumevanje velepodatkov in njihovo umestitev v sistem si poglejmo tri praktične primere iz realnega življenja:

1. Sprejemanje odločitev ob koncu prvega polčasa nogometne tekme

V skladu s sporazumom, sklenjenim z UEFO za obdobje 2021–2024 glede Lige prvakov ter Evropskim nogometnim pokalom v ženski kategoriji (Carp, 2022), se je morala Mednarodna korporacija za dostavo hrane Just Eat (oziroma hčerinsko podjetje La Nevera Roja) odločiti o spletni oglasni kampanji ob nastopu polčasa nogometne tekme Lige prvakov glede na poročila o prodaji med prvim polčasom tekme v omenjenem tekmovanju.

Za odločevalce je v danem trenutku pomembna predvsem hitrost pridobivanja, shranjevanja in obdelave podatkov. Direktor prodaje ni potreboval le podatkov o prodaji, temveč je za hitro reagiranje moral imeti tudi predlog optimalnega oglaševanja s ključnimi besedami preko sistema Google AdWords. Poleg tega je moral obdelanim podatkom tudi zaupati (BBVA Communications, 2021). Direktorju podatki naslednji dan ne bi pomagali; takrat bi lahko le ugotovil, kaj bi lahko storil (bolje).

2. Celovit sistem pametnih semaforjev

Pametne semaforje poznamo že dolgo, novo dimenzijo pa prinaša sodelovanje celotne infrastrukture za upravljanje semaforjev. V prometu namreč ni vsako (semaforizirano) križišče ločen otoček, temveč tvorijo del širše prometne infrastrukture. Poleg tega tudi vplivajo en na drugega. (Al Nuaimi in drugi, 2015). Za optimalno delovanje mora sistem pridobiti čimbolj celovito in objektivno sliko prometnega stanja na širšem področju (npr. na področju mesta Ljubljana).

Možni viri podatkov so senzorji na semaforjih ter že vgrajena senzorika v cestni infrastrukturi. Možen vir podatkov so lahko tudi vozila javnega prometa, kjer sta natančno znana njihov položaj in hitrost premikanja. Dodaten in pomožen vir podatkov lahko predstavljajo tudi podatki iz storitve Google Zemljevidi. Nenazadnje pa so potencialni vir podatkov kar vsa vozila v prometu oziroma vsa vozila, ki so skladna z novim standardom V2I (ang. Vehicle To Infrastructure).

Na podlagi celotne slike mora sistem pravilno upravljati semaforje, in sicer tako, da ti v najboljši meri optimalno sodelujejo skupaj in dosežejo zadani cilj – kar največjo in hkrati uravnoteženo pretočnost prometa. Dejansko to pomeni izvajanje ukrepov, kot so:

- preprečitev blokiranja križišča,
- čim hitreje sprostiti promet v naslednjem križišču ob zaznavi blokade predhodnega križišča,
- preventivna zaustavitev prometa na začetku vpadnic s ciljem preprečevanja večje zgostitve prometa na koncu vpadnice,
- podobni ukrepi.

Predvsem mora celovit sistem pametnih semaforjev posnemati delo policistov na vsakem križišču – policistov, ki so med seboj povezani (s komunikacijsko povezavo) in imajo dovolj podatkov za odločanje. Policist namreč s svojim vidnim zaznavanjem in odločanjem preprečuje primere zgoraj opisanih neželenih prometnih zamaškov.

3. Odprti podatki skupnosti

Skupnost ima s svojo javno infrastrukturo in javnimi službami (glej tudi prejšnji primer) veliko podatkov. Te podatke lahko – ob zagotovitvi vseh varnostnih standardov – preda v uporabo širši javnosti, ki lahko nato uporabi v namene raziskav in razvoja. (Al Nuaimi in drugi, 2015). Tudi v tem primeru lahko prepoznamo večino V-jev. Dodana vrednost takega pristopa se skriva v povečani intelektualni moči uporabe podatkov, skrajša pa se tudi reakcijski čas ob kriznih situacijah, ker so podatki in infrastruktura deležnikom že na voljo.

2.2 5 V-jev v praksi

Za še boljše razumevanje bomo 5 V-jev pogledali z vidika ideje pametnega semaforja.

1. Volumen

Podatke pridobivamo iz namenskih senzorjev in klasične IoT infrastrukture. Namenske senzorje bi namestili v vse semaforje; za zanesljivost delovanja morajo biti semaforji samozadostni in morajo smiselno delovati tudi, če se iz kateregakoli razloga prekine dotok informacij iz vseh posrednih virov. Že v tem koraku si lahko predstavljamo število križišč, število semaforjev v križiščih, število senzorjev v vsakem semaforju, s katerimi se pokrijejo vse točke križišča, ter predvsem praktično zvezno spremljanje stanja. V takem okolju ni dovolj zajem enega podatka na sekundo; prej govorimo o 100 zajemih na sekundo, kar je v rangu zajema žive slike.

Še bolj pa k volumnu prispevajo vse že obstoječe naprave, ki proizvajajo in imajo možnost deljenja podatkov (kar bi lahko poimenovali celotna posredna IoT infrastruktura). Razlika od prejšnjega dela je, da je ta infrastruktura postavljena iz drugih razlogov ali namenov, vendar se jo vseeno lahko uporabi kot uporabljivi vir za maksimalno in optimalno rešitev.

Po podatkih iz leta 2013 je v Ljubljani delovalo 266 semaforjev (Pandur, 2013). Če predpostavimo uporabo treh senzorjev na posamezen semafor ter 100 zajemov na sekundo, pridelamo v enem dnevu skoraj 7 milijard senzorskih podatkov.

Internet stvari dejansko prinaša veliko razliko, saj si ljudje težko predstavljamo 7 milijard transakcij dnevno. Še posebej, če gre za npr. trgovsko podjetje.

2. Hitrost

Tu dilem ni − semafor mora reagirati hipno. Bistveno je, da so vsi podatki na voljo takoj oziroma v realnem času, ki se meri v milisekundah. To je možno doseči na dogodkovno vodeni arhitekturi, kjer sprožitev neke akcije simultano omogoči pošiljanje podatka v centralni sistem.

3. Raznolikost

Deloma je raznolikost že opisana v sklopu volumna. Bi pa poudarili dve podvrsti raznolikosti. Prva in osnovna je, da lahko podatek o stanju križišča pridobimo iz različnih virov. Pri tem imamo lahko naslednje tipe podatkov:

- strukturirane podatke (npr. namenski senzorji bodo pošiljali zelo strukturirane podatke, saj so izdelani izključno za ta namen),
- polstrukturirane podatke (podatki storitve Google Zemljevidi so lahko deloma strukturirani, saj nimajo natančno identificiranega križišča, temveč se do teh podatkov pride posredno – preko geolokacije),
- nestrukturirane podatke (zajem iz videokamer že postavljenega cestnega nadzora ali v skrajnem primeru iz deljenja slike kamer v vozilih).

Taka opredelitev je daleč najbolj pogosta. Pri tej pa sicer vidim eno pomanjkljivost: lahko nas namreč kaj prehitro zadovolji, če imamo različno strukturirane podatke. Zato podajamo še drugačen pogled – lahko bi rekli kar podzvrst.

Raznolikost lahko namreč razumemo tudi z vidika dogodka, ki ga preučujemo. O njem želimo zbrati čim več raznolikih podatkov iz različnih virov. V našem primeru je preučevani dogodek lahko vozilo v križišču. Ta podatek lahko dobimo iz različnih virov, idealno medsebojno čim bolj neodvisnih.

4. Verodostojnost

Prejšnji trije elementi so glavni argument, zakaj je naslednji V prav verodostojnost. Zaradi količine, hitrosti zajemanja in raznolikosti podatkov se porodi vprašanje o verodostojnosti oziroma zaupanju v podatke. Sprva si lahko predstavljamo odločevalca pred množico grafov in tabel, ki dvomi v podatke in bo vse še dodatno preveril.

V primeru podatkov o pretočnosti križišč lahko dobimo podatke o hitrosti vozil skozi posamezno križišče. Pri tem lahko dobimo manjkajoče in nasprotujoče si podatke. Na primer, da je povprečna hitrost 20 km/h, največja pa 320 km/h. Storitev Google Zemljevidi z dodatno storitvijo Promet nam lahko prikaže, da je križišče zablokirano brez pretočnosti, senzor pa prešteje 100 vozil v minuti.

S človeškim vidom bi si seveda hitro ustvarili pravo sliko in bi lahko ustrezno prečistili podatke oziroma pravilno ukrepali. A v našem primeru to ni možno zaradi količine križišč, njihove medsebojne povezanosti ter reakcijskega časa. V našem ekstremnem primeru nimamo niti realnih možnosti napisati klasičnih algoritmov na način »če to, potem ono«, ker je kombinacij in možnosti preprosto preveč.

Edini pravi odgovor lahko ponudijo storitve umetne inteligence oziroma inteligentne uporabe podatkov. Gre za izredno kompleksne rešitve. Posledično je velikokrat ravno ta točka ključna, zakaj podjetje še vedno nima uvedene rešitve, kot npr. naš primer, pa čeprav je izredno preprosto razložljiv.

5. Vrednost

Vrednost je pogosto področje, ki se ga ob uvajanju rešitev nad velepodatki spregleda. Prejšnje štiri točke so tehnično izredno napredne, posledično pa tudi izredno zanimive za tehnični kader, kar lahko razvijalce potegne v razvoj pretiranih rešitev. Hkrati pa velja tudi obratno – ker ne znamo pravilno izračunati vrednosti, ki nam jo lahko neka rešitev prinese, razumemo vse prejšnje V-je kot strošek namesto kot naložbo.

Pri celovitih pametnih semaforjih vrednost ni zgolj odprava nepotrebnega časa v čakanju v zastojih, temveč je tu še ogljični odtis vozil, ki čakajo v križiščih. Če se tega ne da izračunati, je večja težava v prepoznavi vrednosti v odpravi slabe volje in posledično vseh negativnih posledic vseh v prometu čakajočih udeležencev.

2.3 Meje uporabe velepodatkov

Čeprav so velepodatki izjemno orodje, ki lahko pomaga pri poslovnih odločitvah podatkovno gnanih podjetij, imajo kljub temu svoje meje (Croft, 2014). Preko definicije 5-V in opredelitve ekosistema najlažje ugotovimo, kje so te meje:

- **Dajanje prednosti korelacijam**. Analitiki podatkov uporabljajo velepodatke za ugotavljanje korelacije (ko je ena spremenljivka povezana z drugo). Vendar pa vse te korelacije niso bistvene ali smiselne. Natančneje, samo zato, ker sta dve spremenljivki korelirani oziroma povezani, še ne pomeni, da med njima obstaja vzročna zveza. Na primer, med letoma 2000 in 2009 sta se podobno zmanjšala število ločitev v ameriški zvezni državi Maine in poraba margarine na prebivalca (National Center for Health Statistics, 2002). Vendar margarina in ločitev nimata veliko skupnega.
- **Napačna vprašanja**. Velike podatke je mogoče uporabiti za ugotavljanje korelacije in vpogledov z neskončno paleto vprašanj. Vendar pa je odvisno od uporabnika, da ugotovi, katera vprašanja so smiselna. Če na koncu dobite pravi odgovor na napačno vprašanje, naredite sebi, svojim strankam in svojemu podjetju zelo drago uslugo.
- **Varnost**. Kot pri mnogih tehnoloških prizadevanjih je tudi analitika velepodatkov nagnjena k zlorabam. Podatki, ki jih posredujete tretji osebi, bi lahko prišli do strank ali konkurentov.
- **Prenosljivost**. Ker se večina podatkov, ki jih potrebujete za analizo, skriva za požarnim zidom ali v zasebnem oblaku, je za učinkovito posredovanje teh podatkov skupini za analitiko potrebno tehnično znanje. Poleg tega je morda težko dosledno prenašati podatke strokovnjakom za ponovno analizo.
- **Nedoslednost pri zbiranju podatkov**. Včasih so orodja, ki jih uporabljamo za zbiranje velepodatkovnih množic, nenatančna. Na primer, Google je znan po svojih popravkih in posodobitvah, ki spreminjajo izkušnjo iskanja na različne načine; današnji rezultati iskanja bodo verjetno drugačni od jutrišnjih.

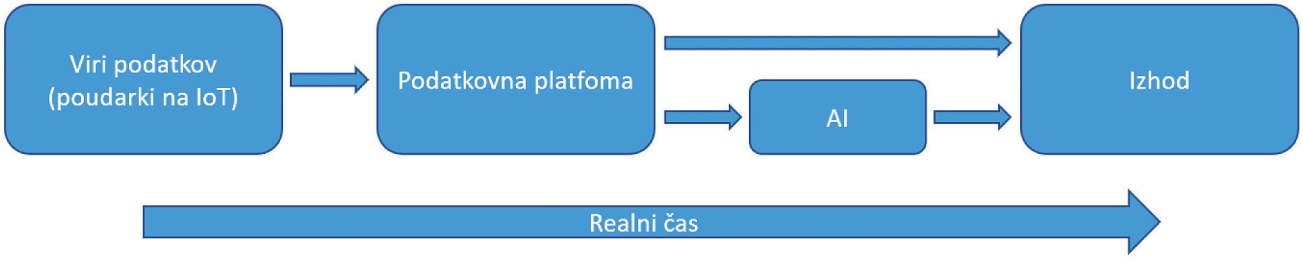
3 PODATKOVNE PLATFORME

Omenili smo že ekosistem velepodatkov: internet stvari ter umetna inteligenca (IoT, AI) v realnem času. Vendar s tem še vedno ne odgovorimo na vprašanje, kaj dejansko implementiramo in kaj je tisto, kar omogoča celotne iniciative velepodatkov. Odgovor so podatkovne platforme.

3.1 Umestitev podatkovnih platform

V spodnji shemi so podatkovne platforme umeščene na najvišjem nivoju.

Zavedati se je treba, da ta shema ni nova, saj obstaja še iz časov pred digitalizacijo poslovanja podjetij. Lahko si predstavljamo, da je podatkovna platforma ustreznica univerzitetne knjižnice, ki hrani ogromno (knjižnega) gradiva, ni pa knjižnica avtor tega gradiva



Slika 1: **Umestitev podatkovnih platform na najvišjem nivoju**.

(ni vir). Knjižnica je hkrati okolje, v katerem se lahko nekaj dela z gradivom (pregleda, izposodi, kopira, obdela …) in na podlagi tega lahko nastane neka (nova) uporabna vsebina ali rezultat (vsebinski izhod).

Podatkovna platforma je torej skupek tehnologij in orodij za zbiranje, shranjevanje in obdelavo podatkov, ki omogoča njihovo uporabo ostalim uporabnikom in orodjem. Ta opredelitev velja za vsa okolja, tudi za okolje velepodatkov. Razlika je le v sestavnih delih (gradnikih) podatkovnih platform.

3.2 Visokonivojska arhitektura podatkovnih platform za velepodatke

Arhitekture so izjemno kompleksne, saj ne obstaja ena arhitektura, ki bi ustrezala vsem potrebam. Podobno kot pri hišah je lahko neka hiša za eno družino idealna, za drugo pa povsem nefunkcionalna. Za dobro arhitekturo je treba imeti celovito znanje o vseh gradnikih podatkovnih struktur, praktične izkušnje na čim bolj reprezentativnih primerih ter sposobnost branja in razumevanja dejanske potrebe konkretnega primera.

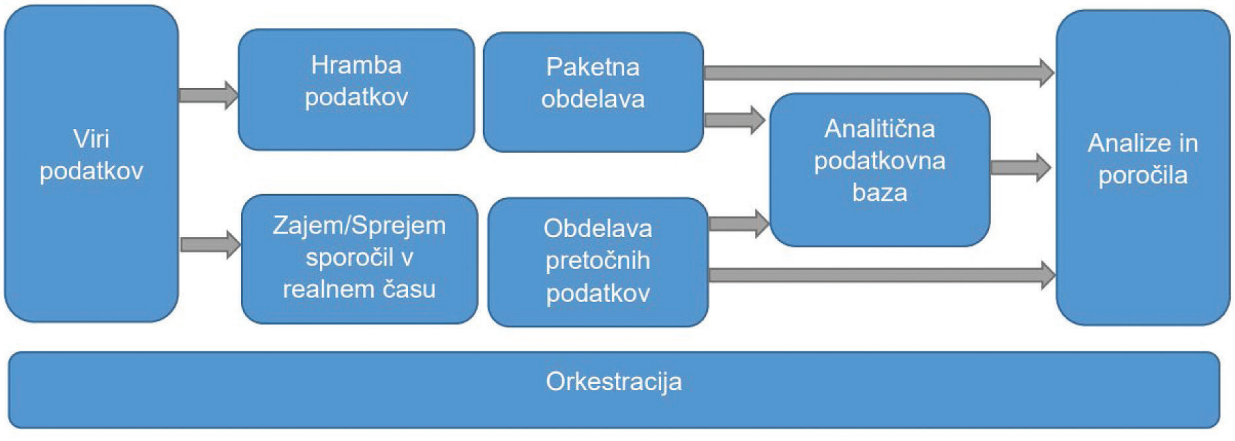
Klasična in še vedno zelo razširjena arhitektura je osredotočena okrog podatkovnega skladišča (ang. Data Warehouse) (Inmon, 2005; Kimball & Ross, 2013). Ko je postalo pomembno, da paketnemu procesiranju dodamo tudi obdelavo podatkov v realnem času, sta se razvili t. i. *Kappa* in *Lambda* arhitekturi (Marz & Warren, 2015; Kreps, 2014). Z nenehnim tehnološkim napredkom in novimi zahtevami se seveda pojavljajo vedno novi arhitekturni vzorci, v katere se v tem članku ne bomo poglabljali, pač pa bomo poskusili nekoliko posplošeno predstaviti različne elemente, ki jih podatkovna platforma lahko ima.

Najbolje je začeti pri osnovah, predvsem pa je vedno priporočljivo iti iz visokonivojske arhitekture v vedno bolj podrobne. Če se v prehodu pripetita nerazumevanje in zmedenost, se je treba vrniti le en korak nazaj in se še nekoliko seznaniti s koncepti tega nivoja arhitekture.

V tej nameri podajamo v spodnji sliki primer visokonivojske arhitekture podatkovnih platform za velepodatke. Predstavljena ideja in diagram sta Microsoftova – sicer pa so visokonivojski koncepti zelo podobni vsepovsod.

Shema prikazuje primer osnovnih visokonivojskih elementov podatkovnih platform.

- **Viri podatkov** (ang. Data Source)
  Brez virov podatkov podatkovne platforme ne morejo obstajati (kot knjižnice ne bi obstajale, če ne bi nihče ustvarjal knjižnega gradiva). Med vire podatkov sodi tudi IoT.
- **Hramba podatkov** (ang. Data Storage)
  V preteklosti so se podatki shranjevali v relacijskih podatkovnih bazah. V platformah za velepodatke ta element nadomešča osnovni datotečni sistem, saj je shranjevanje datotek tako mnogo hitrejše kot zapisovanje v relacijsko bazo. Obstajajo napredni datotečni formati, ki so prilagojeni za velepodatke (npr. Parquet, Orc ipd.). Tak datotečni sitem je primeren tudi za velike binarne datoteke (BLOB formate), kot so slike ali videoposnetki. Tako organizirana shramba podatkov se nato navzven predstavlja kot podatkovno jezero (ang. Data Lake).
- **Paketna obdelava** (ang. Batch Processing)
  Paketno obdelavo si najlažje predstavljamo kot dobro poznani proces ETL. Paketna obdelava po-



Slika 2: **Visokonivojska arhitektura podatkovne platforme. (Microsoft documentation, 2022)**

meni, da v določenih časovnih intervalih (npr. enkrat na dan) obdelamo novo spremenjene zapise ali pa celoten nabor zapisov in jih preoblikujemo v zahtevano končno obliko. Te procese načeloma izvajamo s programskimi okolji za obdelavo velikih količin podatkov (npr. Spark, Databricks ipd.)

- **Zajem/Sprejem sporočil v realnem času** (ang. Real-Time message ingestion)
  Kot smo že omenili, je pri velepodatkih izredno pomemben V tudi hitrost. Največkrat to pomeni, da internet stvari ali dogodkovno orientirana arhitektura (ang. event-driven architecture) omogoča ustvarjanje podatkov v realnem času. V tem primeru mora imeti podatkovna platforma tudi možnost zanesljivega zajema oziroma sprejema teh sporočil/podatkov v realnem času. Najbolj tipičen predstavnik tega elementa je rešitev Apache Kafka.
- **Obdelava pretočnih podatkov** (ang. Stream Processing)
  Ta element je logična posledica prejšnjega in vitalen element za vsak sistem, ki deluje v realnem času. Po zajemu podatkov v realnem času je namreč te treba tudi obdelati in (pred)pripraviti v realnem času. To je povsem nova komponenta, saj nič od obstoječih sistemov ni narejeno na ta način oziroma za ta element. V praksi se največkrat uporablja sisteme Apache Flink ali Apache Spark oziroma njegove nadgradnje/nove generacije Databricks.
- **Analitična podatkovna baza** (ang. Analytical Data Store)
  Gre za bržkone še najbolj tradicionalen element. Tu je mišljeno najbolj osnovno podatkovno skladišče oziroma OLAP-nivo za potrebe klasične podatkovne analitike. Ne glede na ves blišč velepodatkov, klasična podatkovna analitika ne izginja. Še vedno velik del praktične rabe predstavljata klasično analiziranje in predstavitev (vizualizacija) podatkov.
- **Analize in poročila** (ang. Analysis and Reporting)
  Tudi ta element je zelo klasičen. Najlažje ga razumemo kot podatkovno analitiko oziroma obveščanje (ang. Business Intelligence). Zaradi razvoja sodobnih orodij za samopostrežno analitiko pa ta element vendarle ni tako tradicionalen kot prejšnji.
- **Orkestracija** (ang. Orchestration)
  Z dodajanjem novih elementov že v visokonivojsko arhitekturo (sploh pa s številnimi storitvami v dejanski izvedbi) se poveča tudi zahteva po orkestraciji oziroma učinkovitem sodelovanju vseh storitev (ang. Services) med seboj. Zato je ta element izpostavljen in postavljen že kot samostojen v visokonivojski arhitekturi.

V ta del lahko uvrstimo tudi element upravljanja in ravnanja s podatki (ang. Governance), saj se podatki pretakajo čez mnogo storitev, nivojev ter sistemov. S tem se potenčno poveča možnost nastanka šumov v podatkih. V tem primeru pride do izraza dober sistem upravljanja in ravnanja s podatki.

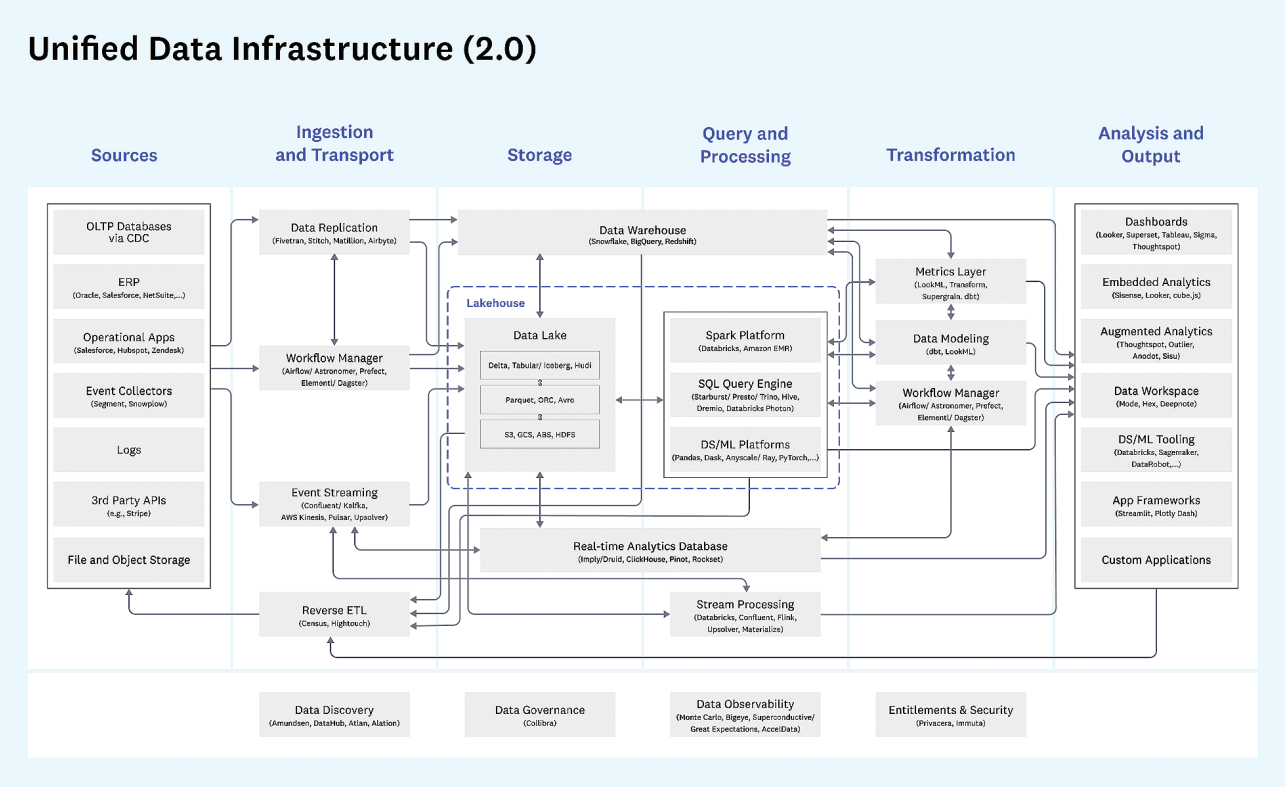
Izkušnje iz klasičnih in tradicionalnih podatkovnih struktur kažejo, da se v bistvu dodajata le dva nova elementa: zajem/sprejem sporočil v realnem času in obdelava pretočnih sporočil. To je seveda povsem razumljivo, saj smo v podpoglavju ekosistema posebej poudarili dimenzijo realnega časa.

3.3 Podrobna arhitektura podatkovnih platform za velepodatke

V visokonivojski arhitekturi pravzaprav nismo naredili velikega preskoka, kar je seveda prav, saj smo izpostavili, da je treba korakati postopoma. Tako stališče pa ne velja za podrobno arhitekturo. Tu se ne dodajo zgolj posamezne storitve, temveč se zelo spremenijo tudi storitve za izvedbo tradicionalnih elementov; podobno kot so se ob pojavu avtomobilov na prelomu 20. stoletja spremenila celotna infrastruktura in pravila. Avtomobili se niso le dodali kočijam, ampak so se morale tudi kočije prilagoditi novim časom.

Pri podrobnih arhitekturah velja enako kot pri visokonivojskih – ni le ene različice resnice. Vendarle pa večina snovalcev stremi k uveljavljenim konceptom in terminologiji, ki pa se na podrobnem nivoju drastično spreminja glede na tradicionalne sisteme.

Za osnovo razlage koncepta sodobnih podatkovnih platform tokrat izhajamo iz ideje, ki jo je podprl članek na portalu Andreesen Horowitz in je prikazana na spodnji sliki:



Slika 3: **Primer podrobne arhitekture podatkovne platforma za velepodatke. (Bornstein, Li, & Casado, 2020)**

Predstavljena arhitektura je neodvisna od velikih proizvajalcev rešitev in ponudnikov storitev (AWS, MS Azure, Google, Oracle …), pri njeni zasnovi pa so sodelovali številni inženirji posameznih predstavnikov rešitev.

Pri tej shemi je zelo ilustrativen način prikaza arhitekture, razdeljene na sklope (stolpce), ki jih lahko uskladimo s prej predstavljenimi visokonivojskimi elementi. V sklopih so našteti ključni elementi ter vodilna orodja oziroma ogrodja, s katerimi je možno izvesti te elemente. Za podroben opis vseh bi potrebovali nov (ali daljši) prispevek, vseeno pa je smiselno izpostaviti tiste, ki prinašajo drastične spremembe:

- **Zajem sprememb v relacijskih bazah** (ang. OLTP Databases via CDC)
  Ta blok pretvarja klasične relacijske podatkovne baze v dogodkovno orientirano arhitekturo. Izraz CDC namreč pomeni zajem sprememb podatkov (ang. *Change Data Capture*), kar pomeni, da lahko vsaka sprememba v relacijski bazi sproži dogodek, ki se ga posreduje na sprejem sporočil v realnem času. S tem blokom lahko vsako obstoječo tradicionalno programsko rešitev precej enostavno pretvorimo v sodobno dogodkovno usmerjeno rešitev. Dobro izpostavljeno orodje za to ponuja Oracle z rešitvijo Oracle Golden Gate, priljubljena odprtokodna alternativa pa je Debezium.
- **Upravljalec podatkovnih operacij** (ang. Workflow Manager)
  Ta blok je neposredna rešitev orkestracije iz visokonivojske arhitekture. Izpostavili smo že, da je orkestracija pomemben element. Obstajajo vgrajena ali neodvisna orodja, katerih glavna naloga je orkestracija vsega dogajanja. Zelo prodorno orodje v tem sklopu je Apache Airflow.
- **Modeliranje podatkov in upravljanje z meta podatki** (ang. Data Discovery, Data Governance)
  Ob razumevanju petih V-jev vemo, da imamo veliko podatkov, ki so hkrati tudi zelo raznoliki. Razumevanje podatkov in upravljanje z njimi je za velike sisteme precejšnje breme, zato sta tu enakovredno poudarjena bloka podatkovnega modeliranja in upravljanja z meta podatki.
- **Podatkovno jezero** (ang. Data Lake)
  Podatkovno jezero je verjetno najtežje razumljiv element te podatkovne strukture. Težava je v idej-

ni opredelitvi, ki se posplošeno nanaša le na kopijo surovih podatkov iz operativnih sistemov. Zato je velikokrat poenostavljeno enačeno kot področje priprave podatkov (ang. staging), kar pravzaprav tudi je (lahko). Vendarle pa je na nivoju podatkovnih platform za velepodatke ta nivo vseeno mišljen za učinkovito shranjevanje vseh podatkov, tudi pol- in nestrukturiranih (spomnimo se na element raznolikosti pri opredelitvi 5-V). Zavedati se moramo, da pri velepodatkih shranjujemo tudi slike, videoposnetke, pomanjkljive podatke, podatke s spreminjajočo se strukturo ali povsem nepoznane podatke in strukture. V tem kontekstu pa se namen podatkovnega jezera loči od področja priprave podatkov (saj se velepodatki ločijo od običajnih podatkov).

- **Podatkovno skladišče** (ang. Data Warehouse)
  Podatkovno skladišče je verjetno daleč najbolj prepoznaven izraz v podani arhitekturi. Kar nas lahko preseneti, je premik od tradicionalnih konceptov do platform velepodatkov. V tradicionalnih sistemih je bila podatkovna platforma enaka podatkovnemu skladišču, sedaj pa je podatkovno skladišče le eden izmed elementov podatkovne strukture.
  To je pa pravzaprav logično, saj že iz visokonivojske arhitekture sledi, da se ne moremo izogniti analitičnim podatkovnih bazam, a te niso več glavni igralec zaradi dimenzije obdelave v realnem času.
- **Kolišče** (ang. Lakehouse)
  Arhitektura kolišča postaja vse bolj prepoznavna, zlasti ko ga je začel podpirati nabor ponudnikov (vključno z AWS, Databricks, Google Cloud, Starbrust in Dremio) ter pionirji podatkovnih skladišč. Temeljna vrednosti kolišča je združiti prednosti podatkovnega jezera in podatkovnega skladišča. Tako dobimo cenovno dostopno shranjevanje podatkov v odprtih formatih, hkrati pa imamo še vedno na voljo napredne funkcionalnosti, npr. ACID transakcije, indeksiranje, verzioniranje podatkov itd. (Armbrust, 2021).
- **Spontane poizvedbe** (ang. Ad Hoc Query Engine)
  Ta element je zanimiv, saj povezuje tradicionalni pristop z ekstremi velepodatkov. Če smo v tradicionalnih sistemih imeli relacijsko bazo, ki je bila na nek način črna škatla, kjer nikjer nismo imeli pravega vpliva na to, kako se bodo podatki shranjevali in kako bomo zares dostopali do njih, imamo zdaj ločene bloke za isto uporabniško izkušnjo. Uporabniki namreč še vedno želijo brskati po podatkih s programskim jezikom SQL. V sodobnih sistemih imamo blok podatkovnega jezera, ki je pravzaprav datotečni sistem s shranjenimi datotekami. Ta blok nam omogoča, da lahko do vsebin teh datotek dostopamo preko jezika SQL. Tako navzven ne spreminjamo uporabniške izkušnje, navznoter pa imamo možnost popolne prilagodljivosti za uporabo velepodatkov.

Katere od vseh elementov, ki jih imamo na izbiro, v arhitekturi svoje podatkovne platforme dejansko uporabimo, je odvisno od naših potreb, pričakovanj in načrtov. Vsekakor lahko začnemo le s klasičnimi elementi, ki jih morda že imamo, nato pa jih ob pravem trenutku nadgradimo z dodatnimi, naprednejšimi.

4 ZAKLJUČEK

Videli smo, da velepodatke težko natančno definiramo, lahko pa jih opredelimo s 5 V-ji (obseg, hitrost, raznolikost, verodostojnost in vrednost). Z razumevanjem vseh teh dimenzij se nam odpre pravi pogled na priložnosti velepodatkov, ki jih mora podatkovno gnano podjetje razumeti, da bi lahko v pravem trenutku še dodatno dvignilo vrednost svojih podatkov in tako povečalo svojo podatkovno zrelost.

Ker je treba tudi tehnično podpreti vse operacije, ki morajo obdelovati vseh 5 V-jev, se v ta namen pojavlja gradnik podatkovne platforme za velepodatke. Ta je skupek tehnologij in orodij za omogočanje klasičnih ter tradicionalnih procesiranj podatkov, mora pa se tudi uspešno spopadati z najbolj ekstremnimi izzivi velepodatkov. Te platforme pa le niso tako zapleten, kot je morda videti na prvi pogled, saj so le naravna evolucija dosedanje poti. Le pogledati jih moramo postopoma iz grobe visokonivojske arhitekture do podrobne logične.

Dejstvo je, da so velepodatki realnost. Predvsem zaradi interneta stvari je podatkov več kot kadarkoli, podatki so mnogo bolj raznoliki, vse več pa je tudi nestrukturiranih podatkov. Ti se ustvarjajo hitreje kot kadarkoli. Pri tem se je treba zavedati, da zgolj proizvedeni in shranjeni podatki predstavljajo predvsem strošek, zato je nujno, da podatke uporabimo tako, da nam prinesejo čim večjo vrednost, saj se le tako smotrno sklene celotna veriga življenjske dobe podatka.

Velepodatki prinašajo popolnoma nove ekstremne dimenzije možnosti in priložnosti obdelave, a

vendar je z dobrim razumevanjem preskok lahko narediti tudi postopoma in z majhnimi koraki. Iz lastnih izkušenj podamo nekaj smernic:

izkoristimo priložnost razumeti ozadja ter jih smiselno vpeljati v našo trenutno realnost in pričakovan prihodnji razvoj, saj bomo tako še uspešnejše transformirali naše poslovanje v podatkovno gnano organizacijo in družbo;

povečajmo zavedanje, da sta podatkovna kultura in zrelost ena izmed načinov, da se razume namen ter korist, ki ga imamo z uvedbo moderne »big data« platforme;

načrtujmo zaposlitev ustreznih kadrov, ki bodo zmožni zgraditi platforme za ravnanje z velepodatki, kot so podatkovni inženirji, znanstveniki in analitiki.

Vsekakor ima uvedba velepodatkov v podjetje pozitivne učinke − če se seveda procesa transformacije lotimo pravilno in po korakih, prilagojenih trenutnemu stanju v podjetju, zmožnostim in potrebam.

VIRI IN LITERATURA

[1] Al Nuaimi, E., Al Neyadi, H., Mohamed, N., & Al-Jaroodi, J. (2015). Applications of big data to smart cities. Journal of Internet Services and Applications, 6, 25. doi:10.1186/s13174-015-0041-520

[2] Alabi, M. O. (2020). Big Data, 3D Printing Technology, and Industry of the Future. (I. R. Association, Ured.) Additive Manufacturing: Breakthroughs in Research and Practice, 503-520. doi:10.4018/978-1-5225-9624-0.ch021

[3] Allen, M. (2016). The Challenge of Big Data–It's more than just big files. Pridobljeno 30. oktober 2017 iz Pro2Col: https://pro2col.com/challenge-big-data-more-than-just-big-files

[4] Armbrust, M. et al (2021). Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. CIDR '21, Jan. 2021.

[5] Awanish. (19. september 2021). Big Data Tutorial: All You Need To Know About Big Data! Pridobljeno iz Big Data Tutorial: All You Need To Know About Big Data!: https://www.edureka.co/blog/big-data-tutorial

[6] BBVA Communications. (19. september 2021). The five V's of big data. Pridobljeno iz https://www.bbva.com/en/five-vs-big-data/

[7] Bornstein, M., Li, J., & Casado, M. (2020). Emerging Architectures for Modern Data Infrastructure. Pridobljeno iz Future: https://future.com/emerging-architectures-modern-data-infrastructure/

[8] Carp, S. (2022). Uefa's Just Eat sponsorship covers Champions League and Women's Euro. Pridobljeno 9. julij 2022 iz SportsPro SmartSeries: https://smartseries.sportspromedia.com/news/uefa-just-eat-sponsorship-champions-league-womens-euro

[9] Croft, C. (2014). The Limits of Big Data. SAIS Review of International Affairs, 34, 117–120. doi:10.1353/sais.2014.0005

[10] Dehghani, Z. (2019). How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh. Pridobljeno 3. avgust 2022 iz https://martinfowler.com/articles/data-monolith-to-mesh.html

[11] Dice. (2020). Dice Tech Job Report: The Fastest Growing Hubs, Roles and Skills. Pridobljeno iz ISSUE #1: Q1 2020: https://techhub.dice.com/Dice-2020-Tech-Job-Report.html

[12] Edjlali, R., & Beyer, M. (2012). Understanding the Logical Data Warehouse: The Emerging Practice. Gartner Tech. Rep. G00234996.

[13] Hashem, I. A., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Ullah Khan, S. (2015). The rise of »big data« on cloud computing: Review and open research issues. Information Systems, 47, 98–115. doi:10.1016/j.is.2014.07.006

[14] IBM. (2014). IBM Big Data & Analytics Hub: The Four V's of Big Data. Pridobljeno 26. oktober 2017 iz http://www.ibmbigdatahub.com/infographic/four-vs-big-data

[15] Inmon, W. H. (2005). Building the Data Warehouse 4th Edition. Wiley.

[16] Kimball, R., & Ross, M. (2013). The Data Warehouse Toolkit: : The Definitive Guide to Dimensional Modeling 3rd Edition. Wiley.

[17] Kreps, J. (2014). I Heart Logs: Event Data, Stream Processing and Data Integration. O'Reilly Media.

[18] Laney, D. (2001). 3D Data Management: Controlling Data Volume, Velocity, and Variety. META Delta. Application Delivery Strategies, 1–4.

[19] Linstedt, D., & Olschimke, M. (2015). Building a Scalable Data Warehouse with Data Vault 2.0. Morgan Kaufmann.

[20] Login5 Foundation. (22. June 2022). LoginEKO. Pridobljeno iz https://www.logineko.com/

[21] M. Cox, & D. Ellsworth. (1997). Managing Big Data for Scientific Visualization.

[22] Mansoor, I. (30. June 2022). Netflix Revenue and Usage Statistics (2022). Pridobljeno 8. julij 2022 iz Business of Apps: https://www.businessofapps.com/data/netflix-statistics/

[23] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. (2011). Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute.

[24] Marz, N., & Warren, J. (2015). Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning.

[25] Microsoft documentation. (2022). Big data architecture style. Pridobljeno 9. julij 2022 iz docs.microsoft.com: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/big-data

[26] National Center for Health Statistics. (7. februar 2002). National Vital Statistics System. Pridobljeno iz cdc.gov: https://www.cdc.gov/nchs/nvss/marriage-divorce.htm?CDC_AA_refVal=https%3A%2F%2Fwww.cdc.gov%2Fnchs%2Fmardiv.htm

[27] Oracle. (2022). Oracle Cloud Infrastructure (OCI) GoldenGate. Pridobljeno 9. julij 2022 iz https://www.oracle.com/integration/goldengate/

[28] Pandur, S. (2013). Kjer imajo nadzor nad vsemi rdečimi lučmi. Delo.

[29] Priebe, T., Neumaier, S., & Markus, S. (2021). Finding Your Way Through the Jungle of Big Data Architectures. 2021 IEEE International Conference on Big Data (Big Data) (str. 5994–5996). Orlando, FL, USA: IEEE. doi:10.1109/BigData52589.2021.9671862

[30] Van Rijmenam, M. (2013). Why The 3V's Are Not Sufficient To Describe Big Data. Pridobljeno 8. July 2022 iz Datafloq: https://datafloq.com/read/3vs-sufficient-describe-big-data/

[31] Van Rijmenam, M. (2014). Think Bigger: Developing a Successful Big Data Strategy for Your Business. American Management Association.

[32] Wikipedia. (19. september 2021). Pridobljeno iz Big Data: https://en.wikipedia.org/wiki/Big_data
[33] Zaidi, E., Thoo, E., De Simoni, G., & Beyer, M. (2019). »Data Fabrics Add Augmented Intelligence to Modernize Your Data Integration. Gartner, Tech. Rep. G00450706

▪

Jure Jeraj je diplomiral na Fakulteti za računalništvo in informatiko ter opravil specializacijo na Fakulteti za organizacijske vede s področja Organizacije in managementa informacijskih sistemov. Trenutno je zaposlena pri podjetju Result, d.o.o. kot vodja oddelka za podatke in umetno inteligenco. Predvsem v zadnjih 10 letih se osredotoča na informacijske rešitve v povezavi s podatki. Opravljal je naloge arhitekta podatkovnih skladišč in sistemov za poslovno analitiko, naknadno se je specializiral za arhitekturo modernih podatkovnih platform in uporabe velepodatkov. Pridobljeno znanje deli na raznih strokovnih konferencah in ostalih dogodkih. Je tudi programski direktor Foruma podatkovne analitike.

▪

Urška Nered je diplomirala iz fizike na Fakulteti za matematiko in fiziko Univerze v Ljubljani. Med študijem je pridobila izkušnje kot podatkovni analitik pri podjetju Result, d.o.o, kjer se je kasneje tudi zaposlila kot podatkovni inženir. Urška se v svojem delu osredotoča na podatkovne arhitekture ter sodeluje pri številnih domačih in tujih projektih, kjer je prispevala k uspehu in kvaliteti izvedbe.

▪

Stevanče Nikoloski je doktor znanosti na področju informacijske in komunikacijske tehnologije. Pet let je delal na Inštitutu Jožef Stefan v Ljubljani, kot doktorski študent in raziskovalec na oddelku Tehnologije znanja, in sicer na področju umetne inteligence. Svoje raziskovalno delo je objavil v zelo prestižnih revijah in ga predstavil na konferencah. Trenutno je zaposlen kot vodilni podatkovni znanstvenik v podjetju Result, d.o.o. v Ljubljani in je odgovoren za grajenje Data Science kompetenc. Poleg tega uči kot visokošolski učitelj predmeta »Programski inženiring« in »Digitalizacija poslovnih procesov« na Fakulteti za ekonomijo in informatiko na Univerzi v Novem mestu.

Samoupravljana digitalna identiteta na verigi blokov Cardano

Nikolay Vasilev, Dejan Lavbič
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
nv7834@student.uni-lj.si, dejan.lavbic@fri.uni-lj.si

Izvleček

Problem današnjega interneta je, da je na voljo veliko informacij, ki jih nekdo nadzoruje (npr. oseba, organizacija), teh ljudi pa sploh ne poznamo. Samoupravljana identiteta je okvir zaupanja, pri katerem imajo entitete, kot so: ljudje, organizacije in abstraktne entitete, svoje popolnoma avtonomne identitete. Ta vrsta decentraliziranih digitalnih identitet s pomočjo decentraliziranih identifikatorjev in preverljivih poverilnic entitetam omogoča varno komunikacijo, nadzor nad lastnimi podatki ter izbiro, kaj in s kom bodo delili. Izmenjava podatkov se zgodi z uporabo verige blokov in tehnologije razpršene evidence, ki nam omogoča zaščitene in varne transakcije. S pomočjo te nove tehnologije in verige blokov Cardano smo razvili odprtokodno rešitev, ki bo študentom v pomoč pri izobraževalnem procesu. Ta rešitev pomaga predstaviti funkcionalnosti, ki nam jih decentralizirane digitalne identitete ponujajo pri izobraževanju. Z uporabo SWOT-analize smo primerjali sistem z že obstoječimi rešitvami, pri čemer pokažemo, da uporaba samoupravljene digitalne identitete naredi našo rešitev varnejšo, zmogljivejšo, cenejšo in uporabnejšo.

Ključne besede: decentralizirana digitalna identiteta, decentraliziran identifikator, samoupravljana identiteta, preverljiva poverilnica, preverljiva predstavitev, veriga blokov Cardano

SELF-SOVEREIGN DIGITAL IDENTITY ON THE CARDANO BLOCKCHAIN

Abstract

Nowadays, there is a problem on the Internet with the overabundance of information controlled by a single entity (e.g. person, organization), which we don't know at all. Self-sovereign identity is a framework of trust where entities such as people, organizations, and abstract entities, have their own fully autonomous identities. With the help of decentralized identifiers and verifiable credentials, this type of decentralized digital identity enables entities to securely communicate, control their data, and choose what and with whom to share. Data exchange occurs using a blockchain and a distributed ledger technology, which allows for protected and safe transactions. With the help of this new technology and the Cardano blockchain, we have developed an open-source solution to help students in their educational process.
This solution allows us to present the decentralized digital identities' functionalities in education. Using SWOT analysis, we compared the system to existing solutions, proving that self-sovereign digital identity makes our solution more secure, robust, cheaper, and practical.

Keywords: Decentralized digital identity, decentralized identifier, self-sovereign identity, verifiable credential, verifiable presentation, Cardano blockchain

1 UVOD

Zaradi naraščajoče potrebe po digitalnih identitetah (*angl.* Digital identity) [4, 9] se je zanimanje za upravljanje z identitetami v zadnjih letih povečalo. Digitalna identiteta je velik del življenj ljudi na spletu in se večinoma uporablja kot digitalni dokaz, da je lastnik tisti, za katerega se predstavlja, ko komunicira s storitvami.

Upravljanje identitet na spletu lahko najbolje predstavimo s tremi modeli, ki so se z leti razvijali in dopolnjevali [10, 12]. Prvi model identitete, ki je najbolj znan in že dolgo časa uporabljan s skoraj vsemi identifikatorji in poverilnicami, je centraliziran model identitete (*angl.* Centralized identity model) [6]. Tovrstne digitalne identitete se vzpostavijo z ustvarjanjem računa na spletnem mestu, storitvi ali aplika-

ciji. To pomeni, da entitetam neki drug centraliziran organ posoja poverilnice, ki omogočajo omejen nadzor in dovoljenja, vendar na koncu te poverilnice še vedno pripadajo organu, ki jih je izdal. Entitete ne morajo obstajati v tem centraliziranem sistemu, če nimajo ustvarjenega uporabniškega računa. Zato je možnost dostopa do storitev preklicana, ko entiteta izbriše vse svoje račune, povezane s tem centraliziranim ponudnikom. Druge težave tega modela so naslednje: vsaka spletna stran izvaja svoje varnostne politike in politike o zasebnosti, ki se med seboj razlikujejo; podatki o identiteti niso prenosljivi ali ponovno uporabni; upravljanje vseh različnih računov (uporabniških imen in gesel) je težko in lahko postane breme za entiteto; centralizirane baze podatkov lahko povzročijo resne kršitve varnosti osebnih podatkov. Drugi model identitete, imenovan model federativne identitete (*angl.* Federated identity model) [22], odpravi nekatere težave centralizirane identitete. Izboljšava je, da je med entiteto in centraliziranim ponudnikom dodan ponudnik identitete (*angl.* Identity provider, IDP). Tako ima lahko entiteta en uporabniški račun pri IDP, kar bo omogočilo skupno uporabo nekaterih osnovnih podatkov o identiteti v katerem koli spletnem mestu, storitvi in aplikaciji, ki uporablja ta IDP. Obstaja veliko znanih in uspešnih protokolov, ki uporabljajo ta identitetni model (SAML, OAuth, OpenID Connect), vendar ima tudi ta model nekaj resnih težav, kot so: še vedno obstaja težava, da je treba imeti več kot en račun (identiteta), saj ne obstaja le en IDP, ki bi deloval z vsemi spletnimi mesti, storitvami in aplikacijami; uporabniki morajo zaupati upravljanje in nadzor svojih podatkov nekomu (IDP), ki ga sploh ne poznajo; računi niso bolje prenosljivi kot računi centralizirane identitete; tudi ponudniki identitet uporabljajo centralizirano bazo podatkov, kar pomeni, da podatki niso zaščiteni proti kibernetskim napadom, zato IDP tudi onemogoča uporabnikom, da bi lahko varno delili nekatere svoje najpomembnejše osebne podatke. Najnovejši model – model decentralizirane identitete (*angl.* Decentralized identity model) – ne temelji več na uporabniških računih, vendar deluje kot identiteta v resničnem svetu. Ta model temelji na neposrednem odnosu med dvema entitetama (vrstnikoma), zato se komunikacija imenuje »Vsak z vsakim« (*angl.* Peer-to-peer, P2P). Tako decentralizirane identitete dajejo entiteti popolno avtonomijo nad lastno identiteto, ki je lahko podprta z uporabo verige blokov (*angl.* Blockchain) [2] in tehnologije razpršene evidence (*angl.* Hyperledger technology) [1], tako da si oba vrstnika delita povezavo, ki je zavarovana z uporabo decentralizirane infrastrukture javnih ključev (*angl.* Decentralized Public Key Infrastructure, DPKI) [7]; javni ključi se izmenjajo za omogočanje zasebnih in varnih povezav med dvema vrstnikoma; nekateri izmed teh javnih ključev so shranjeni v javnih verigah blokov za preverjanje podpisov na poverilnicah.

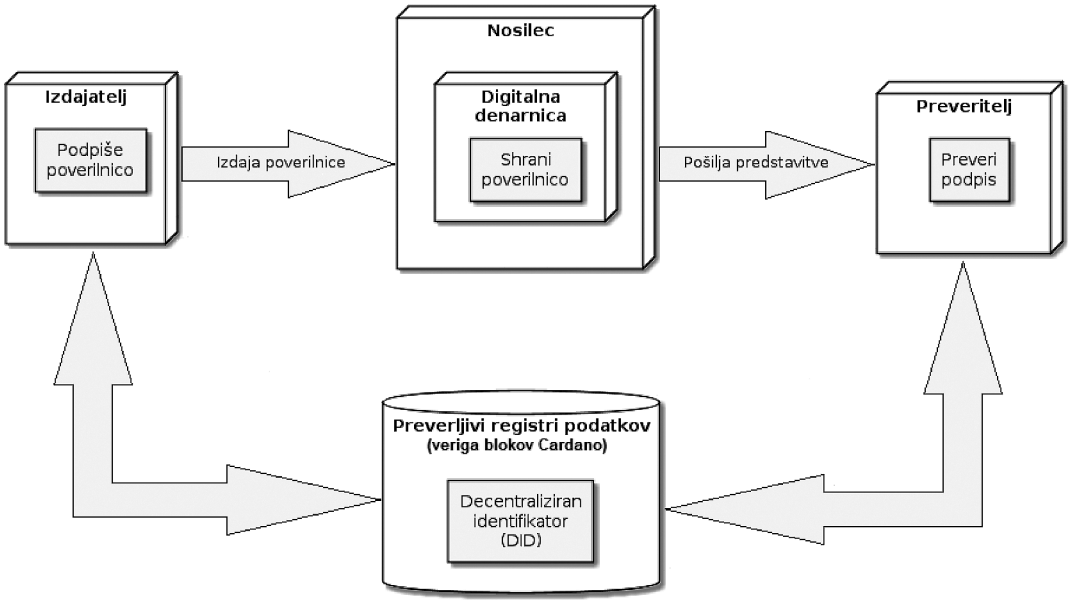
Vrsta decentralizirane identitete je samoupravljana identiteta (*ang.* Self-sovereign identity, SSI) [12, 17], ki vpeljuje tudi decentralizirane identifikatorje (*angl.* Decentralized identifier, DID) [19] in preverljive poverilnice (*angl.* Verifiable credential, VC) [20], ki imajo naslednjo vlogo: decentralizirani identifikatorji omogočajo varno komunikacijo med entitetami; entitete si izmenjujejo preverljive poverilnice prek že ustvarjenega varnega komunikacijskega kanala na tak način, da lahko entitete nadzorujejo svoje osebne podatke in izbirajo, kaj in s kom bodo delile. Da bi predstavili, kako samoupravljane identitete obvladujemo v praksi in predvsem pri izobraževanju, bomo razvili primer uporabe za podporo študentom med študijskim procesom, ki bo uporabljal SSI na verigi blokov Cardano. Študenti morajo zelo pogosto znati: upravljati s svojimi osebnimi informacijami, dokazovati svojo identiteto ter varno in hitro opravljati svoje obveznosti.

Čeprav je že veliko razvitih sistemov, ki študentom pomagajo pri tem, se bomo osredinili na manjkajoče dele teh rešitev in jih z implementacijo SSI nadgradili. Analiza sistema in primerjava z drugimi podobnimi rešitvami s pomočjo SWOT-analize nam bo pomagala ugotoviti, kaj nam implementacija SSI omogoča, kaj so njene slabosti in kaj še lahko izboljšamo. Tako lahko ocenimo uporabnost SSI kot tehnološke rešitve na izbrani problemski domeni.

V razdelku 2 tega prispevka bomo opisali, kaj predstavljajo SSI na splošno, katere so njene najpomembnejše komponente in kakšna je njihova arhitektura. Razdelek 3 se bo začel z opisom že obstoječih sistemov in s predstavitvijo, kako nam bodo pomagali pri razvoju naše rešitve; opisali bomo celotno kompleksnost naše rešitve. V razdelku 4 sledita predstavitev glavnih funkcionalnosti odprtokodne rešitve in ovrednotenje naše rešitve s pomočjo SWOT-analize. V razdelku 5 bomo povzeli rezultate analize, predlagali možne nadgradnje tehnologije in predstavili razpon njene potencialne uporabe v prihodnosti.

2 SAMOUPRAVLJANA IDENTITETA

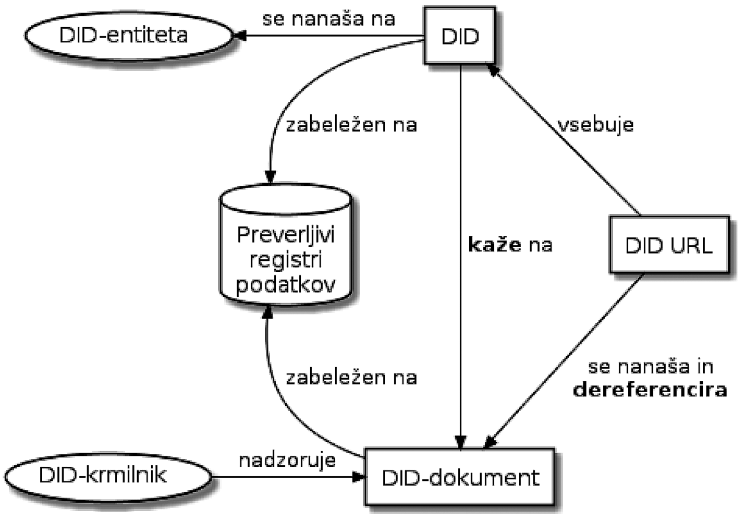
Samoupravljana identiteta je okvir zaupanja, pri katerem imajo entitete, kot so: ljudje, organizacije in abstraktne entitete, svoje identitete. Ta vrsta decentraliziranih digitalnih identitet s pomočjo decentraliziranih identifikatorjev in preverljivih poverilnic entitetam omogoča varno komunikacijo, nadzor nad lastnimi podatki ter izbiro, kaj in s kom bodo delili. DID uporabnikom omogoča vzpostavitev varnega komunikacijskega kanala z uporabo decentralizirane infrastrukture javnih ključev. Ko je kanal vzpostavljen, lahko entitete izmenjajo preverljive poverilnice, ki lastnikom omogočajo nadzor nad svojo identiteto oziroma nad osebnimi podatki. VC dejansko predstavljajo globalno enolične zahtevke izdajatelja o nekom ali nečem drugem. Da bi lahko entitete dobile nadzor nad lastnimi podatki oziroma možnost izbrati, katere podatke bodo delile (podatke iz ene ali več poverilnic), je W3C (*angl.* World Wide Web Consortium) [21] predlagal preverljive predstavitve (*angl.* Verifiable presentation, VP) [20]. VP predstavlja dokaz, da ima entiteta preverljive poverilnice (podatke) za določene trditve, oziroma omogoča entitetam izbrati, katere svoje poverilnice (zahtevke) in s kom jih bodo delile. Kot je prikazano na sliki 1, VC in VP omogočata dokazovanje in preverjanje podatkov ter sta glavni del SSI-arhitekture oziroma sta glavni del komunikacije med izdajateljem (*angl.* Issuer), nosilcem (*angl.* Holder) in preveriteljem (*angl.* Verifier). V naši rešitvi so preverljivi registri podatkov verige blokov Cardano, ki uporabnikom omogočajo zaščitene in varne transakcije; izdajatelji in preveritelji so profesorji in asistenti, čeprav je lahko vsaka entiteta, ki ima svoj DID, objavljena na verigi blokov; nosilci pa so študenti, ki potrebujejo le svojo digitalno denarnico za zbiranje in deljenje svojih poverilnic.



Slika 1: **Glavne vloge pri SSI-arhitekturi**

V rešitvi bomo implementirali DID skladno s standardom W3C [19] na verigi blokov Cardano. Kot opisuje standard, je nekaj glavnih konceptov, s katerimi se srečamo, ko govorimo o arhitekturi DID (slika 2):

- **DID:** globalno enolični identifikator oziroma enotni identifikator vira (*angl.* Uniform Resource Identifier, URI), ki kaže na DID-dokument in je sestavljen iz treh delov (primer 1). DID je objavljen na verigi blokov, da lahko uporabnikom omogoča dostop do DID-dokumenta pri preverjanju veljavnosti poverilnic. Tako je shranjevanje podatkov skladno s splošno uredbo o varstvu podatkov (*angl.* General Data Protection Regulation, GDPR), saj entiteti ni treba objaviti osebnih podatkov na verigi blokov. Prvi del je identifikator sheme URI (*angl.* URI scheme identifier) »did«. V sredini se nahaja DID-metoda (*angl.* DID me-



Slika 2: **DID-arhitektura skladno s standardom W3C [19]**

thod), ki določa, kako se določena vrsta DID in z njim povezan DID-dokument ustvari, posodobi ali deaktivira. Na koncu je identifikator, specifičen za DID-metodo (*angl.* DID method specific identifier), ki ga določa DID-metoda.
did : *prism* : 6*e*368*b*3*a*7*c*42276*f* 9*ed...* (1)

- **DID URL:** enotni iskalnik virov (*angl.* Uniform Resource Locator, URL), ki razširi sintakso osnovnega DID, tako da vključi druge standardne komponente URI-ja, kot so: pot, poizvedba in fragment, da bi poiskali določen vir (primer 2).
did : *prism* : 6*e*368*b*3*a*7*c*42276*f* 9*ed.../poverilnica# diploma* (2)
- **DID-dokument (*angl.* DID document, DDO):** sestavljen iz nabora podatkov, ki opisuje DID-entitete. Do informacije pa lahko dostopajo tudi druge entitete prek DID, ki kaže nanj.
- **DID-krmilnik (*angl.* DID controller):** entiteta (oseba, organizacija ali avtonomna programska oprema), ki ima možnost, kot je opredeljeno z DID-metodo, spreminjati DID-dokument.
- **DID-entiteta (*angl.* DID subject):** glavna entiteta (oseba, skupina, organizacija, stvar ali koncept), ki jo identificira DID ter jo opiše DDO. DID-entiteta lahko pooblasti DID-krmilnik za spreminjanje DDO in je lahko hkrati tudi DID-krmilnik.
- **Preverljivi registri podatkov (*angl.* Verifiable Data Registries, VDR):** osnovni sistem ali omrežje, v katerem so DDO, DID in VC zabeleženi.

VC in VP bosta implementirana skladno s standardom W3C [20] in na verigi blokov Cardano. Kot lahko vidimo na sliki 3, bodo preverljive poverilnice vsebovale naslednje komponente: metapodatke poverilnice, kot so: vrsta poverilnice, datum izdaje in izdajatelj; eno ali več dokazil, ki je dejansko en ali več podpisov izdajatelja (vrsta, datum izdelave, nonce[1] in vrednost podpisa); enega ali več zahtevkov, ki so lahko v odvisnosti od implementacije enolični za vsak sistem, saj predstavljajo enega ali več pa-

Slika 3: **VC- in VP-struktura skladno s standardom W3C [20]**

[1] V kriptografiji je nonce poljubno število, ki se lahko uporabi samo enkrat v kriptografski komunikaciji.

rov (ključ-vrednost) podatkov o nosilcu. Preverljive predstavitve imajo podobno strukturo z majhnimi razlikami v komponentah: metapodatki v predstavitvi določajo le vrsto predstavitve in pogoje uporabe; en ali več zahtevkov iz ene ali več poverilnic skupaj z njihovimi metapodatki in dokazili so prav tako del predstavitve; podpis v VP ima enako strukturo kot podpis v VC, vendar v tem primeru podpis ustvari nosilec kot dokaz, da je on tisti, ki je izdal predstavitev preveritelju.

3 PREDLOG VPELJAVE SSI PRI PODPORI ŠTUDIJSKEGA PROCESA

Z razvojem primera uporabe, ki temelji na DID, VC in VP na verigi blokov Cardano, smo predstavili vlogo samoupravljanih identitet v študijskem procesu oziroma pri izobraževanju. Tako študentom omogočimo: upravljati s svojimi osebnimi informacijami; dokazovati svojo identiteto; hitro in preprosto uporabniško izkušnjo, ki temelji le na uporabi digitalne denarnice.

Vse tehnologije, uporabljene pri razvoju rešitve, so izbrane tako, da so čim bolj združljive z rešitvijo, ki smo jo želeli razviti. Tako smo imeli možnost sistem razviti hitreje in ga narediti učinkovejšega, varnejšega in preprostejšega za uporabnike.

3.1 Obstoječe rešitve

Za osnovo naše aplikacije smo uporabili obstoječe sisteme. Pogled na ta sorodna dela nam je pomagal razumeti, kako bomo postopali pri razvoju našega primera uporabe.

Moodle/StudIS

Moodle [11] je centralizirana platforma, ki uporabnikom omogoča komunikacijo, opravljanje nalog in spremljanje njihovih dosežkov. StudIS [14] pa je centralizirana platforma, prek katere se študentje lahko prijavijo na izpite, preverijo nekatere svoje obveznosti za tekoče leto in se prijavijo v izbrane letnike študijskih programov. Po ideji sta ti aplikaciji najbližje rešitvi, ki smo jo ustvarili, vendar s to razliko, da implementirata model centraliziranih identitet. Mi pa smo izkoristili model decentraliziranih identitet in smo uporabljali SSI na verigi blokov, kar uporabnikom ponuja popolno avtonomijo nad lastno identiteto ter hitrejši in varnejši način upravljanja z lastnimi podatki.

EduCTX

EduCTX [16] predstavlja decentralizirano platformo, ki povezuje ustanove (univerze, podjetja itn.) in njihove člane ter jim omogoča varno komunikacijo oziroma izmenjavo in preverjanje certifikatov. Ta sistem uporablja model decentraliziranih identitet na verigi blokov Ethereum [8]. Pri razvoju naše rešitve pa smo uporabili SSI na verigi blokov Cardano ter tako podprli standarda W3C za DID in VC, kar uporabnikom omogoča vzpostavitev varnejših komunikacijskih kanalov med seboj ter lažjo, hitrejšo in varnejšo izmenjavo in preverjanje poverilnic (podatkov) prek teh kanalov. Velika prednost našega sistema je tudi implementacija preverljivih predstavitev, ki uporabnikom omogočajo izbiro le tistih atributov iz svojih poverilnic, ki jih želijo deliti.

Projekt DE4A

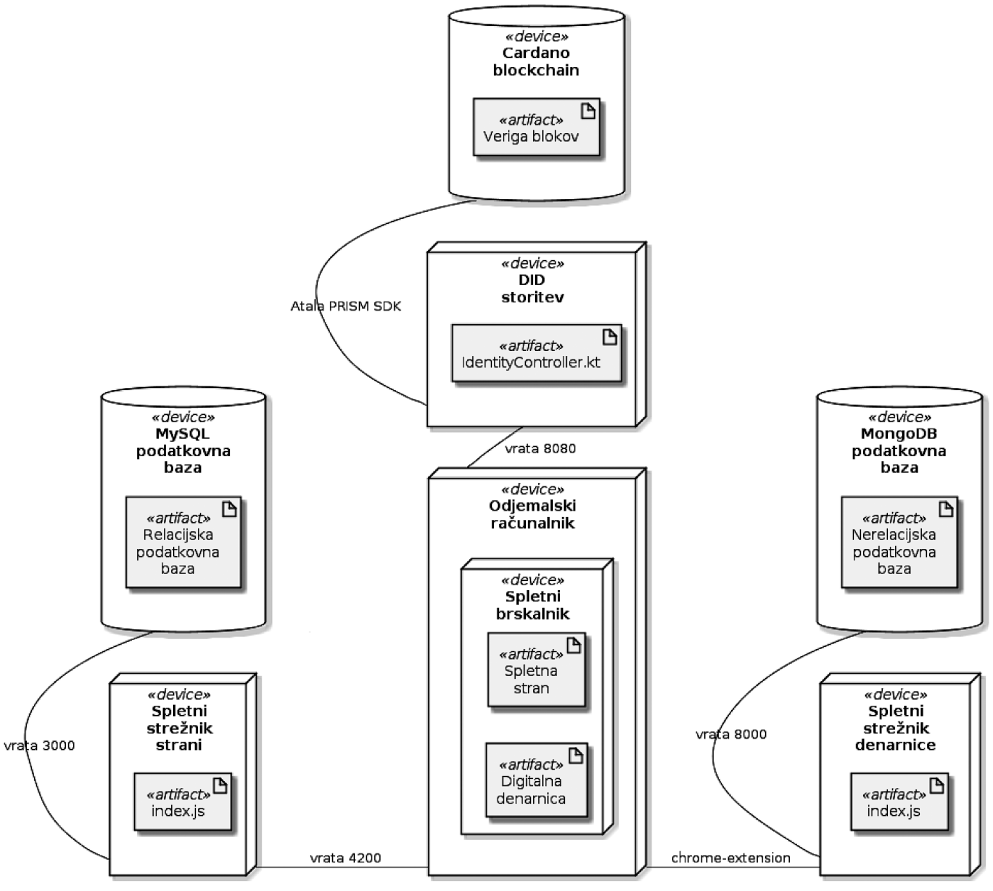
Projekt DE4A [13] vsebuje pilotne projekte različnih problemskih domen, ki temeljijo na čezmejni izmenjavi podatkov. Eden izmed njih je »Studying Abroad Pilot« [15], ki uporablja SSI oziroma DID, VC in VP za izmenjavo dokazil med posameznimi državami (univerzami), članicami EU. Primer uporabe, ki ga vključuje ter je po funkcionalnosti najbližje našemu, se imenuje »Diploma/Certs/Studies/Professional Recognition« [5] in se osredinja na priznavanje diplom, potrdil ali drugih dokazil o študiju ali tečajih. Ta primer uporabe uporablja SSI le za dokazovanje opravljenega izobraževanja, medtem ko smo obvladovanje SSI v naši rešitvi uredili za več področij, saj smo se osredinili tudi na izobraževalni proces. Z uporabo digitalne denarnice lahko študenti lažje, hitreje in varneje spremljajo svoje študijske obveznosti in potrdila v procesu študija.

3.2 Implementacija rešitve

Lastno idejo smo razvili na podlagi pregleda obstoječih rešitev in jo posodobili z uporabo SSI ter implementacijo DID, VC in VP. Glavni cilj aplikacije, ki smo jo razvili, je, da študentom ponuja naslednje možnosti: izdajo potrdil za opravljene obveznosti (izpiti, vaje, predavanja, naloge itn.), predmete; prijavo na posamezno dejavnost (izpit, kolokvij, test itn.); decentralizirano prijavo na spletni strani z uporabo DID; varno shranjevanje podatkov; preprost za uporabo in do uporabnika prijazen vmesnik; cenejšo in varnejšo uporabo verige blokov. Enkrat, ko je DID-povezava med dvema entitetama vzpostavljena (med profesorji

in študenti, med asistenti in študenti, med asistenti in profesorji, med dvema študentoma, dvema profesorjema, med študenti in delodajalci itn.), si lahko oba udeleženca v tej P2P-komunikaciji med seboj izmenjata VC in VP. Izdajatelj (profesor, asistent) v DID--komunikaciji bo nosilcu (študentu) poslal enega ali več VC, ki vsebujejo veljavne trditve o njem. Vsak VC bo podpisal izdajatelj, ki ga je izdal, kar bo omogočilo preveritelju preprosto verifikacijo veljavnosti atributov v VC. Iz ene ali več poverilnic, ki so mu bile izdane, bo lahko nosilec (študent) izbral samo potrebne atribute (podatke) in jih dodal v VP skupaj s podpisi iz vseh poverilnic, iz katerih je nosilec izbral atribute. Tako bo lahko nosilec dokazoval svojo identiteto (ime, priimek, EMŠO, datum rojstva itn.), svoje opravljene obveznosti (domače naloge, kolokvije, izpite, tečaje, programe itn.), svoj žeton dostopa do spletne strani (uporabniško ime in geslo za prijavo na spletno stran, žeton za avtentikacijo na spletni strani itn.). Preden nosilec predstavitev pošlje preveritelju (drugemu profesorju ali asistentu na isti fakulteti, drugi fakulteti ali univerzi, delodajalcu itn.), jo tudi podpiše, tako da bo lahko preveritelj preveril, ali so bili podatki v njej spremenjeni. Ko dobi VP, preveritelj preprosto prebere iz verige blokov DID nosilca in vseh izdajateljev, katerih podpisi so v VP. Tako dostopa do njihovih DID-dokumentov, da lahko prebere njihove javne ključe in preveri veljavnost vseh podpisov oziroma preveri, ali so podpisi res ustvarjeni z ustreznimi zasebnimi ključi in trenutno vrednostjo podatkov v VP. To pomeni, da mora nosilec tudi objaviti svoj DID na verigi blokov Cardano, da bi lahko ustvaril in predstavil VP preveriteljem. Glede na veljavnost podpisov in atributov se bo preveritelj odločil, ali bo nosilcu izdal poverilnico (potrdilo o opravljenem tečaju, potrdilo o opravljenem programu, potrdilo o diplomi, pogodbo o delu).

Naša rešitev je sestavljena iz različnih, medsebojno odvisnih komponent. Kot je vidno na postavitvenem diagramu (slika 4), potrebuje uporabnik na svojem računalniku le brskalnik, da lahko dostopa do spletne strani in digitalne denarnice. Ta dva sistema



Slika 4: **Postavitveni diagram**

komunicirata med seboj in z DID-storitvijo, kar uporabnikom omogoča interakcijo z lastnimi podatki.

Digitalna denarnica

Digitalna denarnica (*angl.* Digital wallet) je osrednji element in vmesnik za komunikacijo z verigo blokov. Običajno ima izbrana veriga blokov lastno denarnico, ki jo lahko uporabljajo vsi (razvijalci in uporabniki). Na žalost veriga blokov Cardano še nima digitalnih denarnic, ki bi podpirale DID, VC in VP. Zato smo ustvarili DID-storitev, ki namesto denarnice opravlja vse pomembne funkcionalnosti med entitetami. V naši rešitvi uporabljamo digitalno denarnico le kot vmesnik, ki nam prikaže, kako bi bil videti celotni sistem v praksi, če bi obstajala taka denarnica.

Razvili smo preprosto razširitev za brskalnik Chrome (*angl.* Chrome extension) z uporabo JavaScripta, CSS in HTML, ki shrani šifrirane poverilnice in podatke za generiranje javnega in zasebnega ključa v podatkovno bazo MongoDB. V praksi je shranjevanje kritičnih podatkov v centralizirani podatkovni bazi izjemno nevarno. Dejansko bi digitalna denarnica shranjevala podatke samo lokalno na napravi ali decentralizirano v obliki varnostne kopije oziroma na verigi blokov. V našem primeru poskušamo predstaviti le glavne funkcionalnosti DID, VC in VP. Poudarek torej ne bo na varnih načinih shranjevanja poverilnic in ključev pri delu z digitalno denarnico, vendar na razvoju storitve, ki bo omogočala varno in hitro izdajo, izmenjavo in potrjevanje poverilnic med entitetami pri izobraževanju.

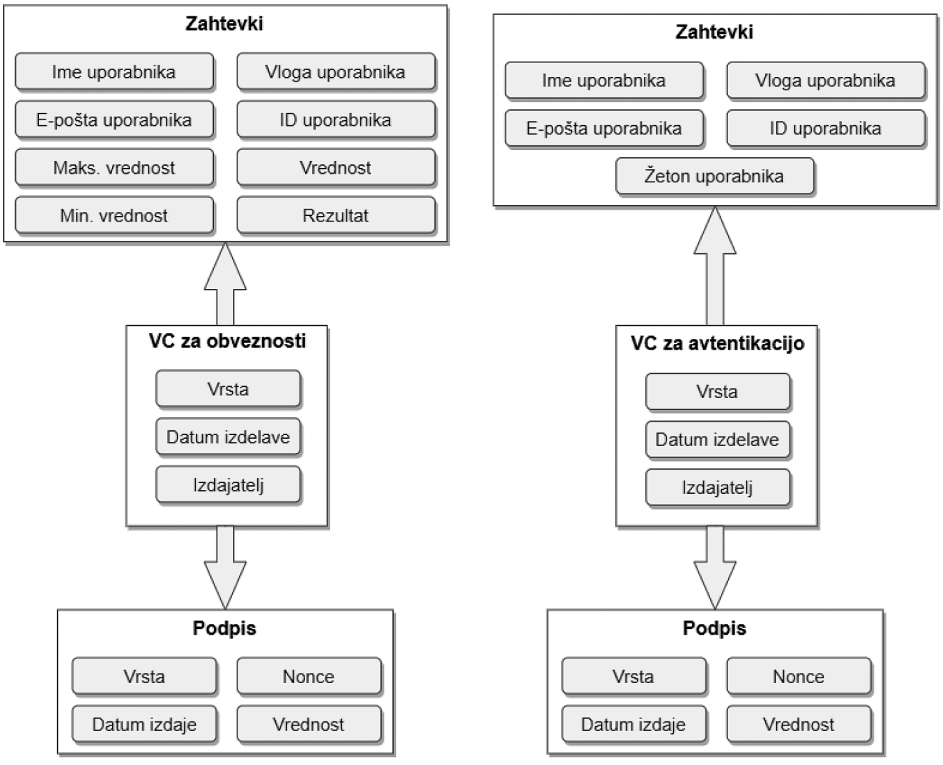
DID-storitev

Najpomembnejši del naše rešitve je DID-storitev. To je spletna storitev, ki je razvita v Kotlinu, je popolnoma decentralizirana in uporablja REST API, da bi lahko druga dva sistema komunicirala z njo. Implementira knjižnico Atala PRISM SDK [3], kar je eden izmed glavnih razlogov, zakaj smo izbrali verigo blokov Cardano, saj uporablja verigo blokov Cardano in nam omogoča cenejšo namestitev SSI ter implementacijo serije (*angl.* Batch), ki izdajateljem omogoča izdajo več poverilnic le z eno transakcijo. Atala PRISM SDK je trenutno na voljo le za razvijalce pionirskega programa Atala PRISM. Mogoče ga je preprosto implementirati v programske jezike, ki ciljajo na virtualno izvajalsko okolje Java (*angl.* Java-Virtual-Machine, JVM) in omogoča najpomembnejše funkcionalnosti skladno s standardom W3C za DID, VC in VP, kot so: generiranje, posodabljanje in preklic DID ter z njim povezan DID-dokument; objava DID na verigi blokov Cardano, kar je potrebno za poznejšo izdajo in preverjanje proverilnic; preverjanje stanja transakcije oziroma ali je transakcija uspešno objavljena na verigi blokov Cardano; branje DID-dokumenta, kar je mogoče, le če je DID že objavljen na verigi blokov Cardano; ustvarjanje, izdaja, preverjanje ter preklic poverilnice, serije in predstavitve.

Poleg implementacije velikega števila funkcionalnosti je bila ena najpomembnejših stvari določiti shemo VC v našem sistemu. Kot lahko vidimo na sliki 5, smo glede na vsebino (zahtevki) preverljive poverilnice razdelili na dve vrsti – VC za obveznosti in VC za avtentikacijo. Obe vrsti VC bosta imeli določene zahtevke enake, kot so: ime uporabnika na spletni strani, e-poštni naslov uporabnika na spletni strani, ID uporabnika na spletni strani in vloga uporabnika na spletni strani. Ti skupni atributi so v pomoč le na spletni strani, ki smo jo ustvarili v našem sistemu, da bi uporabnikom omogočili lažjo in varnejšo avtentikacijo in identifikacijo. VC za avtentikacijo je prav tako specifično za to spletno stran, saj bo vseboval tudi žeton v obliki niza, ki bo uporabniku omogočil prijavo v spletno stran s svojim DID. VC za obveznosti bo imela zahtevke, ki bodo omogočali njeno uporabo ne le na tej spletni strani: pogoj za opravljeno obveznost (maksimalna in minimalna vrednost) v numerični obliki; vrednost, ki jo je študent dosegel pri tej obveznosti v numerični obliki; rezultat v obliki niza (opravljeno ali neopravljeno). Vsi ti atributi bodo nosilcu omogočali, da s preveriteljem deli le rezultat in ali svojo oceno.

Spletna stran DIDEdu

Spletna stran DIDEdu je tudi vmesnik našega sistema, ki večinoma sodeluje z digitalno denarnico in s storitvijo DID ter ga uporabnik največ uporablja. Razvita je v Angularju (TypeScript, CSS in HTML) in uporablja podatkovno bazo MySQL za shranjevanje le najpotrebnejših podatkov o univerzah, fakultetah, uporabnikih, programih in o vsem, kar tovrstna spletna stran potrebuje za delovanje. To je aplikacija, ki omogoča uporabnikom uporabo vseh funkcionalnosti, ki jih SSI ponuja.



Slika 5: **Struktura obeh vrst preverljivih poverilnic v DID-storitvi**

Sistemu smo dodali ta centralizirani del, da bi izboljšali uporabniško izkušnjo, vendar smo razvili spletno stran na tak način, da uporabnik potrebuje svojo digitalno denarnico za uporabo funkcionalnosti spletne strani. To pomeni, da tudi če se tretja oseba prijavi prek računa nekoga drugega, ne bo mogla uporabljati funkcionalnosti spletne strani, če nima dostopa do digitalne denarnice, ki je povezana s tem računom.

4 OVREDNOTENJE

Izvorna koda rešitve je na voljo v repozitoriju Github [18], v katerem je predstavljen tudi kratek posnetek, ki prikaže delovanje najpomembnejših funkcionalnosti, ki jih rešitev implementira, kot so: DID-generiranje, preverjanje serije/poverilnice, izdaja serije/poverilnice, ustvarjanje predstavitve, preklic. Pri razvoju se nismo osredinjali na ustvarjanje aplikacije, pripravljene za produkcijsko okolje. Poudarek je bil na razvoju prototipa, ki nam pomaga predstaviti vse funkcionalnosti, ki nam jih ponuja implementacija decentraliziranih digitalnih identitet z uporabo najboljših praks in tehnologij.

V naslednjih razdelkih bomo s pomočjo SWOT-analize identificirali prednosti (*angl.* Strengths), slabosti (*angl.* Weaknesses), priložnosti (*angl.* Opportunities) in grožnje (*angl.* Threats) razvite rešitve v primerjavi s podobnimi projekti, ki smo jih že opisali v razdelku 3.1. Tako bomo povzeli splošne ugotovitve o uporabi SSI.

4.1 Prednosti

Primerjavo naše rešitve smo začeli z Moodle in s StudIS, ki sta zelo blizu naši aplikaciji, vendar obe uporabljata centraliziran model identitete. Glavne prednosti, ki smo jih ugotovili, so naslednje: naša rešitev uporablja verigo blokov, zaradi česar je gotovo varnejša; vse funkcionalnosti, ki jih ponujata oba sistema, so na voljo tudi v naši aplikaciji, kar študentom olajša dostop do njih, saj je vse potrebno pri študijskem procesu dostopno na enem mestu; ker uporabniki hranijo svoje podatke v obliki preverljivih poverilnic v lastni digitalni denarnici, centraliziranemu delu naše aplikacije ni treba hraniti toliko podatkov, kar poveča zmogljivost baze podatkov; uporabljali smo SSI, ki je vrsta decentraliziranih identitet, kar pomeni, da uporabnikom omogoča prenosljivo in ponovno uporabo lastne digitalne identitete in popolno avtonomijo nad lastnimi podatki (poverilnicami).

Nadaljujemo s primerjavo z decentraliziranim sistemom EduCTX, ki uporablja verigo blokov Ethereum: naša rešitev uporablja verigo blokov Cardano, ki je veriga blokov tretje generacije, kar pomeni, da se osredinja na stvari, kot so: razširljivost, trajnost in interoperabilnost; z implementacijo DID, VC in VP omogočimo uporabnikom varno in hitro identifikacijo, avtentikacijo in komunikacijo na spletu; implementacija Atala PRISM SDK v DID-storitvi nam omogoča hitrejšo in cenejšo namestitev SSI na verigi blokov Cardano.

Ko primerjamo naš sistem s primerom uporabe »Studying Abroad Pilot« iz projekta DE4A, ugotovimo naslednje prednosti: pri naši rešitvi ne uporabljamo DID in VC le za priznavanje opravljenega izobraževanja, ampak tudi za vse drugo, kar študenti potrebujejo v procesu študija; naš sistem implementira Atala PRISM SDK na verigi blokov Cardano, kar uporabnikom omogoča ustvarjanje serije oziroma izdajo več poverilnic z eno samo transakcijo.

4.2 Slabosti

Upoštevati moramo tudi slabosti uporabe te tehnologije. Pri primerjavi z Moodle in s StudIS ugotovimo naslednje slabosti: naša rešitev uporablja tudi verigo blokov, ki je sorazmerno nov koncept in ga ljudje še ne razumejo popolnoma ter zato ne uporabljajo decentraliziranih sistemov tako pogosto; uporabnik bo potreboval tudi digitalno denarnico in DID, da lahko izkoristi vse funkcionalnosti naše aplikacije.

Pri primerjavi naše rešitve z EduCTX ugotovimo, da Ethereum omogoča druge funkcionalnosti, zaradi katerih je veriga blokov Cardano lahko slabša; veriga blokov Cardano je še vedno v razvoju oziroma še vedno razvija nekatere funkcionalnosti, ki jih je Ethereum že razvil pred nekaj leti.

Slabosti, ki smo jih našli pri primerjavi naše rešitve s projektom DE4A, so naslednje: naša rešitev prikazuje DID v njihovi normalni obliki, kar ni dobra praksa, saj jih uporabnik ne more preprosto prebrati in uporabljati; kot smo že omenili v razdelku 3.2, veriga blokov Cardano še nima razvite delujoče digitalne denarnice, ki bi podpirala DID, VC in VP; ustvariti smo jo morali sami z uporabo centralizirane podatkovne baze MongoDB za shranjevanje šifriranih poverilnic, kar se v praksi ne bi zgodilo, saj omejuje nadzor uporabnika nad lastnimi podatki.

4.3 Priložnosti

Na splošno so priložnosti, za katere smo ugotovili, da prihajajo z uporabo samoupravljanih identitet na verigi blokov ter pri izobraževanju, naslednje: uporaba verige blokov namesto baze podatkov ni samo varnejša, ampak tudi boljša v smislu zmogljivosti, saj več kot je objavljenih blokov (podatkov), varnejša postane veriga blokov; uporaba DID, VC in VP v več aplikacijah poveča zanimanje razvijalcev in uporabnikov za razvoj in uporabo decentraliziranih sistemov, ki podpirajo SSI; uporaba SSI pri izobraževanju študentom omogoča lažji dostop in dokazovanje svojih izobraževalnih dosežkov do storitev v celotnem študijskem ekosistemu in širše; več implementacij SSI na spletu uporabnikom omogoča lažjo in širšo uporabo SSI ter varno, prenosljivo in ponovno uporabo lastnih podatkov v vsakem sistemu, v katerem so te tehnologije implementirane.

4.4 Grožnje

Obstaja tudi nekaj groženj pri uporabi SSI na verigi blokov in pri izobraževanju: uporabniki imajo večjo odgovornost za varnost svojih podatkov, saj so vse njihove poverilnice šifrirane in shranjene na njihovih osebnih napravah (digitalnih denarnicah); izguba ali deljenje zasebnega ključa oziroma dostopa do denarnice uporabnika prek tretjih oseb omogoča zlonamernim entitetam dostop do osebnih podatkov in kriptovalut lastnika; DID, VC in VP se začnejo uvažati v druge verige blokov, ki imajo mogoče večjo sprejetost uporabnikov kot Cardano, kar lahko povzroči, da razvijalci izberejo drugo verigo blokov; zaradi avtomatizacije številnih procesov s pomočjo SSI pri izobraževanju se bo brezposelnost povečala; Atala PRISM SDK uporablja testno omrežje verige blokov Cardano, kar lahko povzroči, da je naš sistem nestabilen pri funkcionalnostih, ki uporabljajo verigo blokov.

5 SKLEP

Decentralizirana aplikacija, ki smo jo razvili za izobraževalne institucije, študentom omogoča lažje, hitrejše in varnejše spremljanje svojih študijskih obveznosti ter potrdil. Razvoj te rešitve nam je pomagal predstaviti obvladovanje vrste modela decentraliziranih identitet, ki podpira DID, VC in VP pri izobraževanju. Vpeljava SSI na verigo blokov Cardano upo-

rabnikom omogoča: prenosljivost, varno in ponovno uporabo podatkov na spletu; uporabo Atala PRISM SDK oziroma implementacijo DID, VC in VP skladno s standardom W3C ter cenejšo namestitev SSI na verigi blokov; popoln nadzor nad lastnimi podatki. S pomočjo testiranja in SWOT-analize smo ugotovili, da je ta decentraliziran sistem dejansko hitrejši, varnejši in primernejši za študente kot nekatere že obstoječe rešitve. Seveda obstajajo nekatere slabosti, ki prihajajo z uporabo te tehnologije na verigi blokov Cardano, kot so: manjkajoča digitalna denarnica za delo z DID, VC in VP; funkcionalnosti, ki so še vedno v razvoju; uporaba testnega omrežja verige blokov Cardano; težko razumevanje tehnologije verige blokov pri uporabnikih in razvijalcih; večja odgovornost entitet za varnost njihovih osebnih podatkov.

Pokazali smo, da imajo samoupravljane identitete pomembno vlogo pri izobraževanju, vendar jih je vredno uporabiti v čim več ter različnih primerih uporabe. Decentralizacija ter s tem popolna avtonomija in varnost digitalnih identitet se morajo nenehno nadgrajevati, zato pa moramo zbuditi zanimanje več razvijalcev v razvoj decentraliziranih rešitev z uporabo te tehnologije. Več primerov uporabe kot povežemo z njo, večjo prenosljivost in več nadzora nad večjim številom lastnih podatkov bodo imeli uporabniki.

Če bi bila vsaka rešitev decentralizirana in bi uporabljala SSI oziroma DID, VC in VP kot način za izmenjavo, preverjanje in shranjevanje osebnih podatkov s pomočjo verige blokov, bi uporabniki potrebovali le svojo denarnico, da bi lahko varno in hitro delali z lastnimi podatki, kot želijo in kadar želijo. Tako se bo začela nova doba spleta, v kateri bo internet tisto, kar naj bi vedno bil – varno mesto, ki ljudem lajša življenje brez nevarnosti zlorabe njihovih podatkov.

LITERATURA

[1] Aggarwal, S., & Kumar, N. (2021). Chapter Sixteen – Hyperledger Working model. V S. Aggarwal, N. Kumar & P. Raj (Ur.), *The Blockchain Technology for Secure and Smart Applications across Industry Verticals* (str. 323–343). Elsevier. https://doi.org/https://doi.org/10.1016/bs.adcom.2020.08.016

[2] Ahram, T., Sargolzaei, A., Sargolzaei, S., Daniels, J., & Amaba, B. (2017). Blockchain technology innovations. *2017 IEEE Technology and Engineering Management Conference (TEMSCON)*, 137–141. https://doi.org/10.1109/TEMSCON.2017.7998367

[3] *Atala PRISM SDK*. (2022). Pridobljeno 26. septembra 2022, https://docs-ppp.atalaprism.io/

[4] Camp, J. (2004). Digital identity. *IEEE Technology and Society Magazine, 23*(3), 34–41. https://doi. org/10.1109/MTAS.2004.1337889

[5] *Use Case "Diploma/Certs/Studies/Professional Recognition" (SA UC3)*. (2022). Pridobljeno 26. septembra 2022, https://wiki.de4a.eu/index.php/Use_Case_%22Diploma/Certs/Studies/Professional_ Recognition%22_(SA_UC3)

[6] Fang, J., Yan, C., & Yan, C. (2009). Centralized Identity Authentication Research Based on Management Application Platform. *2009 First International Conference on Information Science and Engineering*, 2292–2295. https://doi.org/10.1109/ICISE.2009.382

[7] Isirova, K., & Potii, O. (2018). Decentralized public key infrastructure development principles. *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DES-SERT)*, 305–310. https://doi.org/10.1109/DESSERT.2018.8409149

[8] Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H.-N. (2022). Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract. *IEEE Access, 10*, 6605–6621. https://doi.org/ 10.1109/ACCESS.2021.3140091

[9] Laurent, M., & Bouzefrane, S. (2015). *Digital identity management*. Elsevier.

[10] *The Three Models of Digital Identity Relationships*. (2018). Pridobljeno 5. novembra 2022, https://medium.com/evernym/the-three-models-of-digital-identity-relationships-ca0727cb5186

[11] *Moodle documentation*. (2022). Pridobljeno 26. septembra 2022, https://docs.moodle.org/400/en/ Main_page

[12] Preukschat, A., & Reed, D. (2021). *Self-Sovereign Identity.* Manning.

[13] *DE4A Service Interoperability Solutions Toolbox)*. (2022). Pridobljeno 26. septembra 2022, https://wiki.de4a.eu/index.php/DE4A_Service_Interoperability_Solutions_Toolbox

[14] *StudIS FRI*. (2022). Pridobljeno 5. oktobra 2022, https://studisfri.uni-lj.si/Account/Login?ReturnUrl=%2f

[15] *Studying Abroad Pilot*. (2022). Pridobljeno 26. septembra 2022, https://wiki.de4a.eu/index.php/ Studying_Abroad_Pilot

[16] Turkanovic´, M., Hölbl, M., Košič, K., Heričko, M., & Kamišalic´, A. (2018). EduCTX: A Blockchain- Based Higher Education Credit Platform. *IEEE Access, 6*, 5112–5127. https://doi.org/10.1109/ ACCESS.2018.2789929

[17] van Bokkem, D., Hageman, R., Koning, G., Nguyen, L., & Zarin, N. (2019). Self-Sovereign Identity Solutions: The Necessity of Blockchain Technology. https://doi.org/10.48550/ARXIV.1904.12816

[18] Vasilev, N. (2022). *DIDEdu-DiplomskoDelo* (Ver. 2.0.4). Pridobljeno 26. septembra 2022, https : //github.com/nikolayVv/DIDEdu

[19] *Decentralized Identifiers (DIDs) v1.0.* (2021). Pridobljeno 26. septembra 2022, https://www.w3.org/ TR/did-core/

[20] *Verifiable Credentials Data Model v1.1*. (2022). Pridobljeno 26. septembra 2022, https://www.w3.org/ TR/vc-data-model/

[21] *W3C Standards.* (b.d.). Pridobljeno 22. septembra 2021, https://www.w3.org/standards/

[22] Yan, L., Rong, C., & Zhao, G. (2009). Strengthen Cloud Computing Security with Federal Identity Management Using Hierarchical Identity-Based Cryptography. V M. G. Jaatun, G. Zhao & C. Rong (Ur.), *Cloud Computing* (str. 167–177). Springer Berlin Heidelberg.

▪

Nikolay Vasilev je diplomiral leta 2022 na fakulteti za računalništvo in informatiko Univerze v Ljubljani. Njegova raziskalna zanimanja vključujejo decentralizirane sisteme, varnost, biometrijo in decentralizirane identitete.

▪

Dejan Lavbič je izredni profesor na fakulteti za računalništvo in informatiko Univerze v Ljubljani. Njegova raziskovalna področja so: kakovost informacij, semantični splet, Splet3, decentralizirane verige blokov in decentralizirane identitete. Sodeloval je v številnih EU- in nacionalni projektih ter ima dolgo zgodovino sodelovanja z gospodarstvom..

Pregled in analiza tehnoloških skladov za implementacijo sodobnih IT-arhitektur velepodatkov

Martina Šestak, Muhamed Turkanović
Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru
martina.sestak@um.si, muhamed.turkanovic@um.si

Izvleček

Dandanes se pri implementaciji sodobnih IT-arhitektur velepodatkov podjetja odločajo za uporabo različnih tehnoloških skladov, ki so bodisi odprtokodni bodisi takšni, ki jih na trgu ponujajo oblačni ponudniki, kot so Google, Microsoft, Amazon idr. Na izbiro določenega sklada vpliva nekaj različnih dejavnikov, pogosto pa se najpomembnejši izkažejo človeški viri in strošek uporabe posameznega sklada. Kot alternativa se za zmanjšanje stroškov lahko uporabijo odprtokodne tehnologije, kot je sklad Apache, vendar tudi to prinaša določene implikacije in kompromise. Sodobne arhitekture velepodatkov pa včasih vključujejo ločeni ravni za shrambo in analitiko, pri čemer se na vsaki ravni uporablja različna tehnološka rešitev (tudi znotraj posamezne ravni), a vse z namenom shranjevanja različno strukturiranih oz. nestrukturiranih podatkov in učinkovite analize le-teh. V članku predstavljamo primer dvostopenjske IT-arhitekture, optimizirane za hrambo in analizo velepodatkov. Prav tako prikazujemo orodja in rešitve znotraj izbranih treh tehnoloških skladov (Google, Amazon, Apache), s katerimi se lahko implementira omenjena arhitektura. Analiziramo lastnosti posameznih skladov ter podajamo povzetek prednosti in izzivov pri uporabi določenega sklada.

Ključne besede: IT-arhitektura, velepodatki, masovni podatki, podatkovno skladišče, podatkovne baze, širokostolpčne hrambe, tehnološki sklad

STATE-OF-THE-ART ANALYSIS OF TECHNOLOGY STACKS FOR THE IMPLEMENTATION OF MODERN BIG DATA ARCHITECTURES

Abstract

Every individual and company perceives the dimensions of big data a bit differently. Big data is not measured as 5 hard drives nor as 5 data barrels, neither is flow of data measured in litres. Big data is a mindset that forms the foundation for data-driven business. For mass data to be truly introduced to and used in business, one must first understand it. Only then can we expect business and technology to co-exist and deliver added value. In this article, I present all dimensions of big data (i.e., 5 V's) and shed light on its use in practical examples. I also present the broader ecosystem of big data and the foundations for building an environment for big data.

Keywords: IT architecture, big data, data warehouse, database, wide-column databases, technology stack

1 UVOD

Pojem velepodatkov ali masovnih podatkov, ki se že nekaj časa uporablja kot ena ključnih besed v domeni podatkovnih tehnologij, je v zadnjem desetletju pospešil razvoj novih tehnoloških rešitev za učinkovito upravljanje naraščajočih količin podatkov, ki naslavljajo izzive, kot so razširljivost, količina, hitrost, različnost in drugo. Po trenutnih statistikah se vsak dan proizvede približno 2,5 kvintiljona (1018) bajtov podatkov, sama industrija velepodatkov pa zadnjih nekaj let vedno bolj narašča in je trenutno vredna rekordnih 274 milijard ameriških dolarjev [10]. Slednje

številke potrjujejo, da morajo podjetja neprekinjeno vlagati v sodobne tehnološke rešitve, ki jim lahko olajšajo spopadanje z izzivi v obdobju velepodatkov.

Obenem pa se podjetja, ki v vsakdanjem poslovanju obravnavajo veliko količino podatkov, pogosto srečajo s potrebo po prilagoditvi obstoječih informacijskih sistemov in integracijo novih rešitev z le-temi. Pri tem se večina finančnih sredstev uporablja za namen transformacije obstoječih informacijskih rešitev ob upoštevanju sodobnih pristopov v načrtovanju IT-arhitektur IKT-sistemov. Sodobne zahteve namreč določajo, da podatke generirajo različni elementi IKT-rešitve oz. sistema, kar pomeni, da v takšnih sistemih hitro nastopi beleženje velikih količin podatkov, da se ti generirajo z veliko hitrostjo in s strani različnih virov ter da so podatki vedno bolj nestrukturirane oblike. Slednje so obstajale tudi prej, vendar v manjših količinah, a jih nismo znali obdelati oz. po njih poizvedovati, kar pa se korenito spreminja z vedno večjim prodorom tehnik umetne inteligence.

Kot največji izziv pri vpeljavi sodobnih tehnoloških rešitev za upravljanje velepodatkov podjetja izpostavljajo dodeljena finančna sredstva za takšne projekte ter tehnične izzive pri integraciji z obstoječo infrastrukturo podjetja [10]. Ne moremo pa tudi zanemariti kompleksnosti vedno bolj pomembnega področja podatkovnega inženirstva (angl. data engineering), ki je tesno povezano s področjem podatkovne znanosti (angl. data science). S tema področjema so povezani tudi trije profili strokovnjakov, in sicer podatkovni inženir ter podatkovni znanstvenik in podatkovni analitik. Slednja izvajata aktivnosti, ki so neposredno povezane z metodami umetne inteligence, strojnega učenja, podatkovnega rudarjenja oz. s področjem poročanja in vizualizacije. V primerjavi s tem pa je naloga podatkovnega inženirja gradnja in vzdrževanje logičnih in fizičnih podatkovnih modelov ter s tem povezanih podatkovnih cevovodov. Osredotočajo se na celovito IT-arhitekturo za upravljanje velepodatkov, kakor tudi s tem povezano preoblikovanje, migriranje in združevanje podatkov ter zagotavljanje kakovosti podatkov.

Za zagotavljanje ustreznega in učinkovitega upravljanja velepodatkov je treba oblikovati celotni podatkovni cevovod (en ali več) ter opredeliti posamezne rešitve kot elemente le-tega. Ta korak predstavlja včasih izziv celo za strokovnjake, saj je težko izbrati rešitve, ki bodo med seboj ujemajoče in bodo v celoti zajemale zahteve uporabnikov. Dodaten izziv je izbira ene rešitve med številnimi možnostmi, ki so na voljo na trgu za določen namen (npr. za zajem podatkov). Večja tehnološka podjetja so razvijala rešitve, s katerimi so želela rešiti specifične izzive velepodatkov, s katerimi se srečujejo. Takšen pristop je predstavljal veliko različnih tehnologij in orodij na trgu, pri čemer so razlike med njimi v najmanjši meri oz. jih ni možno ločiti brez podrobne analize rešitev. Posledično izbira tehnološkega sklada pri projektu implementacije arhitekture za upravljanje velepodatkov ni odvisna le od stroškov implementacije, temveč tudi od drugih dejavnikov, kot so potreba po realnočasovni obdelavi, enostavnost integracije z obstoječimi IKT-rešitvami v podjetju, način interakcije s sistemom, časovna zahtevnost vzpostavitve cevovoda itn.

V nadaljevanju bomo predstavili osnovne izzive sistemov za upravljanje velepodatkov ter način naslavljanja teh izzivov sodobne arhitekture. Poglobili se bomo v lastnosti rešitev znotraj tehnoloških skladov, ki jih v sklopu svojih oblačnih storitev ponujajo podjetja Google in Amazon ter odprtokodne alternativne rešitve znotraj sklada Apache. Podrobna analiza zmogljivosti izbranih tehnoloških skladov lahko pomaga pri izbiri določenega tehnološkega sklada za implementacijo na praktičnih primerih.

2 ZAHTEVE IN IZZIVI SISTEMOV ZA UPRAVLJANJE VELEPODATKOV

Sodobni sistemi za upravljanje velepodatkov se srečajo s številnimi izzivi pri zagotavljanju funkcionalnosti, ki jih zahteva določena domena uporabe. Ena izmed od funkcionalnosti je zagotavljanje sledljivosti vsakega podatkovnega toka [9], in sicer beleženje vsake spremembe nad posameznim podatkovnim zapisom v toku (tj. pogosto v dnevniške datoteke). S tem se pridobi večji nadzor nad podatki, ki tečejo v sistemu in omogoči transparentnost celotnega cevovoda.

Kot največji izziv v obdobju velepodatkov se vedno omenjata varnost in kakovost podatkov [3, 28], za kateri je pomembno vzpostaviti ustrezne mehanizme za upravljanje podatkov v sistemu (angl. data governance), kar npr. vključuje nadzor nad uporabo podatkov (tj. kdo, kaj, kdaj, kje in za kateri namen uporablja določeni podatkovni zapis). Varnost in zasebnost podatkov sta kot izziv izpostavljeni tudi v [1], kjer so avtorji predstavili pregled najbolj pogosto omenjenih izzivov v literaturi. Med drugim je varnost podatkov kot večji izziv omenjena v 78 % štu-

dij. Pri tem se s stališča kakovosti podatkov pogosto pojavljajo težave z nekonsistentnostjo podatkov med različnimi komponentami sistema. S stališča IT-arhitekture pa predstavlja večji izziv pravočasnost (angl. timeliness) oz. pravočasno dostopni podatki v določenem koraku cevovoda, ko so ti pričakovani in nujni. Slednje pomeni, da morajo sistemi zagotoviti zelo majhne zakasnitve v vsakem koraku podatkovnega cevovoda, sploh v primeru visokotveganih sistemov (npr. vgrajeni sistemi v avtomobilih).

Naslednji večji izziv v kontekstu sodobnih sistemov velepodatkov je deljenje podatkov, sploh med različnimi oddelki znotraj podjetja. V današnjem okolju več oddelkov znotraj podjetja želi dostopati do iste množice podatkov, ki pa so shranjeni zunaj njihovih IKT-rešitev. V takšnem primeru nastajajo t. i. podatkovni silosi (angl. data silos), kjer ima vsak oddelek svoj interni sistem, ki beleži podatke le 'lokalno', pri čemer se postopek, da se dostop do teh podatkov omogoči tudi zunanjim IKT-rešitvam, lahko močno zaplete. Zaradi tega so zaželene bolj decentralizirane arhitekture, kot je t. i. arhitektura data mesh. V vsakem primeru je treba vzpostaviti učinkovite varnostne mehanizme za nadzor nad deljenjem podatkov, ki jih določajo različne politike, opredeljene kot del upravljanja podatkov.

Za učinkovito deljenje podatkov je treba imeti v mislih tudi optimalno načrtovanje IT-arhitekture glede na potencialna težišča podatkov (angl. data centers of gravity). Težišča podatkov so točke v sistemu, kjer pričakujemo shranjevanje velikih količin podatkov, ki se pozneje premestijo v drugi del sistema (npr. v rešitev za podatkovno analitiko), pri čemer postopek migracije lahko močno vpliva na učinkovitost sistema. Da bi se temu izognili, morajo podatkovni arhitekti pri načrtovanju IT-arhitekture imeti v mislih takšne potencialne točke in oblikovati IT-arhitekturo na način, da je takšnih točk v sistemu čim manj ter da je učinek težnosti podatkov (angl. data gravity) čim manjši. V zvezi s tem se težave pojavljajo tudi zaradi porazdeljenosti okolja oz. podatkov v sistemu [32]. V določenem trenutku je namreč podmnožica podatkov, ki je nujna za izvedbo določene analize, lahko na enem vozlišču v gruči, medtem ko je drugi podnabor na drugem vozlišču. Težava pa nastane, ko je treba v koraku analize uporabiti oba nabora, ki sta logično in velikokrat tudi fizično shranjena na različnih lokacijah v gruči oz. sistemu, in je treba izvesti migracijo potencialno velikih količin podatkov, ne da bi se občutno zmanjšala učinkovitost drugih komponent sistema.

Pri razvoju sistemov za upravljanje velepodatkov se IT-arhitekti in inženirji srečajo s številnimi praktičnimi izzivi, ki nastajajo zaradi lastnosti velepodatkov in hitrega razvoja področja [11]. Eden od izzivov je seveda skaliranje IT-arhitekture, kjer se predvsem usmerimo v horizontalno razširljivost (angl. scale-out). Tukaj nastane izziv, kako določiti optimalno razmerje med kakovostjo podatkov in učinkovitostjo oz. dostopnostjo sistema. Takšna zahteva izhaja tudi iz izbire dveh od treh možnih lastnosti porazdeljenih sistemov, ki ju po teoremu CAP[1] lahko naenkrat zagotovimo. Izbira boljše kakovosti podatkov v sistemu ali učinkovitosti sistema je stvar kompromisa in je zelo odvisna od domenskih zahtev. Naloga IT-arhitektov je načrtovati sistem na način, da poskusijo zagotoviti čim manjše zamude in večjo prepustnost sistema, hkrati pa v določeni meri obdržati konsistentnost podatkov, kolikor je to le možno. Novost področja in nenehno uvajanje novih konceptov, arhitektur in tehnoloških rešitev vpliva na strmo naraščajočo krivuljo učenja pri strokovnjakih, saj je vedno zahtevnejše slediti novostim in najboljšim praksam na področju, tehnološke rešitve pa so vedno bolj zapletene za učenje in implementacijo.

Sodobni sistemi morajo biti zmožni zajemati podatke iz več različnih virov. V tem procesu za integracijo podatkov težave lahko povzroča manjkajoča ustrezna infrastruktura, kar je posebej pomembno v okoljih z različnimi viri podatkov. Slaba implementacija na ravni zajema podatkov predstavlja napačne rezultate analize teh podatkov oz. napačne poslovne odločitve. Hkrati je treba poudariti izziv, da se pri sodobnih IKT-rešitvah, ki so označene kot sistemi velepodatkov, srečujemo tudi z različnimi vrstami podatkov oz. z vedno več pol in nestrukturiranimi podatki (npr. dnevniški zapisi, slike, avdio-video zapisi, prost govor itn.). Nedavno smo se na področju IKT osredotočali zgolj na strukturirane podatke in načine, kako te učinkovito hraniti z uporabo relacijskih modelov ter kako te uporabiti za namen podatkovne analitike, pri čemer smo se kot na orodje za

[1] Teorem CAP (angl. Consistency, Availability, Partition tolerance) pravi, da lahko vsak porazdeljen sistem istočasno zagotavlja le dve od treh možnih lastnosti: celovitost oz. konsistentnost podatkov, razpoložljivost oz. dostopnost sistema ter odpornost sistema na particioniranje.

izvedbo analitike osredotočali na poizvedovalne jezike, kot so SQL, ki so idealni za strukturirane podatke. Nestrukturirane podatke smo rahlo zanemarjali, saj jih nismo znali primerno in učinkovito uporabiti za odkrivanje novih spoznanj. Danes imamo željo in potrebo po tem, da hranimo tudi nestrukturirane podatke, nad katerimi izvajamo podatkovno analitiko, in sicer s pomočjo tehnik podatkovnega rudarjenja, strojnega in globokega učenja itn. Hkrati smo te podatke in to vrsto podatkovne analitike, ki še zmeraj spada pod t. i. OLAP, izvajali zgolj ad-hoc, danes pa so težnje, da se ti podatki in ta vrsta analitike vključi v produkcijske sisteme, ki so povezani s t. i. transakcijsko obdelavo (tj. OLTP). To področje se danes hitro razvija in spada tudi pod t. i. inženiring UI (angl. AI engineering). IT-arhitekti so tako pred kompleksnim izzivom podatkovnega inženirstva, ki je zagotavljati učinkovito podatkovno raven za produkcijske sisteme, večinoma strukturirane podatke in OLTP, ki zahtevajo visoko razširljivost, ter hkrati učinkovito podatkovno raven za analitične procese, nestrukturirane podatke in OLAP, kjer pa se velikokrat srečujemo z masovnimi podatki. Povrh vsega pa se pojavlja težnja za IT-arhitekturo, ki bo omogočala hkratno in učinkovito obdelavo OLTP ter OLAP nad enotno podatkovno ravnjo.

Med trenutnimi izzivi podatkovnega inženirstva so tudi zahteve sodobnih IKT-rešitev in storitev, ki se navezujejo na realnočasovno obdelavo podatkov. Za doseganje ciljev takšnih zahtev se pojavljajo temu primerne podatkovne platforme, kot so platforme sporočilnih sistemov in pretakanja podatkov (npr. Apache Kafka). S tem povezano se tudi pojavljajo vnaprej definirani arhitekturni vzorci, kot sta Lambda in Kappa, ki se osredotočata zgolj na učinkovito obdelavo podatkovnih tokov in pretakanje podatkov [29]. Ker je to precej specifično področje, ki velja zgolj za poslovne primere, kjer je dejanska potreba po realnočasovnem odzivanju na dogodke, se v tem članku na to področje (vele-)podatkovnega inženirstva ne osredotočamo.

Ne nazadnje se sodobni sistemi kot IKT-rešitve srečujejo tudi z nezanemarljivimi stroški vzpostavitve infrastrukture s stališča strojne in programske opreme ter stroški vzdrževanja in človeških virov, saj znajo biti ti zelo visoki ne glede na uporabo oblačne ali lastne infrastrukture (angl. on premise). Takšne sisteme je težko ustrezno nadzorovati, saj je veliko različnih aktivnih komponent v sistemu, nad katerimi je treba v vsakem trenutku imeti popoln nadzor zaradi hitrega odkrivanja okvar v sistemu [11].

3 SODOBNE ARHITEKTURE ZA UPRAVLJANJE VELEPODATKOV

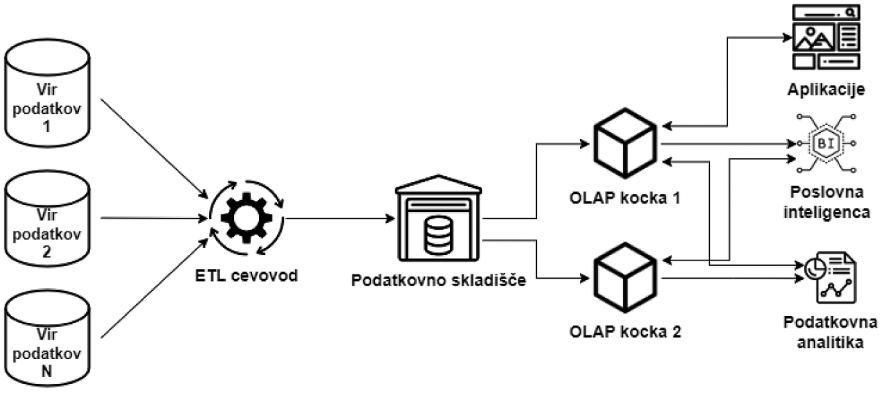
Celovito upravljanje velepodatkov dosežemo z IT-arhitekturo sistema, ki opredeli rešitve in postopke na ravneh zajemanja, shranjevanja, obdelave, vizualizacije in analize velikih količin (potencialno) kompleksnih podatkov [27]. Največja izziva predstavljata zagotavljanje učinkovitega shranjevanja in obdelave velepodatkov, saj zaradi slabega načrtovanja IT-arhitektur nastanejo nemalokrat ozka grla, ki nato vplivajo na celotno učinkovitost sistema. Strokovnjaki si pri načrtovanju podatkovnih cevovodov lahko pomagajo z določenimi smernicami posameznih IT-arhitektur, ki so predstavljene v nadaljevanju.

Za shranjevanje podatkov se uporabljajo različne tehnologije, ki so se razvijale več let in za različne namene. Tako se za shranjevanje transakcijskih podatkov, razen tradicionalnih relacijskih podatkovnih baz, danes pogosto kot zamenjava ali dodatna podpora uporabljajo nerelacijske podatkovne baze (NoSQL), kot sta podatkovna baza ključ-vrednost (npr. Redis) ali dokumentna podatkovna baza (npr. MongoDB), ki rešujejo izzive razširljivosti in prilagodljivosti podatkovne sheme, s katerima se v današnjem okolju srečujejo razvijalci IKT-rešitev. Zahteva večine IKT-rešitev je predvsem podpora za obdelavo OLTP[2], pri čemer je v večini primerov cilj zagotoviti čim večjo konsistentnost podatkov.

Danes imajo večja podjetja več transakcijskih podatkovnih baz oz. podatkovnih zbirk različnih tipov, ki jih je po določenem obdobju treba združiti zaradi izvedbe statističnih analiz in napredne podatkovne analitike, ki so lahko ustrezna podlaga za prihodnje poslovne odločitve, in se podpirajo s pomočjo področja poslovnega obveščanja (angl. business intelligence, BI). Za ta namen je treba ustrezno shraniti združene podatke na eni lokaciji, tj. podatkovno skladišče (angl. data warehouse). Osnova podatkovnih skla-

[1] Teorem CAP (angl. Consistency, Availability, Partition tolerance) pravi, da lahko vsak porazdeljen sistem istočasno zagotavlja le dve od treh možnih lastnosti: celovitost oz. konsistentnost podatkov, razpoložljivost oz. dostopnost sistema ter odpornost sistema na particioniranje.

[2] OLTP (angl. Online Transactional Processing) – način obdelave podatkov kot transakcij, pri čemer se le-te v sistemu shranjujejo in uporabljajo pri obdelavi v realnem času.



Slika 1: **Primer arhitekture z uporabo podatkovnega skladišča.**

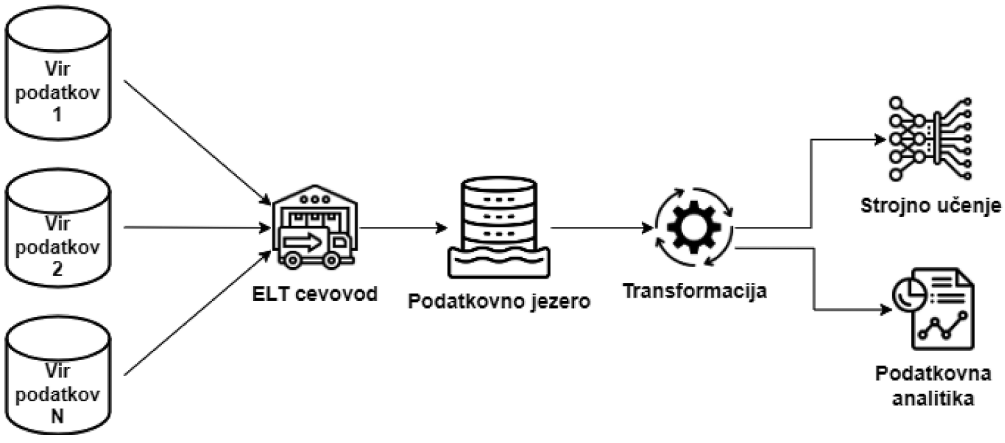
dišč so dimenzijski modeli, ki so v nasprotju z relacijskimi modeli namenski in usmerjeni neposredno v zahteve analitike oz. poslovnega obveščanja. Primer takšne arhitekture je prikazan na sliki 1. Podatki prihajajo iz več podatkovnih virov v cevovod ETL[3], kjer se izvajajo določene transformacije in čiščenje teh v skladu s podatkovno (dimenzijsko) shemo ciljnega podatkovnega skladišča. Prečiščeni podatki se nato shranjujejo v podatkovno skladišče, na podlagi katerega se oblikujejo t. i. kocke OLAP[4] namenjene določenim analizam nad podatki v agregirani obliki. Kocke OLAP vsebujejo podmnožico podatkov, shranjenih v podatkovnem skladišču, in predstavljajo vir podatkov za orodja za poslovno obveščanje ali poročanje. Kot največja izziva pri takšni IT-arhitekturi se poudarjata implementacija učinkovitega procesa ETL in vzpostavljanje mehanizmov za upravljanje kock OLAP v skladu s spremembami pri virih podatkov. Uporabnikom je potem omogočen dostop le do agregiranih podatkov za namene analitike, ne pa tudi izvirnih transakcijskih podatkov.

Zaradi vpliva velepodatkov vlogo podatkovnih skladišč danes prevzemajo širokostolpčne shrambe (angl. wide-column stores), kot sta HBase ali Cassandra, pri katerih so implementirane določene izboljšave za izvajanje analitike (tj. OLAP), ki pozitivno vplivajo na hitrost sistemov in omogočajo shranjevanje transakcijskih ter agregiranih podatkov.

V zadnjem desetletju so se za shranjevanje velepodatkov začela uporabljati tudi podatkovna jezera (angl. data lake) (slika 2), ki omogočajo shranjevanje masovne količine izvirnih (angl. raw) in različno strukturiranih podatkov brez vnaprej določene podatkovne sheme. Kot osrednji repozitoriji izvirnih podatkov so podatkovna jezera primerna za izvedbo zapletenejših analiz z uporabo strojnega učenja. Posledično se najbolj pogosto uporabljajo za podatkovne znanosti, vendar podjetja kot njihovo glavno slabost vidijo odsotnost mehanizmov za zagotavljanje konsistentnosti podatkov, tj. podatkovna jezera ne podpirajo modela ACID kot podatkovna skladišča ali baze[5]. Pri podatkovnih jezerih je cilj čim prej shraniti podatke s pomočjo vzpostavljenega cevovoda ELT[6] cevovoda. Ob branju shranjenih podatkov se podatki transformirajo iz surove oblike (npr. objekti) v obliko, ustrezno za namen uporabe (podatkovne analitike ali strojnega učenja).

Čez leta praktične uporabe se je izkazalo, da današnja podjetja nimajo eksplicitno ločene uporabe transakcijskih od agregiranih podatkov, zaradi česar uporaba le ene predhodno omenjenih tehnologij ne zadostuje, da bi podjetja na optimalen način hkrati zadovoljila potrebe OLTP in OLAP pri upravljanju podatkov. Hkrati je zaradi zahtev po visoki razširljivosti sistema edini primeren način razširjanje navzven (tj. horizontalno), kjer pa so seveda izzivi povezani s teoremom CAP in omejitve relacijskih podatkovnih baz, ki ne podpirajo visoke razširljivosti in hkrati dostopnosti oz. omejitve nerelacijskih podatkovnih baz, ki ne podpirajo visoke razširljivosti in hkrati konsisten-

[3] ETL (angl. Extract-Transform-Load) – proces zajema podatkov iz izvirnega sistema, čiščenja in transformacije prevzetih podatkov glede na ciljni (dimenzijski) podatkovni model ter uvoz pripravljenih podatkov v ciljno shrambo.
[4] Kocka OLAP (angl. Online Analytical Processing cube) – podatkovna struktura, ki omogoča večdimenzijski vpogled v podatke in hitro analizo le-teh.
[5] Model ACID (angl. Atomicity, Consistency, Isolation, Durability) – transakcijski model, s katerim npr. relacijske podatkovne baze zagotavljajo celovitost, konsistentnost, hkratnost in trajnost podatkov v podatkovni bazi.
[6] ELT (angl. Extract-Load-Transform) – proces zbiranja in shranjevanja podatkov v podatkovno jezero, pri čemer se transformacija podatkov izvaja šele ob branju podatkov.



Slika 2: **Primer arhitekture z uporabo podatkovnega jezera.**

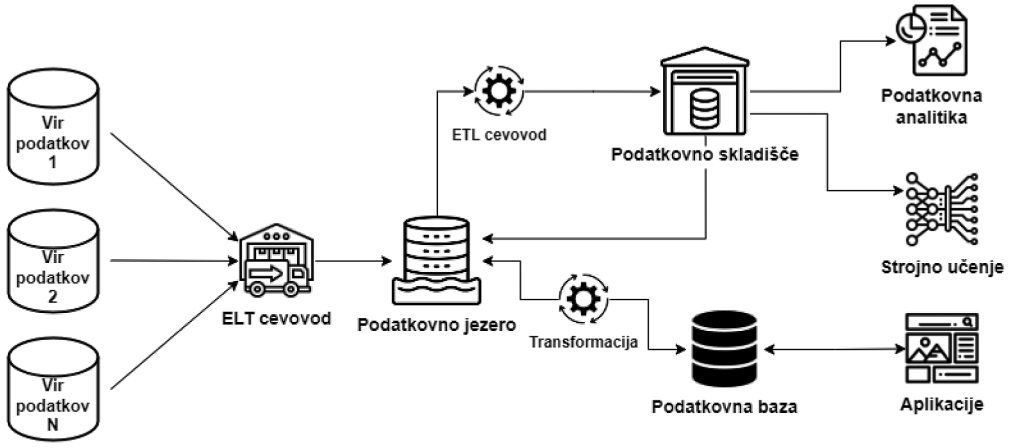
tnosti. Posledično se danes pogosto uporablja dvostopenjska (angl. 2-tier) arhitektura, prikazana na sliki 3, ki vključuje podatkovno jezero in skladišče, s čimer podjetja združijo prednosti obeh tehnologij. V tem primeru se vsi vhodni (transakcijski) podatki shranjujejo v podatkovnem jezeru, del teh pa se agregira in shrani v podatkovno skladišče za potrebe OLAP, od koder se uporabljajo za potrebe poslovnega obveščanja. Vmes pa so vsi podatki shranjeni v izvirni obliki v podatkovnem jezeru in jih lahko podatkovni znanstveniki kadar koli uporabijo za izvedbo zapletenejših algoritmov.

Trenutno je dvostopenjska arhitektura priljubljena izbira podatkovnih arhitektov, saj podjetju omogoča shranjevanje različno strukturiranih podatkov v podatkovnem jezeru, ki so uporabni v sedanjosti in prihodnosti. Dodatni sloj podatkovnega skladišča prinaša podporo za obdelavo podatkov OLAP oz. omogoča izvedbo učinkovitih analiz nad agregiranimi podatki. Podatkovno skladišče se, kot že prej omenjeno, lahko zamenja tudi s širokostolpčno podatkovno bazo. Kot pogosti vzorec se pojavlja tudi vključevanje transakcijske podatkovne baze nad jezerom, s katerimi podjetja lahko izboljšajo učinkovitost obstoječih podatkovnih baz na ravni upravljanja velepodatkov in tudi pohitrijo postopek transformacije surovih podatkov iz jezera za namen podatkovne analitike. V tem primeru je podatkovna baza odložišče podatkov iz jezera, ki so vnaprej transformirani in pripravljeni za analitiko. Uporabljajo se pa tudi IT-arhitekture, kjer pa se transakcijski podatki primarno shranjujejo v relacijsko podatkovno bazo in hkrati tudi v podatkovno jezero.

Čeprav so zajete zahteve z dvostopenjsko arhitekturo, se podjetja srečujejo z določenimi slabostmi takšne implementacije. Največji izziv je zapleteno upravljanje rešitev na obeh stopnjah, kar pomeni, da je treba zagotoviti vire oz. strokovnjake, ki bodo skrbeli, da podatkovno jezero in skladišče oz. tako OLTP kot OLAP delujeta nemoteno. Največ časa pri takšni IT-arhitekturi je treba vložiti v implementacijo zapletenih poslov ETL in ELT, s katerimi se morajo podatki iz podatkovnega jezera transformirati v ustrezni podatkovni model podatkovnega skladišča [2]. Dve ravni shranjevanja pomenita, da se čas izvedbe povpraševanj na strani uporabnikov lahko močno podaljša, saj podatkovna jezera, kakor tudi podatkovna skladišča, imajo določeno mejo, do katere je možno izboljšati učinkovitost izvedbe povpraševanj.

Da bi se izognili potencialno zapletenemu vzdrževanju dvostopenjske arhitekture, so se v novejšem času začela uporabljati t. i. podatkovne hiše ob jezeru ali kolišča[7] (angl. data lakehouses), katerih arhitektura je prikazana na sliki 4. V podatkovnih koliščih so prednosti podatkovnih jezer in skladišč združene znotraj formata oz. modela za shrambo, vendar je dodan sloj metapodatkov, ki vsebuje podrobne opise izvirnih podatkovnih zapisov v obliki objektov, shranjenih v podatkovnem jezeru. Na ta način se znotraj sloja metapodatkov lahko zagotovijo lastnosti ACID-a in dodatni indeksi za izboljšanje konsistentnosti podatkov in učinkovitosti sistema. V tem primeru uporabniki dostopajo do podatkov neposredno iz podatkovnega kolišča, pri čemer se zahtevani podatki najprej poiščejo z uporabo metapodatkov v po-

[7] Podatkovna hiša ob jezeru še nima ustaljenega ali uveljavljenega prevoda v slovenščino, zato uporablja predlog iz islovar.org, tj. kolišče.



Slika 3: **Prikaz dvostopenjske arhitekture za upravljanje velepodatkov.**

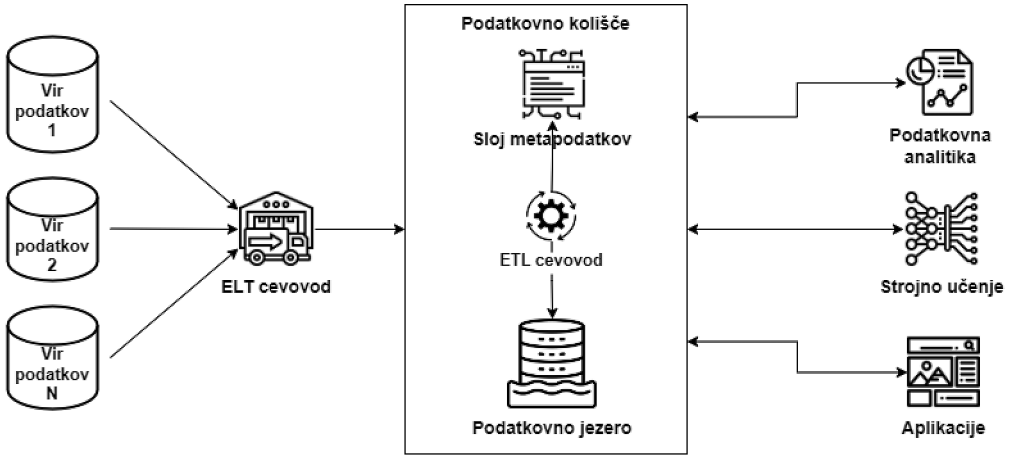
datkovnem jezeru, se transformirajo in/ali agregirajo glede na pravila, shranjena v sloju metapodatkov, se ponovno shranijo na drugo lokacijo v podatkovnem jezeru ter hkrati vrnejo kot rezultat poizvedbe uporabniku. Kot tehnologija so podatkovna kolišča precej nov koncept na trgu, tehnološke rešitve pa se (predvsem za sloj metapodatkov) še vedno razvijajo z vzorci načrtovanja za najboljšo implementacijo arhitekture. Trenutno najbolj uveljavljen primer platforme podatkovnega kolišča je Delta Lake.

Izbira najbolj primerne arhitekture za upravljanje velepodatkov je predvsem odvisna od stopnje strukturiranosti izvirnih podatkov, potreb po prilagodljivosti sheme, konsistentnosti podatkov in razširljivosti ter namenu uporabe shranjenih podatkov oz. ustreznem podatkovnem modelu za določeni namen uporabe (npr. dimenzijski model za analitiko, izvirni podatki za podatkovno znanost, model ključ-vrednost za hitro iskanje ipd.). Izbiro vedno bolj zapletejo novi vzorci načrtovanja in IT-arhitekture, ki želijo združiti prednosti obstoječih rešitev. V nadaljevanju se osredotočimo na analizo tehnoloških rešitev za implementacijo različic dvostopenjske arhitekture, ki je trenutno najbolj priljubljena.

4 ANALIZA IZBRANIH TEHNOLOŠKIH SKLADOV

4.1 Tehnološki sklad Google

Podjetje Google kot eno vodilnih ponudnikov oblačnih storitev v kontekstu upravljanja velepodatkov, ponuja v sklopu Google Cloud Platform precej širok nabor tehnoloških rešitev za implementacijo predhodno predstavljenih arhitektur. Temu v korist je tudi dejstvo, da je za implementacijo dvostopenjske arhitekture pri Googlu možno oblikovati nekaj različic tehnološkega sklada, glede na zahteve strank in željeno stopnjo odvisnosti od ponudnika oblačnih storitev. Ob analizi ponudbe tehnološkega sklada

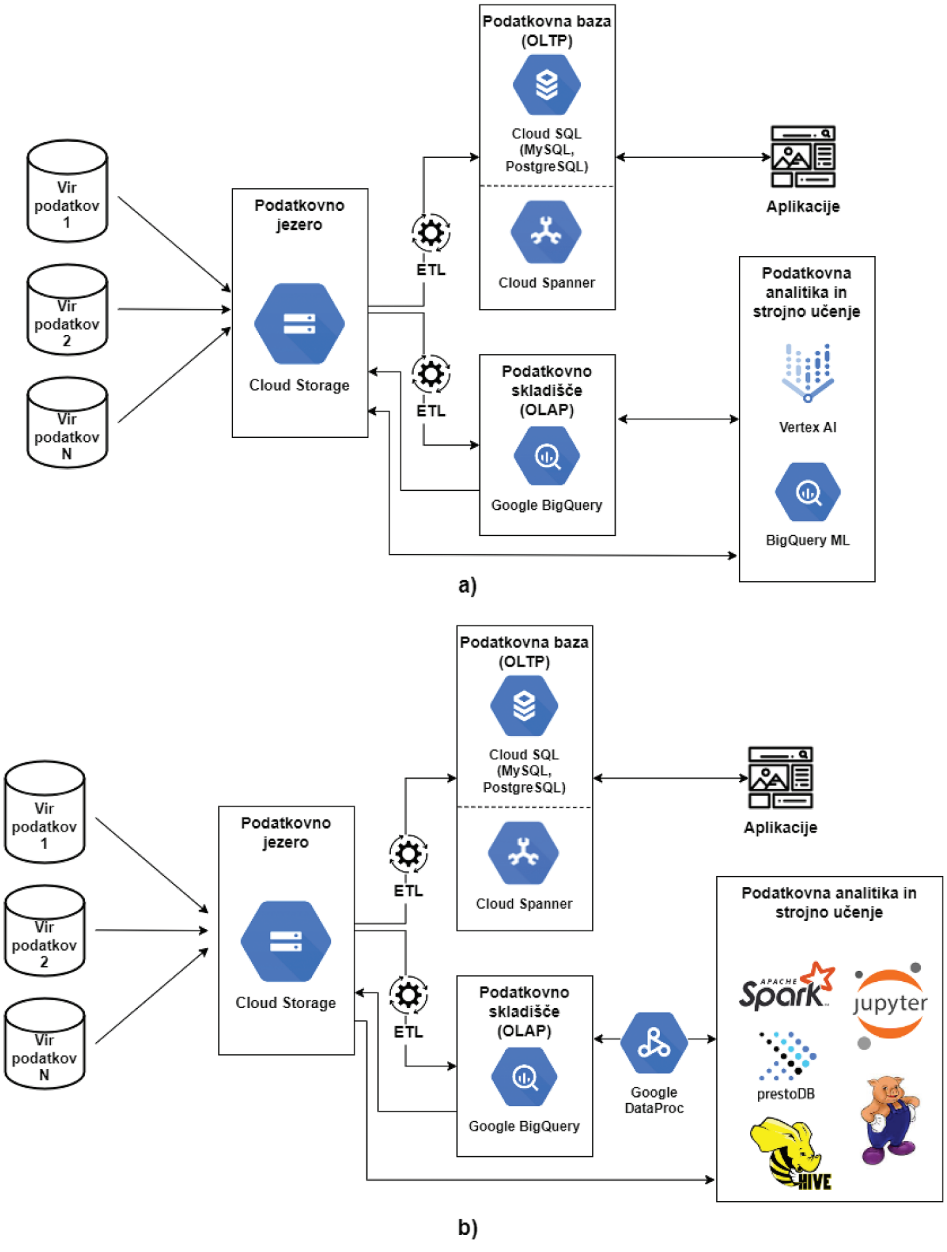


Slika 4: **Prikaz arhitekture z uporabo podatkovnega kolišča.**

Google, smo prišli do pregleda osnovnih rešitev za implementacijo dvostopenjske arhitekture, s katerimi lahko oblikujemo nekaj različic tehnološkega sklada, ki jih predstavljamo v nadaljevanju.

Na ravni podatkovnega jezera se različice tehnološkega sklada ne razlikujejo, saj se za ta namen priporoča uporaba oblačne rešitve za shranjevanje objektov Google Cloud Storage [26]. Cloud Storage je primeren za shranjevanje tudi nestrukturiranih podatkov oz. predstavlja začetno točko shranjevanja izvirnih podatkov. Kot tehnologija temelji na HDFS-u (angl. Hadoop Distributed File System), datotečnem sistemu platforme in ekosistema Hadoop, ki omogoča porazdeljeno shranjevanje velikih količin podatkov v datoteke. Cloud Storage omogoča neomejeno shranjevanje datotek velikosti do pet terabajtov v obliki objektov ter integracijo z različnimi rešitvami za zajem in obdelavo le-teh, pri čemer je optimiziran za prilagodljivo izvedbo povpraševanj in nizke stroške hrambe.

Na sliki 5 sta prikazani dve arhitekturi podatkovnega cevovoda, pri čemer se tehnološki sklad v ozadju razlikuje v izbiri rešitev za podatkovno analitiko



Slika 5: **Prikaz tehnološkega sklada Google z uporabo oblačne relacijske podatkovne baze in različnimi rešitvami za podatkovno analitiko in strojno učenje.**

in strojno učenje. Predstavljena arhitektura cevovoda omogoča podjetjem, da razširijo zmogljivost obstoječih relacijskih podatkovnih baz z vključevanjem podatkovnega skladišča, s čimer implementirajo IT-arhitekturo, ki je zmožna izpolniti analitične potrebe (OLAP) in potrebo po hitri izvedbi neposrednih povpraševanj po relacijski podatkovni bazi (OLTP). Treba je poudariti, da se kot osnova za podatkovno skladišče in analitično obdelavo (OLAP) uporablja širokostolpčna podatkovna baza BigQuery.

V takšni arhitekturi vlogo relacijske podatkovne baze prevzame Google Cloud SQL [25], ki je oblačna storitev za popolno upravljanje relacijske baze (trenutno so podprte MySQL, PostgreSQL in SQL Server). Storitev se lahko integrira v obstoječe IS-okolje podjetja s pomočjo storitev za orkestracijo, kot so App Engine, Google Kubernetes ipd., ter obstoječimi aplikacijami in storitvami, kot je Google BigQuery. Integracija z rešitvijo BigQuery omogoča vpeljavo analitičnih možnosti bodisi neposredno nad komponento BigQuery Storage bodisi nad podatkovno bazo Cloud SQL, kar lahko močno izboljša učinkovitost dostopa do podatkov.

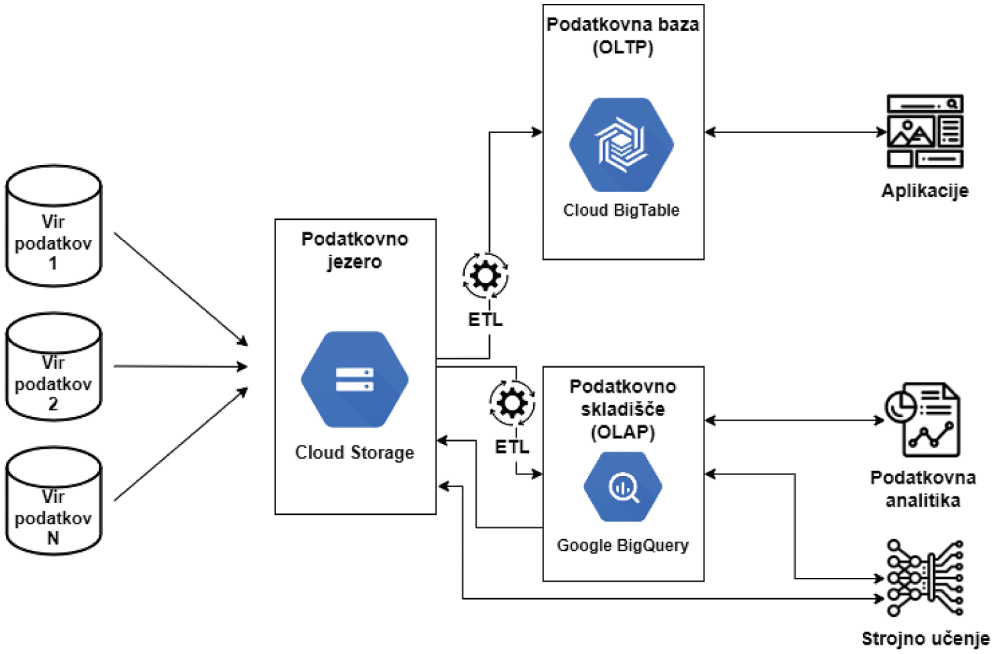
Če podjetje želi svoj transakcijski del (OLTP) učinkoviteje skalirati, se lahko uporabi rešitev Google Cloud Spanner za upravljanje relacijskih podatkovnih baz [24]. To je podatkovna baza, ki zagotavlja lastnosti ACID-a, a omogoča podprto horizontalno razširljivost, samodejno replikacijo podatkov, črepinjenje (angl. sharding) in obdelavo transakcij.

Privzeto je nastavljen Cloud Spanner, ki odvisno od količine podatkov in obremenitve sistema samodejno izvaja črepinjenje podatkov s ciljem izboljšanja učinkovitosti, vendar ni namenjen za uporabo v primerih, kjer je treba zadovoljiti osnovne potrebe SQL oz. OLTP (npr. osnovna analitika nad manjšo količino podatkov). Kot rešitev je Cloud Spanner uporabnejši v primerih, ko se pričakujeta masovna količina podatkov s pogostim pisanjem v podatkovno bazo (npr. tisoče operacij pisanja v sekundi) in visoka razširljivost sistema OLTP.

Po drugi strani se v kontekstu implementacije podatkovnega skladišča oz. sistema OLAP v tehnološkem skladu Google najbolj pogosto omenja Google BigQuery [20] – rešitev, ki je že od začetka predstavljala kritično točko prehoda v ekosistem velepodatkov in predstavlja navdih številnim sodobnim rešitvam za upravljanje velepodatkov. BigQuery je implementacija podatkovnega skladišča v več oblakih (angl. multicloud), kar omogoča analizo podatkov, shranjenih v več oblačnih storitvah. V osnovi to ni tipično podatkovno skladišče, ampak stolpčna shramba s strojem za izvedbo povpraševanj (angl. query engine), optimiziranim za analitične potrebe oz. hitre operacije branja. Predstavlja ustrezno rešitev poizvedbam, ki zahtevajo skeniranja celih tabel in izvedbo operacij grupiranja podatkov (npr. iskanje povprečja). Uporabnikom je rešitev BigQuery prijazna, saj tam lahko shranijo podatke, ki imajo strukturo relacijske tabele, dodatno pa vključuje nabor orodij za podatkovno analitiko, s katerim se že lahko neposredno ustvarijo nadzorne plošče (angl. dashboards) in generirajo poročila. Za namen analize podatkov BigQuery vključuje naslednji nabor orodij:

- BigQuery ML (angl. Machine Learning) – nabor orodij namenjen podatkovnim znanstvenikom in analitikom za izgradnjo in vzpostavitev modelov za strojno učenje z uporabo sintakse SQL (angl. Structured Query Language) na velikih količinah različno strukturiranih podatkov, shranjenih neposredno v BigQuery-u;
- BigQuery BI Engine (angl. Business Intelligence) – servis za podatkovno analitiko v pomnilniku (angl. in-memory), vgrajen v BigQuery, ki omogoča interaktivno analizo velikih in kompleksnih naborov podatkov, pri čemer se ohranjajo učinkovitost in hitrost izvedbe povpraševanj ter visoka konkurenčnost.

Google BigQuery je popularna rešitev s širokim naborom možnosti uporabe samo za shrambo in/ali obdelavo podatkov. Posledično je tudi model plačevanja storitve ločen na plačevanje rešitve izključno za namen učinkovitega shranjevanja podatkov in plačevanje za namen uporabe orodij za analitiko. BigQuery je priljubljen tudi zaradi podpore za združevanje zmogljivosti podatkovnega skladišča in podatkovnega jezera brez potrebe po uporabi tretje rešitve. Na ravni analize podatkov pa se pri uporabi rešitve BigQuery tehnološki sklad Google lahko oblikuje na različne načine. Kot primer sta na sliki 5 prikazani dve arhitekturi. Pri arhitekturi na sliki 5a se za podatkovno analitiko in strojno učenje uporabljajo orodja vgrajena v ekosistem Google (BigQuery), pri čemer uporabniki plačajo stroške shranjevanja in obdelave v rešitvi BigQuery. Kot alternativa za znižanje stroškov uporabe Google Cloud Platforme (BigQuery) se lahko vzpostavi tudi arhitektura, prikazana na sliki 5b, kjer se za analizo podatkov uporabljajo odprtokodne



Slika 6: **Prikaz tehnološkega sklada Google z uporabo oblačne nerela**cijske podatkovne baze.

rešitve, kot so Apache Spark, Presto, Hive, Pig. Na ta način uporabniki poravnajo le strošek shranjevanja podatkov v BigQueryju, za obdelavo podatkov pa se lahko v BigQuery integrirajo zgornje rešitve znotraj ekosistema Apache. Slednja integracija je možna z uporabo orodja Google DataProc [22], s katero te rešitve lahko pišejo ali berejo podatke neposredno v/iz baze BigQuery s pomočjo BigQuery Storage API-ja.

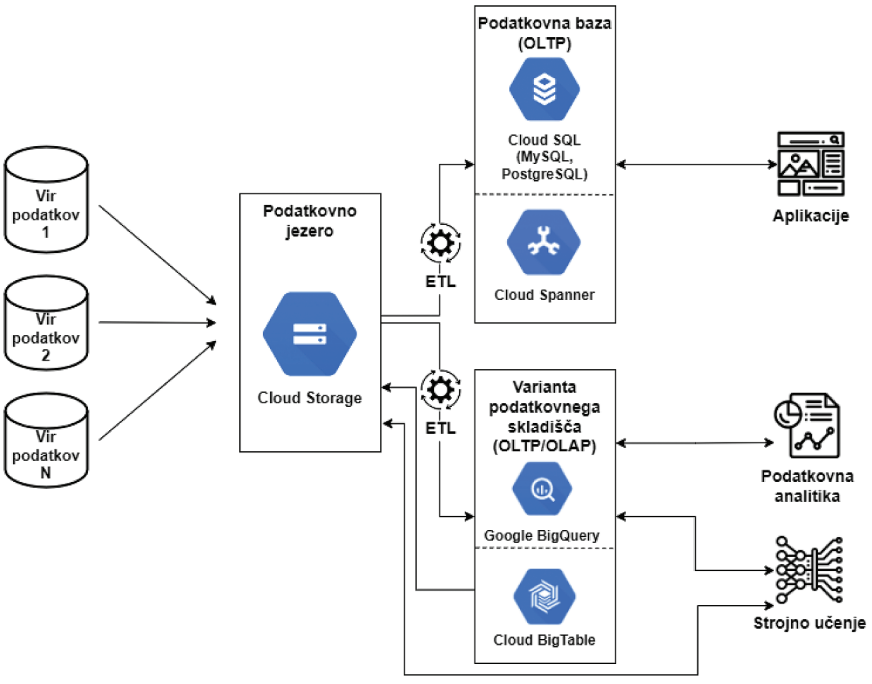
Če podjetje želi zamenjati obstoječo relacijsko bazo zaradi morebitnih izzivov in pomanjkljivosti, kot sta razširljivost in neprilagodljivost sheme, se podjetje lahko odloči tudi za uporabo nerelacijske baze za del OLTP, pri čemer nastane različica arhitekture, prikazana na sliki 6.

V tehnološkem skladu Google se za ta namen uporablja Google BigTable [21], ki je razširljiva storitev NoSQL za uspešno upravljanje in izvedbo velikih analitičnih in operativnih delovnih obremenitev (angl. workload). Podpira model sčasome konsistentnosti (angl. eventual consistency), kar pomeni, da se podatki v bazo shranijo enkrat in se samodejno replicirajo na vozlišča po potrebi. Kot baza BigTable podpira visoko prepustnost za operacije pisanja in branja z nizko zakasnitvijo in predstavlja idealen vir podatkov za posle MapReduce, saj je zaledni stroj za hrambo (angl. storage engine) načrtovan v smeri, ki je primeren za lažje strojno učenje in napredno analitiko.

Podatkovni model baze Google BigTable je kombinacija shranjevanja podatkov v obliki ključ-vrednost in delno širokostolpčne shrambe. Podatki se namreč shranjujejo v visoko razširljivih tabelah, vsaka tabela pa se predstavlja kot razvrščena mapa ključev in vrednosti. Zaradi slednjega je oblika shrambe BigTable primerna tudi za izvedbo nalog OLAP, saj omogoča hitro iteracijo po ključih. Po drugi strani se lahko pri oblikovanju uporabljajo tudi družine stolpcev (angl. column families), s čimer imajo uporabniki dostop do določenih prednosti širokostolpčnih shramb. Koncept družine stolpcev je pozneje razširjen v izjemno razširjenih široko stolpčnih bazah, kot sta Apache HBase ali Cassandra, ki so nastale na podlagi BigTabla.

Pri izvedbi povpraševanj BigTable ne podpira poizvedovalnega jezika SQL, vendar se lahko integrira z rešitvijo Google BigQuery, pri čemer nastane različica arhitekture, prikazana na sliki 7. Integracija BigTable in rešitev BigQuery predstavljata rešitev, ki lahko prevzame vlogo podatkovnega skladišča, saj zmore obvladati naloge OLTP in OLAP. V tem primeru nerelacijska baza, kot je BigTable, shranjuje transakcijske podatke v obliki, ki je vnaprej optimizirana za izvedbo povpraševanj, medtem ko močna rešitev, kot je BigQuery, lahko bere podatke in izvaja osnovne in napredne analize in agregacije.

Včasih se lahko zgodi, da uporaba relacijske podatkovne baze kot sistema OLTP ne zadostuje potrebam podjetja, saj imajo relacijske baze določene znane



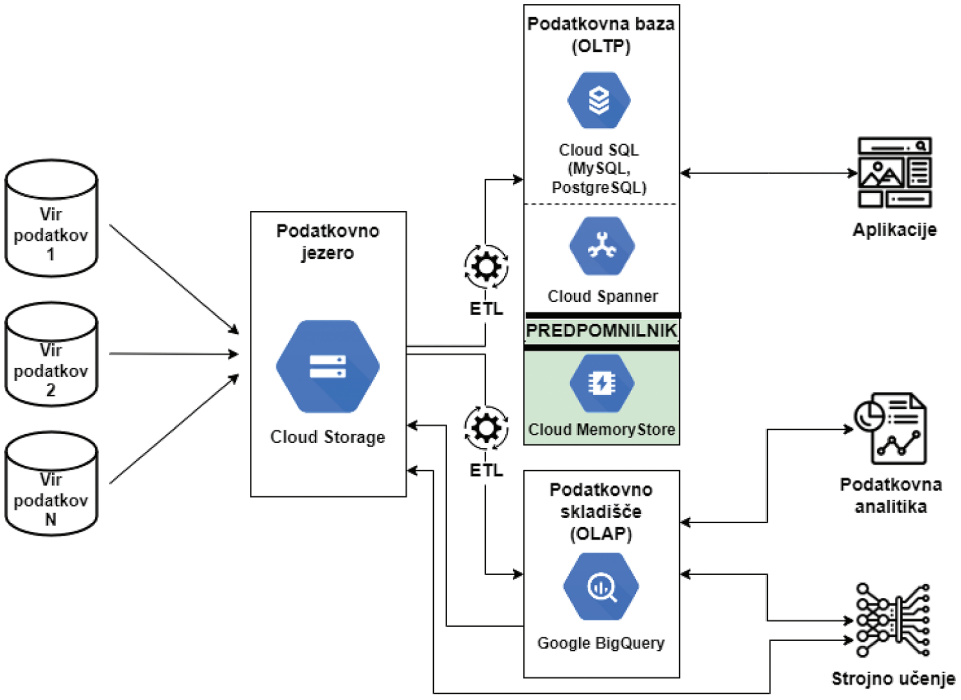
Slika 7: **Prikaz tehnološkega sklada Google z uporabo različice podatkovnega skladišča OLTP/OLAP.**

pomanjkljivosti v kontekstu velepodatkov (npr. razširljivost, prilagodljivost sheme). Ne glede na izbrano različico arhitektur, ki so predhodno predstavljene, se lahko zgodi, da podjetje ni zadovoljno z učinkovitostjo, ki jo v osnovi ponuja relacijska baza (tudi z ukrepi za optimizacije in izboljšave) ali pa je obremenitev sistema OLTP preprosto previsoka. V takšnem primeru se priporoča uporaba določene rešitve, ki temelji na dostopu do podatkov znotraj delovnega pomnilnika, saj se na ta način močno zmanjša število dostopov do diska, kar je posledica hitrejših poizvedb. Za ta namen Google ponuja uporabo storitve v delovnem pomnilniku Cloud Memorystore (slika 8). Rešitev je popolnoma kompatibilna z odprtokodnimi rešitvami, kot sta Redis in Memcached, ki se tudi uporabljajo za izboljšanje učinkovitosti sistema OLTP [23].

4.2 Tehnološki sklad Amazon

Tehnološki sklad Amazon vključuje ožji nabor rešitev, ki se sicer lahko uporabijo za več različic pri implementaciji dvostopenjske arhitekture. Kot je prikazano na sliki 9, vlogo podatkovnega jezera pri skladu Amazon prevzame Amazon S3 [15], dobro znana rešitev, ki se pogosto uporablja kot osnovna rešitev za shranjevanje podatkov v obliki datotek (angl. file server). S3 je največja in ena najbolj učinkovitih storitev za shranjevanje objektov za strukturirane in nestrukturirane podatke, vendar v novejšem času postaja tudi priljubljena rešitev za vzpostavitev podatkovnega jezera. Eden od razlogov za to je, da storitev samodejno ustvarja in shranjuje kopije vseh vnesenih objektov S3 v več porazdeljenih vozliščih. Kot razlog za uporabo rešitve S3 se poudarja tudi močna razširljivost na ravni petabajtov, ki omogoča navidezno neomejeno shranjevanje podatkov v kakršni koli obliki. Osnovni princip v ozadju je porazdelitev med hrambo in obdelavo podatkov, kar prinaša več prednosti predvsem pri vzdrževanju in uporabi virov. Glede na pogostost uporabe in dostopanja do različnih naborov podatkov, se uporabniki lahko odločijo za nakup različic storitev S3 Standard ali S3 One Zone Infrequent Access [18]. Prva je splošna različica za shranjevanje podatkov, ki jo pogosto uporabljamo pogosto za različne namene, medtem ko je različica Infrequent Access ustrezna izbira za shranjevanje dolgotrajnih podatkov, do katerih dostopamo redko.

Pri uporabi rešitve S3 kot podatkovnega jezera je treba vzpostaviti različna območja (angl. zones) [30, 12] oz. sloje za shranjevanje podatkov kot objektov glede na različne stopnje obdelave in korak v podatkovnem cevovodu. Tako imenovano območje landing predstavlja izhodiščno točko shranjevanja prispelih podatkov iz virov podatkov v izvirni obliki v cevovodu (npr. JSON, XML, CSV), preden se začnejo obde-



Slika 8: **Prikaz tehnološkega sklada Google z uporabo predpomnilnika.**

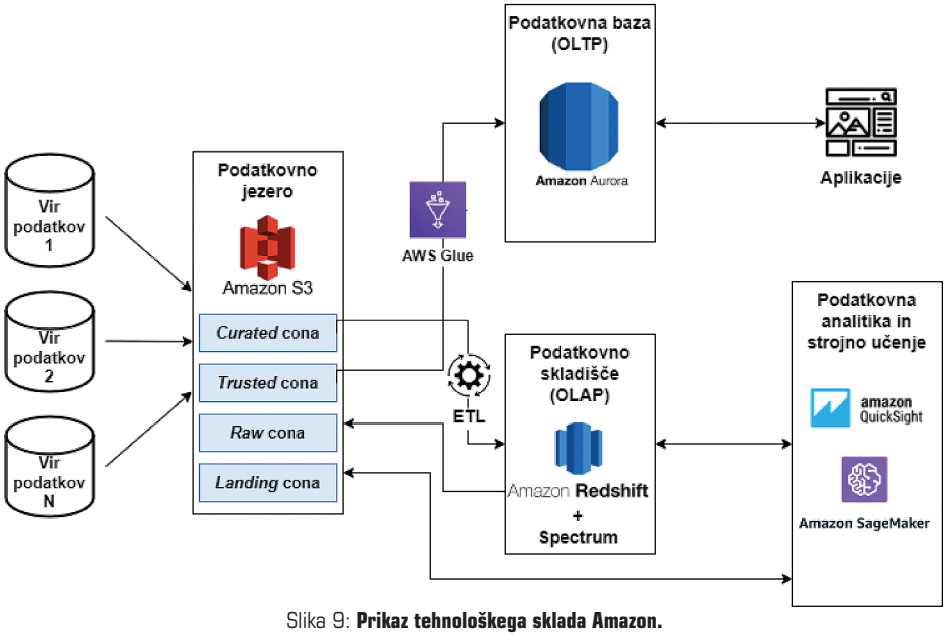
lovati na višjih ravneh. Podatki iz območja landing se validirajo in ustrezno reorganizirajo ter shranijo v t. i. območju raw, ki še vedno vsebuje podatke v izvirni obliki, vendar je struktura oz. organizacija objektov izboljšana glede na potrebe, podatki pa se shranijo v formatu, ki zagotavlja boljšo učinkovitost nadaljnjih analiz (npr. Parquet).

Izvirni podatki se nato transformirajo v skladu z določeno podatkovno shemo in hranijo v t. i. območju trusted. Do območja trusted lahko dostopa kateri koli sistem OLTP oz. podatkovna baza, saj so podatki konsistentni, prečiščeni in usklajeni z določeno shemo. Na najvišji ravni se podatki v območju trusted združujejo glede na potrebe analiz in se združeni hranijo v t. i. območju curated. Ta je tako glavni vir podatkov za podatkovno skladišče, kjer se podatki agregirajo za potrebe analitike in prikaza uporabniku.

Zaradi zahtevnejše organizacije podatkov znotraj shrambe S3 se priporoča vzpostavitev mehanizma za samodejni zajem podatkov iz virov, ustvarjanje in vzdrževanje kataloga metapodatkov ter zagotavljanje točnosti podatkovnih tokov od različnih območij. Za ta namen se priporoča uporaba rešitve AWS Glue [14], ki je popolnoma upravljana storitev za ETL, ki lahko olajša procese razvrščanja, čiščenja, transformacije in prenosa podatkov med različnimi lokacijami. AWS Glue Data Catalogue [17] je orodje znotraj rešitve AWS Glue, ki zagotavlja enotni repozitorij metapodatkov za izvedbo operacij za namen analitike nad več različnimi viri podatkov, kot so Amazon EMR, Athena, Redshift (Spectrum) ali katere koli aplikacije kompatibilne s hrambo Hive za metapodatke (angl. Hive metastore). Data Catalogue je predvsem podatkovna baza, v katero se shranjujejo metapodatki in tabele podatkovnih shem, lokacija podatkov v sistemu in različne metrike za delovanje rešitve oz. sistema. Ker je kompatibilen s hrambo Hive za metapodatke, pa se Data Catalogue lahko uporablja kot osrednji repozitorij za shranjevanje strukturiranih in nestrukturiranih metapodatkov.

Ko so podatki shranjeni v S3 se v naslednjem koraku cevovoda transformirajo skozi procese ETL, ki postavijo le tiste v ustrezno obliko za potrebe analitike ali vizualizacije. Pri transformaciji podatkov se lahko uporabijo trije pristopi [14]:

- Ustvarjanje gruče Amazon EMR z nameščeno rešitvijo Hive – surovi podatki, shranjeni v S3, se transformirajo v tabele Hive in se shranijo v S3 v formatu Parquet.
- Uporabljanje rešitve Spark na Amazon EMR.
- Uporabljanje rešitve AWS Glue, ki samodejno najde surove podatke, shranjene v S3, identificira njihov format shrambe ter predlaga ciljno shemo in transformacije.



Slika 9: **Prikaz tehnološkega sklada Amazon.**

V tehnološkem skladu Amazon se kot sistem OLTP oz. transakcijska podatkovna baza uporablja Amazon Aurora [13], storitev za implementacijo relacijske baze, ki združuje hitrost in dostopnost komercialnih podatkovnih baz s preprostostjo in stroškovno učinkovitostjo odprtokodnih podatkovnih baz. Aurora je popolnoma kompatibilna s sistemi za upravljanje podatkovnih baz MySQL in PostgreSQL, kar bistveno olajša integracijo z obstoječimi aplikacijami, ne da bi bile nujne velike spremembe. Prilagodljiva je vsem potrebam po razširljivosti, saj se viri shrambo po potrebi samodejno razširijo. Poslovni model vzpostavitve rešitve Aurora vključuje plačilo storitve po uri uporabe posamezne instance Aurore.

Na drugi strani sklada se kot glavna rešitev za OLAP uporablja Amazon RedShift [16] – hitro podatkovno skladišče na ravni petabajtov, s katerim lahko uporabniki analizirajo svoje podatke, shranjene na več različnih lokacijah. Na ravni implementacije je RedShift širokostolpčna podatkovna baza, ki jo je z uporabo širokega nabora vtičnikov možno povezati z različnimi odjemalci oz. bazami, kot je PostgreSQL. Rešitev je preprosto razširljiva, saj je po potrebi možno dodati nova vozlišča v oblačno storitev s pomočjo spletne konzole ali zasebnega API-ja. Vozlišča lahko dosežejo kapaciteto med 160 gigabajtov in 16 terabajtov. Uporabniki poravnajo dejansko porabo virov. RedShift predstavlja izjemno močno rešitev znotraj sklada Amazon, s katero lahko podjetja zadovoljijo analitične potrebe, dodatna prednost pa je tudi možnost povezovanja RedShifta z relacijsko podatkovno bazo ali rešitvijo za poslovno obveščanje zunaj ekosistema Amazon. Za delo z RedShiftom se uporablja robusten API, ki omogoča izvedbo poizvedb nad podatki shranjeni v bazi. Za dodatno optimizacijo uporabe virov ob izvedbi povpraševanj RedShift uporablja algoritme strojnega učenja za napovedovanje in analizo povpraševanj. RedShift podpira različne formate podatkov v obliki datotek, kot so Parquet ali ORC (angl. Optimized Row Columnar), ki lahko vplivajo na učinkovitost sistema. Kot omejitev uporabe RedShifta se omenjata OLAP, omejitve, povezane z učinkovitostjo operacij vnašanja, posodabljanja in brisanja ter izostanek mehanizma za upravljanje indeksov znotraj platforme AWS.

Znotraj rešitve RedShift obstaja tudi orodje, ki omogoča hitro izvedbo kompleksnih analiz nad objekti, shranjenih v določenih oblačnih rešitev sklada Amazon (S3, RedShift itn.). Amazon RedShift Spectrum [19] je orodje, ki se pogosto uporablja z RedShiftom, saj omogoča samodejno skaliranje in optimalno izvedbo povpraševanj (na ravni eksabajtov podatkov) nad gručo RedShift. Uporaba orodja se poravna tudi po porabi, nujno pa potrebuje vzpostavljeno RedShift gručo in povezanega odjemalca SQL. Z uporabo znane sintakse jezika SQL znotraj orodja Spectrum lahko uporabniki hitro dostopajo do velikih količin podatkov porazdeljenih med več gruč RedShift (ali vozlišč) in izvajajo kompleksna povpraševanja nad podatki. Rezultati povpraševanj se lahko

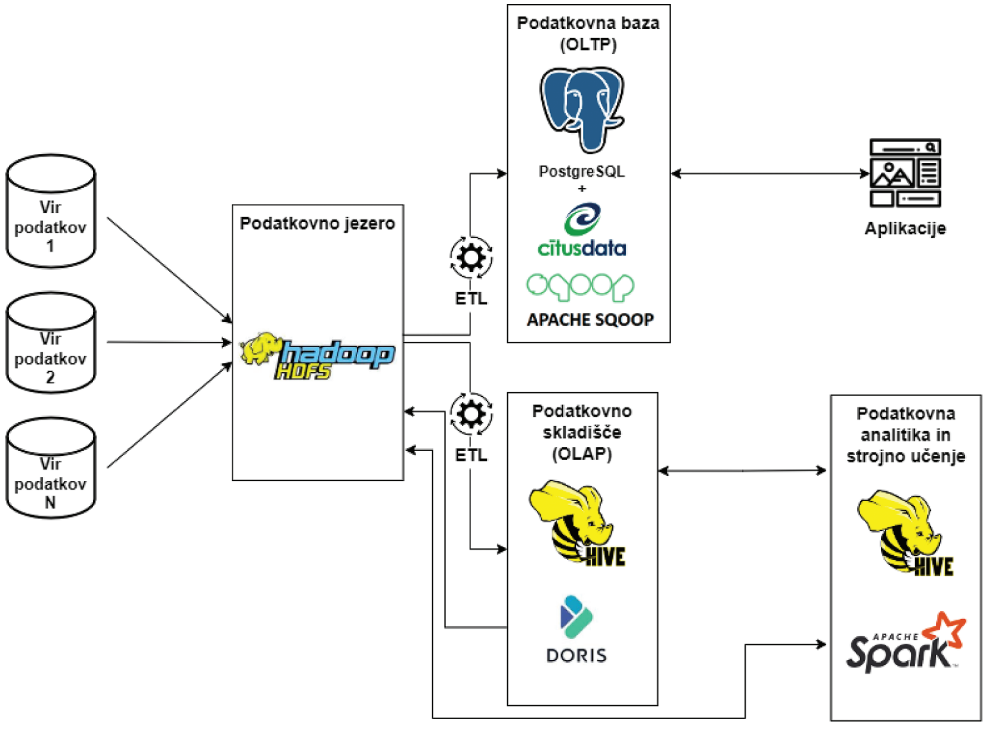
nato uporabijo za namen podatkovne analitike, ki jo znotraj sklada Amazon izvajamo z orodjem Amazon QuickSight, ali strojnega učenja, za katero potrebujemo rešitev Amazon SageMaker.

4.3 Odprtokodni tehnološki sklad Apache

Kot alternativa plačljivim storitvam in rešitvam, ki jih ponujajo vodeči ponudniki na trgu, kot so Google (Google Cloud Platform – GCP), Amazon (AWS) in Microsoft (Azure), se podjetja lahko obrnejo tudi proti odprtokodnim tehnologijam, kar je tudi prednostna smer implementacije pri večini podjetij z omejenim proračunom za implementacijo sistema. Pri tem se kot sopomenka za odprtokodne rešitve na področju velepodatkov večinoma uporablja Apache, saj predstavlja nabor programske opreme, ki je na voljo vsem uporabnikom. Rešitve, ki jih ponuja sklad Apache, so dovolj hitre in zanesljive za uporabo ter se lahko prilagodijo posameznim potrebam podjetja, kar je največja dodana vrednost za podjetja, ki imajo specifične poslovne procese ali zahteve in želijo imeti večji nadzor nad implementacijo. Čeprav implementacija sklada Apache zahteva več časa in znanja, brezplačna uporaba rešitev za upravljanje velepodatkov, ki niso t. i. 'črna škatla' (angl. black box), podjetju predstavlja upravičen strošek za uporabo prav tistega sklada namesto plačevanja oblačnih storitev pri predhodno omenjenih ponudnikih.

Čeprav je nabor možnih rešitev za posamezo komponento dvostopenjske arhitekture pri skladu Apache precej širši kot pri skladih Google in Amazon, je v nadaljevanju na sliki 10 predstavljen osnovni predlog sklada z rešitvami, ki predstavljajo jedro tehnoloških skladov za upravljanje velepodatkov.

Za funkcionalnosti podatkovnega jezera pri skladu Apache poskrbi rešitev HDFS (angl. Hadoop Distributed File System) [8], porazdeljeni datotečni sistem, ki predstavlja osnovno lokacijo za shranjevanje podatkov v obliki datotek znotraj platforme in ekosistema Hadoop. Podatki se znotraj HDFSa shranujejo kot bloki, pri čemer se vsak blok za namen replikacije razdeli trikrat na različna HDFS podatkovna vozlišča. HDFS predstavlja hrbtenico ekosistema Hadoop in podobnih datotečnih sistemov drugih ponudnikov na trgu, kot so Microsoft HDInsight, Cloudera CDH, IBM Open Platform itn. Kot alternativa uporabi HDFS-a kot podatkovnega jezera so dandanes na trgu dostopne tudi rešitve zunaj sklada Apache, in sicer se lahko uporabi npr. MinIO kot oblačna rešitev visoke zmogljivosti, ki je hkrati kompatibilna z rešitvijo Amazon S3. HDFS deluje po principu, da vsaka naprava v gruči Hadoop hrani podmnožico podatkov,



Slika 10: **Prikaz tehnološkega sklada Apache.**

ki tvorijo datotečni sistem. Ob potrebi po shranjevanju več podatkov se lahko v gručo doda več naprav z več diski in se celotni datotečni sistem preprosto razširi. Podatki, shranjeni v HDFSu, se nato posredujejo v nabor poslov ETL, ki jih lahko opravimo z Apache Sparkom ali katerim koli orodjem, ki omogoča uvoz in izvoz podatkov v/iz HDFSa. Primer takšnega orodja je Apache Sqoop [5], ki omogoča prenos paketnih podatkov med gručo Hadoop in strukturiranimi viri podatkov, kot so relacijske baze ali skladišča.

Z uporabo Sqoopa se podatki iz HDFSa lahko transformirajo in shranijo v relacijsko bazo, kot je PostgreSQL, ki nato prevzame vlogo sistema OLTP v celotni arhitekturi. Za izboljšano učinkovitost in razširljivost klasične baze PostgreSQL se uporablja odprtokodna razširitev Citus [4], ki vpeljuje možnost porazdeljenega shranjevanja in obdelave podatkov v bazi PostgreSQL in učinkovito razširi zmogljivost obstoječe podatkovne baze PostgreSQL. Z vpeljavo rešitve Citus lahko podjetja razširijo obstoječo podatkovno bazo PostgreSQL na gručo vozlišč PostgreSQL, ki so zmožna učinkovito hraniti več podatkov. Nastane gruča Citus, v kateri eno vozlišče predstavlja koordinatorja Citus, ki posreduje povpraševanja, napisana v jeziku SQL, na ustrezno vozlišče PostgreSQL in vodi evidenco o lokaciji, kjer je določeni podatek v gruči.

Vlogo podatkovnega skladišča v skladu Apache lahko prevzame rešitev Apache Doris [6], podatkovno skladišče OLAP, ki omogoča uporabo poizvedovalnega jezika SQL in temelji na principu masovne vzporedne obdelave (angl. massively parallel processing). To je relativno nova rešitev, razvita v inkubatorju Apache, ki je nastala kot rezultat integracije rešitev Apache Impala (porazdeljeni stroj SQL za povpraševanja nad gručo Hadoop) in Google Mesa (visoko razširljivo podatkovno skladišče, razvito s strani Googla). Doris omogoča preprosti načrt IT--arhitekture, ki hkrati zagotavlja visoko zanesljivost, dostopnost, razširljivost ter toleranco na napake. V ozadju rešitve Doris je širokostolpčna podatkovna baza, vendar je le-tista nadgrajena s tehnologijo vektorizacije (angl. vectorization technology) zaradi optimizacije pri izvedbi povpraševanj. Tradicionalni stroji SQL za povpraševanja (angl. SQL query engines) analizirajo podatke v tabelah po principu vrstic (angl. row- based), kar prinaša dodatni strošek pri uporabi procesorskih enot CPU. Pri Doris želi tehnologija vektorizacije izboljšati pristop na način, da se podatki obdelajo po principu stolpcev, kar zmanjšuje uporabo virov CPU in v celoti eliminira odvečno število operacij branja nad podatki. Doris podpira tudi visoko konkurenčnost dostopa med več uporabniki ter horizontalno razširljivost. Uporabniki lahko dostopajo do podatkov, shranjenih v skladišču, z različnimi odjemalci in orodji za poslovno obveščanje. Kot največjo prednost rešitve poudarjajo možnost realnočasovnih nadzornih plošč, ad hoc poizvedbe in celovito rešitev za razširljivo podatkovno skladišče, ki lahko zagotovi funkcionalnosti, ki bi jih sicer mogli nadomeščati z več rešitvami (Spark, Hive in HBase). Kot alternativa Apache Doris je vsekakor tudi Apache HBase, ki je ena pravih odprtokodnih rešitev širokostolpčne podatkovne baze.

Kot pomembna komponenta sklada Apache se lahko kljub temu uporablja Hive [7] – podprt sistem SQL za analiziranje podatkov, shranjenih v HDFSu. Zaradi preprostega jezika za pisanje poizvedb (Hive Query Language, HQL) je Hive ustrezna rešitev za obdelavo podatkov in izvedbo procesa ETL, zaradi neposrednega dostopa do podatkov v HDFSu pa se lahko uporablja tudi kot alternativa za podatkovno skladišče. V primerjavi z drugimi rešitvami sklada Apache (npr. Impala ali Doris) je bolj namenjen podatkovnim inženirjem in ne podatkovnim znanstvenikom. V tistem primeru se podatki, shranjeni v podatkovnem jezeru oz. HDFSu, indeksirajo v komponenti Hive MetaStore, ki deluje kot katalog metapodatkov, v katerem se struktura datotek HDFS mapira v obliko razpredelnice. Hive je zgrajen nad Hadoopom, kar pomeni, da je že v načrtu ustrezna rešitev za potrebe upravljanja velikih datotek na ravni petabajtov. Kot rešitev omogoča sloj abstrakcije nad HDFSom, na katerem so podatki predstavljeni kot tabele z vrsticami, stolpci in podatkovnimi tipi, katere uporabniki lahko analizirajo po vmesniku HiveQL. Kot dodatna prednost je podpora za transakcije ACID, kar pomaga pri zagotavljanju konsistentnosti podatkov. Hive sledi pristopu sheme ob branju (angl. schema-on- read), kar pomeni, da se podatki hitro shranjujejo v skladišče Hive brez validacije ali dodatnih preverjanj, končno obliko pa pridobijo iz Hive MetaStora ob izvedbi povpraševanj. Glede integracije z drugimi rešitvami, kot je Amazon S3, je Hive dovolj odprta rešitev. Razen shranjevanja združenih podatkov v skladišču se Hive lahko uporablja tudi na ravni podatkovne analitike v primeru osnovnih analiz. Za naprednejšo analitiko in strojno učenje se priporoča uporaba rešitve Apache Spark, ki predstavlja danes najbolj razširjeno rešitev za obdelavo ve-

lepodatkov [31] zgrajeno nad porazdeljenim strojem SQL za izvedbo povpraševanj nad velepodatki.

5 DISKUSIJA

IT-arhitekti sodobnih podatkovnih cevovodov imajo izziv izbrati tehnološki sklad za implementacijo dvostopenjske arhitekture, ki je v skladu z dostopnimi viri ter omejitvami in zahtevami vodstva podjetja. Čeprav ponudniki, kot sta Google ali Amazon, težijo k plačljivemu načinu uporabe njihovih rešitev, je možno določene rešitve znotraj njihovega sklada integrirati z odprtokodnimi rešitvami znotraj sklada Apache ali drugega sklada, s katerima si podjetja lahko znižajo stroške implementacije. V Tabeli 1 so predstavljeni izsledki analize rešitev tehnoloških skladov Google, Amazon in Apache, ki so arhitektom na voljo za implementacijo posameznih komponent dvostopenjske arhitekture, in sicer možne rešitve za podatkovno jezero, podatkovno skladišče (OLAP), podatkovno bazo (OLTP) ter rešitve za podatkovno analitiko in strojno učenje. Poudariti pa je treba, da IT-arhitekti niso zgolj omejeni z uporabo odprtokodnih rešitev za podatkovno analitiko, kot prikazuje slika 5b, temveč lahko posamezne komponente (npr. podatkovno jezero, podatkovno skladišče, podatkovna baza) izbirajo s strani različnih ponudnikov in te med seboj združujejo. Primer tega bi bila uporaba Amazon S3 za podatkovno jezero, relacijsko podatkovno bazo PostgreSQL na lastni infrastrukturi za namen OLTP, Google BigQuery kot osnova za podatkovno skladišče oz. OLAP ter odprtokodna orodja za podatkovno analitiko, kot so Apache Spark, Hive, Pig itn. Kombinacij je toliko, koliko je orodij in ponujenih storitev na trgu, pri čemer načeloma omejitev ni, temveč so zgolj morebitni zadržki zaradi zapletenosti upravljanja takšne IT-arhitekture. Prednosti takšnih morebitnih kombinacij so vsekakor manjši stroški ter nezavezanost enemu ponudniku storitev. Prav tako pa bi lahko IT-arhitekti uporabili določene ponudnike oblačnih storitev zgolj za infrastrukturo kot storitev (IaaS), med tem ko bi sami nameščali in upravljali odprtokodne rešitve za del podatkovnega inženirstva ali za podatkovno analitiko.

Eden večjih izzivov pri načrtovanju arhitekture sodobnega cevovoda je predvsem izbira najbolj ustrezne rešitve za hitro in učinkovito shranjevanje podatkov ter rešitve za preprosto in morebitno kompleksno obdelavo in analizo le-teh. Zaradi tega se podjetja pogosto odločajo za izbiro oblačne storitve, ki predstavlja sistem OLTP (relacijski ali nerelacijski), saj se na ta način izognejo stroškom razširjanja in vzdrževanja obstoječih sistemov OLTP, lahko pa preprosto nadgradijo svoje obstoječe baze oz. najdejo kompatibilno oblačno rešitev kot nadgradnjo obstoječih baz (npr. Google Cloud SQL podpira implementacijo klasične baze MySQL). Na ravni podatkovnega jezera zadostuje hramba podatkov v obliki objektov ali datotek, kar veliko podjetij že uporablja zaradi shranjevanja vseh podatkov v sistemih. To pomeni, da je za podjetja dovolj, da so podatki v obliki objektov ali datotek ustrezno organizirani glede na potrebe prihodnjih analiz in njihove uporabe v naslednjem koraku cevovoda. Glede na to, da večina rešitev za podatkovna jezera v oblaku temelji na odprtokodnem HDFSu, je odločitev o uporabi rešitve za podatkovno jezero predvsem odvisna od možnosti samostojnega vzdrževanja rešitve. Podjetja z večjim proračunom se ponavadi odločajo za uporabo zanesljive oblačne rešitve, kot je Amazon S3 ali Google Object Storage, pri drugih komponentah cevovoda pa poskusijo biti čim bolj neodvisni od plačljivega ponudnika in uporabiti nadgrajeno različico komponente, ki jo že uporabljajo (npr. transakcijska podatkovna baza ali skladišče, ročno razviti proces ETL, brezplačna orodja za vizualizacijo ali podatkovno analitiko).

6 SKLEP

V članku smo predstavili teoretično ozadje načrtovanja sodobnih IT-arhitektur za upravljanje velepodatkov ter osnovne komponente takšnih sistemov. Področje upravljanja velepodatkov se tehnološko intenzivno razvija v zadnjem desetletju. Kot rezultat so nastale številne tehnologije različnih ponudnikov na trgu, kar predstavlja velik izziv pri izbiri tehnološkega sklada za podjetja, ki želijo izboljšati zmogljivost svojih obstoječih sistemov na ravni velepodatkov. Čez leta so se uporabljale različne rešitve za shranjevanje in obdelavo velikih količin podatkov za namene OLTP ali OLAP, kot so transakcije relacijske in nerelacijske podatkovne baze, podatkovna skladišča, podatkovna jezera in (po novem) podatkovna kolišča. Slednje se vključujejo kot komponente za shranjevanje podatkov v sodobnih podatkovnih cevovodih, ki zagotavljajo pravilne podatkovne toke, od podatkovnih virov do ciljne aplikacije ali orodij za podatkovno analitiko. Točnost in učinkovitost upravljanja podatkov v tistih cevovodih se zagotovita z ustreznim podatkovnim inženirstvom in IT-arhitekturo, ki

Tabela 1: **Pregled tehnoloških rešitev za implementacijo dvostopenjske arhitekture znotraj skladov Google, Amazon in Apache.**

	Sklad Google	**Sklad Amazon**	**Sklad Apache**
Podatkovno jezero	Google Cloud Storage	Amazon S3	Hadoop HDFS
Podatkovno skladišče (OLAP)	Google BigQuery	Amazon RedShift + RedShift Spectrum	- Apache Hive - Apache Doris
Podatkovna baza (OLTP)	- Google Cloud SQL ali Cloud Spanner (relacijska PB) - Google BigTable (nerelacijska PB) - Google Cloud MemoryStore (predpomnilnik)	Amazon Aurora	PostgreSQL + Citus
Podatkovna analitika in strojno učenje*	- Vertex AI in BigQuery ML - Apache Spark / Presto / Hive / Pig (vključuje uporabo rešitve Google DataProc)	- Amazon QuickSight - Amazon SageMaker	- Apache Hive - Apache Spark

vključuje eno ali več rešitev za zajem, shranjevanje, obdelavo ter dostop do podatkov. Že več let je najbolj priljubljena dvostopenjska arhitektura, saj hkrati zagotavlja hitrost poizvedb po transakcijskih podatkih, shranjenih v sistemih OLTP (oblačne ali on premise podatkovne baze), in učinkovitost kompleksnih poizvedb nad združenimi podatki, shranjenih v sistemih OLAP (podatkovna skladišča ali prilagojene širokostolpčne baze).

Kot je razvidno iz narejene analize tehnoloških skladov Google in Amazon, velike razlike med tema plačljivima ponudnikoma ni, saj sta arhitektura in način delovanja podatkovnega cevovoda v obeh primerih podobni in so uporabnikom na voljo močne in zelo razširljive rešitve. Prav tako temeljijo storitve, ki jih ponujajo na skoraj enakih osnovah ter se zgolj promovirajo z ločenimi blagovnimi znamkami in dodatnimi funkcionalnostmi. Z odprtokodnim skladom imajo Apache uporabniki večji nadzor nad posameznimi komponentami arhitekture, vendar to zahteva neprekinjeno nadzorovanje sistema zaradi pravočasnega ukrepanja za odpravljanje napak in skaliranje rešitev. Izbira tehnološkega sklada je torej odvisna od več dejavnikov, med katerimi sta najpomembnejši dostopnost človeških in finančnih virov v podjetju ter želena prilagodljivost in stopnja nadzora nad celotnim sistemom.

V tem članku smo se predvsem omejili na podatkovno inženirstvo in zagotavljanje IT-arhitekture za učinkovito uporabo in upravljanje velepodatkov, pri čemer pa smo izpustili določene izzive sodobnega podatkovnega inženirstva, kot so zahteve po realnočasovni obdelavi dogodkovnih podatkov itn. Prav tako smo analizirali in predstavili zgolj dva od treh velikih ponudnikov oblačnih storitev (tj. Google in Amazon). V prihodnosti bomo zajeli tudi analizo arhitektur Lambda in Kappa ter trenutno analizo razširili tudi na Microsoftov sklad in druge ponudnike, kot tudi predstaviti morebitne IT-arhitekture, ki kombinirajo uporabo komponent različnih ponudnikov. Prav tako se bomo osredotočili na analizo učinkovitosti in smotrnosti uporabe zgolj infrastrukture kot storitve (angl. Infrastructure-as-a-Service, IaaS) ter na osnovi tega snovanja IT-arhitekture, ki temelji na skladu Apache.

ZAHVALA

Rezulati so delno financirani s strani raziskovalnega programa št. P2-0057 Javne agencije za raziskovalno dejavnost Republike Slovenije iz državnega proračuna ter projektov ZeRoW (1001036388) in Data4Food2030 (101059473) financiranih iz okvirnega programa EU za raziskave in inovacije – Obzorje H2020 in Obzorje Evropa.

LITERATURA

[1] Memoona J Anwar, Asif Q Gill, Farookh K Hussain, and Muhammad Imran. Secure big data ecosystem architecture: challenges and solutions. EURASIP Journal on Wireless Communications and Networking, 2021(1):1–30, 2021.

[2] Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of CIDR, 2021.

[3] Abhay Kumar Bhadani and Dhanya Jothimani. Big data: challenges, opportunities, and realities.

[4] Effective big data management and opportunities for implementation, pages 1–24, 2016.

[5] Citus Data. Citus dokumentacija. [Na spletu]. Dostopno: https://docs.citusdata.com/en/v11.1/get_started/what_is_citus.html. [Dostopano: 27-okt-2022], 2022.

[6] The Apache Software Foundation. Apache Sqoop dokumentacija. [Na spletu]. Dostopno: https://sqoop.apache.org/. [Dostopano: 27-okt-2022], 2019.

[7] The Apache Software Foundation. Apache Doris dokumentacija. [Na spletu]. Dostopno: https://doris.apache.org/. [Dostopano: 27-okt-2022], 2022.
[8] The Apache Software Foundation. Apache Hive dokumentacija. [Na spletu]. Dostopno: https://hive.apache.org/. [Dostopano: 27-okt-2022], 2022.
[9] The Apache Software Foundation. Hadoop HDFS Architecture Guide. [Na spletu]. Dostopno: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. [Dostopano: 27-okt-2022], 2022.
[10] Paramita Ghosh. Data architecture challenges. [Na spletu]. Dostopno: https://www.dataversity.net/data- architecture--challenges. [Dostopano: 4-okt-2022], junij 2022.
[11] Josh Howarth. 30+ incredible big data statistics (2022). [Na spletu]. Dostopno: https://explodingtopics.com/blog/big-data-stats. [Dostopano: 3-okt-2022], avgust 2022.
[12] Oliver Hummel, Holger Eichelberger, Andreas Giloj, Dominik Werle, and Klaus Schmid. A collection of software engineering challenges for big data system development. In 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 362–369. IEEE, 2018.
[13] Amazon Web Services Inc. AWS Serverless Data Analytics Pipeline. [Na spletu]. Dostopno: https://docs.aws.amazon.com/pdfs/whitepapers/latest/aws-serverless-data-analytics--pipeline/aws- serverless-data-analytics-pipeline.pdflogical--architecture-of-modern-data-lake-centric-analytics- platforms. [Dostopano: 27-okt-2022], april.
[14] Amazon Web Services Inc. Amazon Aurora dokumentacija. [Na spletu]. Dostopno: https://aws.amazon.com/rds/aurora/. [Dostopano: 27-okt-2022], 2022.
[15] Amazon Web Services Inc. Amazon AWS Glue dokumentacija. [Na spletu]. Dostopno: https://aws.amazon.com/glue/. [Dostopano: 27-okt-2022], 2022.
[16] Amazon Web Services Inc. Amazon AWS S3 dokumentacija. [Na spletu]. Dostopno: https://aws.amazon.com/s3/. [Dostopano: 27-okt-2022], 2022.
[17] Amazon Web Services Inc. Amazon RedShift dokumentacija. [Na spletu]. Dostopno: https://aws.amazon.com/redshift/. [Dostopano: 27-okt-2022], 2022.
[18] Amazon Web Services Inc. AWS Glue Components. [Na spletu]. Dostopno: https://docs.aws.amazon.com/glue/latest/dg/components-overview.html. [Dostopano: 27-okt-2022], 2022.
[19] Amazon Web Services Inc. Central storage: Amazon S3 as the data lake storage platform. [Na spletu]. Dostopno: https://docs.aws.amazon.com/whitepapers/latest/building-data-lakes/amazon-s3-data-lakestorage-platform.html. [Dostopano: 27-okt-2022], 2022.
[20] Amazon Web Services Inc. Getting started with Amazon Redshift Spectrum. [Na spletu]. Dostopno: https://docs.aws.amazon.com/redshift/latest/dg/c-getting-started-using--spectrum.html. [Dostopano: 27-okt-2022], 2022.
[21] Google Inc. Google Cloud BigQuery dokumentacija. [Na spletu]. Dostopno: https://cloud.google.com/bigquery. [Dostopano: 27-okt-2022], 2022.
[22] Google Inc. Google Cloud BigTable dokumentacija. [Na spletu]. Dostopno: https://cloud.google.com/bigtable. [Dostopano: 27-okt-2022], 2022.
[23] Google Inc. Google Cloud DataProc dokumentacija. [Na spletu]. Dostopno: https://cloud.google.com/dataproc. [Dostopano: 27-okt-2022], 2022.
[24] Google Inc. Google Cloud MemoryStore dokumentacija. [Na spletu]. Dostopno: https://cloud.google.com/memorystore. [Dostopano: 27-okt-2022], 2022.
[25] Google Inc. Google Cloud Spanner dokumentacija. [Na spletu]. Dostopno: https://cloud.google.com/spanner. [Dostopano: 27-okt-2022], 2022.
[26] Google Inc. Google Cloud SQL dokumentacija. [Na spletu]. Dostopno: https://cloud.google.com/sql. [Dostopano: 27-okt-2022], 2022.
[27] Google Inc. Google Cloud Storage dokumentacija. [Na spletu]. Dostopno: https://cloud.google.com/storage. [Dostopano: 27-okt-2022], 2022.
[28] Godson Koffi Kalipe and Rajat Kumar Behera. Big data architectures: A detailed and application oriented review. International Journal of Innovative Technology and Exploring Engineering, 8:2182–2190, 2019.
[29] Avita Katal, Mohammad Wazid, and Rayan H Goudar. Big data: issues, challenges, tools and good practices. In 2013 Sixth international conference on contemporary computing (IC3), pages 404–409. IEEE, 2013.
[30] Jimmy Lin. The lambda and the kappa. IEEE Internet Computing, 21(05):60–66, 2017.
[31] Gaurav Mishra. Setting up a Data Lake architecture with AWS. [Na spletu]. Dostopno: https://www.srijan.net/resources/blog/setting-up-a-data-lake-architecture-with-aws. [Dostopano: 27-okt-2022], avgust 2019.
[32] Pointer, Ian. What is Apache Spark? The big data platform that crushed Hadoop. [Na spletu]. Dostopno: https://www.infoworld.com/article/3236869/what-is-apache-spark-the--big-data-platform-that- crushed-hadoop.html. [Dostopano: 27-okt-2022], marec 2020.
[33] Zhi-Hua Zhou, Nitesh V Chawla, Yaochu Jin, and Graham J Williams. Big data opportunities and challenges: Discussions from data analytics perspectives [discussion forum]. IEEE Computational intelligence magazine, 9(4):62–74, 2014.

▪

Martina Šestak je asistentka z doktoratom in raziskovalka na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Trenutno se raziskovalno ukvarja s sodobnimi arhitekturami za upravljanje velepodatkov, podatkovnimi bazami grafov in podatkovnimi prostori. Raziskovalno in aplikativno sodeluje tudi na več projektih, ki se odvijajo v okviru Inštituta za informatiko.

▪

Muhamed Turkanović je visokošolski učitelj, izredni profesor, na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Je vodja raziskovalne skupine Blockchain Lab:UM Inštituta za informatiko, namestnik predstojnika Inštituta za informatiko, vodja slovenskega EDIH-a DIGI-SI, vodja Digitalnega inovacijskega stičišča Univerze v Mariboru, vodja projektov H2020, Horizont Evropa, Interreg Alpine Space ter ARRS CRP. Njegovi trenutni raziskovalni interesi vključujejo področja tehnologij veriženja blokov, podatkovnih tehnologij ter digitalnih identitet.

Popestritev predavanj o kibernetski varnosti z interaktivnimi računalniškimi simulacijami

Saša Divjak
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko
sasa.divjak@fri.uni-lj.si

Izvleček

Prispevek obravnava problem, kako narediti predavanja s področja kibernetske varnosti bolj zanimiva in razumljiva dijakom, študentom in udeležencem kakšnih izobraževalnih tečajev. Rešitev najde v simulacijah in animacijah, ki lahko vizualno ponazorijo dogajanja v računalniških mrežjih, na katera prežijo napadalci. Po kratkem vpogledu v sorodja dela se posveti kiberletom, to je spletnim aplikacijam, ki take simulacije predstavljajo. Delo prikazuje nekaj tipičnih simulacij in v grobem poda ozadje, ki omogoča tudi nadgradnjo z lastnimi primeri.

Ključne besede: Kibernetska varnost, računalniške simulacije, izobraževanje.

Enriching cyber security lectures with interactive computer simulations

Abstract

The paper deals with the problem of how to make lectures in the field of cyber security more interesting and comprehensible to pupils, students and participants of certain educational courses. The solution is found in simulations and animations that can visually illustrate the events in computer networks that attackers are lurking on. After a brief insight into similar solutions, the paper focuses on cyberlets, i.e. online applications that present such simulations. The paper presents some typical simulations and gives a rough background, which also allow for the development of own examples.

Keywords: Cyber security, digital simulations, education

1 UVOD

Običajna predavanja s pomočjo PPT prosojnic so lahko tudi dolgočasna in jih skušamo popestriti z različnimi vložki, kot so video, morda vmesna vprašanja s pomočjo glasovalnih aplikacij itd.

Eden od možnih pristopov je tudi uporaba različnih, po možnosti interaktivnih simulacij in poigritve predavanj.

Takšne, grafično podprte simulacije in animacije najdemo na primer na področju fizike, ki je za kaj takega zelo priročna veda. Med takimi rešitvami najdemo na primer fizlete [1] (angleško physlets), ki so bili v fizikalno področje usmerjeni spletni apleti. Najdemo jih v več svetovnih jezikih, tudi v slovenščini [2]. Danes jih seveda nadomeščajo istoimenske simulacije, tipično programirane z JavaScript. Seveda obstaja ogromno zelo kvalitetnih računalniških simulacij s področja fizike, a to ni fokus te predstavitve. Po analogiji s skovanko »physlets« lahko za področje kibernetske varnosti sestavimo besedo »cyberlets« (cyber applets), kar lahko prevedemo v kiberlete. Seveda na spletu najdemo tudi podobne skupine simulacij za druga naravoslovna področja (»mathlets« za matematiko [3], »chemlets« za kemijo [4],..)

2 SORODNA DELA

Simulacija kibernetskih napadov, imenovana tudi simulacija groženj, je nastajajoča varnostna tehnologija IT, ki lahko pomaga odkriti vrzeli, ranljivosti in napačne konfiguracije v naši varnostni infrastrukturi.

Simulacija nasprotnika je postopek posnemanja vedenja napadalca. Ponuja zmožnost testiranja odpornosti organizacije proti naprednemu napadalcu. Čeprav se nasprotnikova simulacija na prvi pogled sliši podobno kot avtomatizirano preskušanje vdorov, ta vrsta simulacije pokriva širši spekter varnostne infrastrukture: določanje različnih poti napada, ki bi jih lahko ubral nasprotnik, blaženje groženj in izbira primernih sanacijskih načrtov.

Simulacija se nanaša na zmožnost posnemanja tehnik, postopkov in taktik zlonamernih akterjev. Večina orodij in platform za simulacijo napadov zagotavlja avtomatsko ali polavtomatizirano sredstvo za pridobitev napadalčevega pogleda na žrtvino omrežje.

Obstaja kar nekaj platform oziroma orodij za pridobivanje potrditve resničnega varnostnega stanja v neki organizaciji [7]. Te rešitve so običajno komercialne, v večini primerov pa ponujajo le preizkusno verzijo. Nekatera sorodna orodja pas so odprtokodna.

Taka orodja so namenjena tudi usposabljanju osebja v organizacijah, pa tudi za upravljanje kibernetske varnosti. Tako namen in tudi sposobnost takih orodij verjetno presegata potrebe, ki naj bi jih imeli pri poučevanju kibernetske varnosti v sklopu klasičnega izobraževanja tako v gimnazijah kot tudi na univerzitetnem nivoju. Večinoma so tudi zelo poglobljena. Kljub pomislekom, da so večinoma komercialna, lahko nudijo ideje, ki bi jih uporabili pri razvoju simulacij in animacij za popestritev naših predavanj. Zato naštejmo nekaj primerov:

Attack simulator [5]: Pri simulaciji kibernetskega napada organizacija posnema dejanski vdor v lastno omrežje, infrastrukturo in sredstva z uporabo orodij, taktik in postopkov znanih kibernetskih kriminalcev. Cilj simulacije je odkrivanje ranljivosti v obrambi organizacije, ki jih lahko odpravi varnostna ekipa, s čimer se zmanjša izpostavljenost napadom iz resničnega sveta.

FourCore Attack [6]: Simulacija vdora in napada, Prepoznavanje ranljivosti s posnemanjem resničnih poti napada, ki jih uporabljajo akterji groženj.

Firedrill [8]: firedrill je odprtokodna knjižnica podjetja FourCore Labs za preprosto izdelavo simulacij zlonamerne, izsiljevalske programske opreme. Zgradili so skupino štirih različnih simulacij napada, ki jih lahko uporabljamo in nadgrajujemo: simulacijo izsiljevalske programske opreme, simulacijo odkrivanja, obvod UAC in simulacijo vztrajnosti.

Infection Monkey [9]: Infection Monkey je odprtokodna platforma za simulacijo vdorov in napadov, ki nam pomaga preveriti veljavnost obstoječih kontrol in ugotoviti, kako lahko napadalci izkoristijo naše trenutne varnostne vrzeli v omrežju. Na voljo je vizualni prikaz zemljevid našega omrežja z vidika napadalca, z razčlenitvijo strojev, ki jih je Monkey uspel vdreti. Pride do okužbe naključnega računalnika in sledi samodejno odkrivanje varnostnega tveganja. Preizkusimo lahko različne scenarije:krajo poverilnic, ogrožanje naprav in druge varnostne napake.

CyberCIEGE [10]: CyberCIEGE pokriva široko paleto tem kibernetske varnosti. Igralci kupujejo in konfigurirajo računalnike in omrežne naprave, da so zahtevni uporabniki zadovoljni (npr. z zagotavljanjem dostopa do interneta), hkrati pa ščitijo sredstva pred različnimi napadi. Igra vključuje številne različne scenarije, od katerih se nekateri osredotočajo na osnovno usposabljanje in ozaveščanje, drugi pa na naprednejše koncepte varnosti omrežja.

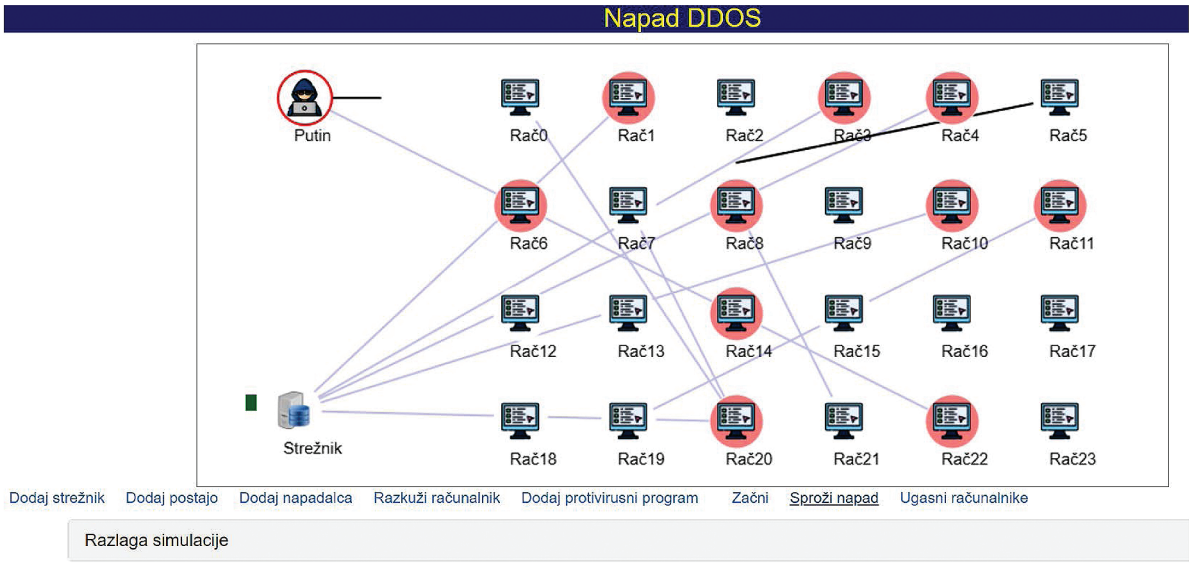
V nadaljevanju se bomo posvetili spletni aplikaciji, ki je enostavna in morda za dodatek pri predavanjih bolj primerna.

3 KIBERLETI

3.1 Razvoj kiberletov

V uvodu smo omenili fizlete predvsem zato, ker temeljijo na lastni, vendar odprtokodni platformi, ki je bila izhodišče za razvoj lastnih simulacij s področja kibernetske varnosti. Po zgledu fizletov smo razvili »kiberlete«, torej spletne računalniške animacije in simulacije, seveda grafično podprte, namenjene popestritvi predavanj in poučevanja na področju kibernetske varnosti.

Tako fizleti kot kiberleti so pisani v JavaScript. V obeh primerih je središče animacije knjižnica Animator.js, ki vsebuje kot razrede tipične gradnike s svojega problemskega področja. V primeru fizletov so to predvsem fizikalni delci, v primeru kiberletov pa so to računalniške komponente omrežja. Predvsem so to delovne postaje (stacionarni računalniki ali notesniki), pa strežniki. V primeru fizletov potekajo simulacije med spreminjanjem časa v inkrementih



Slika 1: **Simulacija porazdeljenega napada DDOS. Rdeče pobarvani računalniki so okuženi in delujejo v mreži BotNet**

dt, torej z oponašanjem sicer zveznega poteka pri fizikalnih pojavih. V primeru kiberletov pa v bistvu prehajamo od dogodka do dogodka. Več o ozadju kiberletov najdemo pri opisu njihove arhitekture.

Za boljše razumevanje delovanja kiberletov si najprej oglejmo en primer.

Primer take simulacije kaže slika 1, ki je utrinek iz računalniške upodobitve znanega porazdeljenega napada za zavračanje storitev (DDOS)

V tej simulaciji imamo mrežo delovnih postaj in en strežnik. Ena postaja pa je »posebna« in predstavlja napadalca. Simulacija začne s klikom na gumb »Začni«. Računalniki si začenjajo naključno pošiljati sporočila. In to na žalost počne tudi napadalec. Posebnost strežnika je, da na vsako sporočilo odgovori s svojim sporočilom (kar naj bi predstavljalo storitev, ki jo od njega pričakuje kličoči računalnik).

Sporočila napadalca pa so okužena. Prejemnik takega sporočila pordeči in odslej tudi sam s svojimi sporočili kuži svojo okolico. Sčasoma je tako na našem zaslonu čedalje več okuženih računalnikov.

V primernem trenutku kliknemo na link »Sproži napad«- Vsi okuženi računalniki začenjajo svoja sporočila usmerjati na napadeni strežnik, ki postaja preobremenjen (to kaže rast zelenega stolpca ob njem). To je le del tega scenarija.

3.2 Arhitektura kiberletov

Grobo zgradbo kiberletov prikazuje naslednji poenostavljen razredni diagram:

Glavni razred, Animator vsebuje podatke o sceni in njeni velikosti na spletni strani. Vsebuje tudi podatke o komponentah, ki to sceno sestavljajo. Poleg tega vsebuje metode za časovno krmiljenje simulacije oziroma animacije.

Objekti iz razreda Component imajo podatke o svojem položaju na sceni, o sliki, ki ta tak objekt predstavlja, o tipu (obliki) objekta ter o tem, ali je objekt fiksiran na scenu, ali pa ga lahko izbiramo in premikamo po sceni.

Objekti iz razreda Asset so v bistvu naprave, tipično računalniki ipd, ki jih uporabljamo v simuliranem okolju in jih primerno ščitimo.

Objekti iz razreda Person so uporabniki teh naprav. Njihove lastnosti vsebujejo podatke o njihovi strokovni in IT usposobljenosti, o njim dodeljenim napravam in o njihovih dovoljenjih.

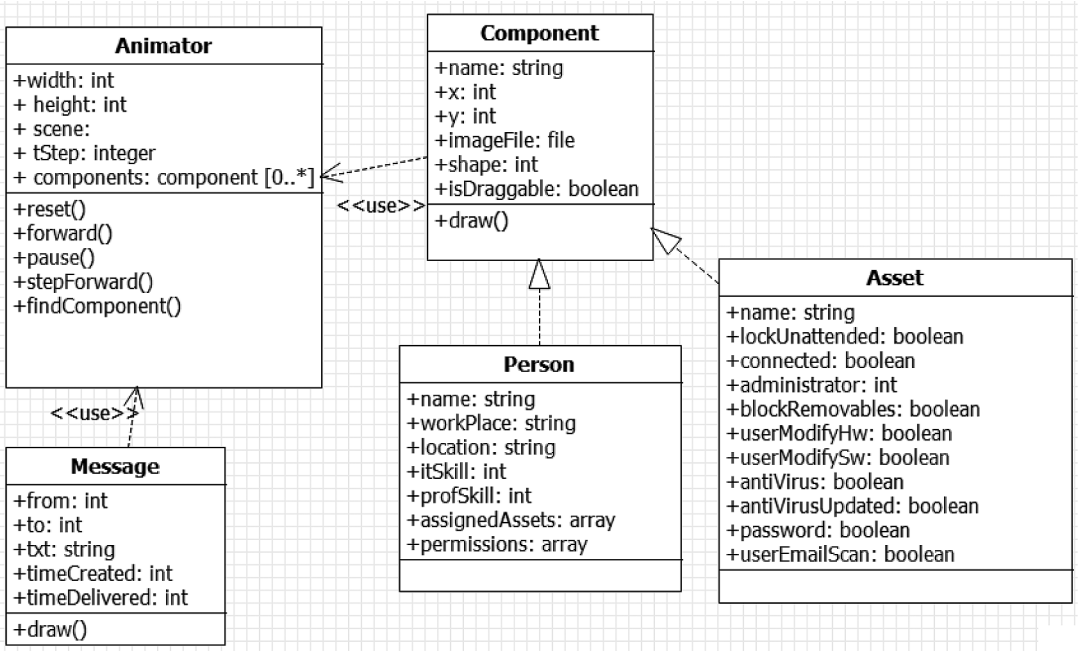
Objekti iz razreda Message vsebujejo podatke o sporočilih, ki potekajo od naprave do naprave.

To je osnovni koncept zgradbe kiberletov, ki pa se od primera do primera razlikujejo in so njihove dodatne značilnosti del JavaScript kode posamezne spletne strani.

3.3 Kaj še kiberleti omogočajo

V trenutni izvedbi so na voljo naslednje

- Širjenje okužbe po omrežju simulacije:
- Napad DDOS
- Napad moža v sredini
- Zaščiteni (šifrirani) kanali



Slika 2: **Poenostavljen razredni diagram kiberletov**

- Požarni zid
- Primerjava IDS in IPS
- Kerberos
- Internet stvari in kibernetska varnost
- Varnost v pisarni (ali manjšem podjetju)

Kiberlete najdemo na naslovu: http://sasa.musiclab.si/KIBERLETI/

V takih simulacijah lahko na »sceno« dodajamo različne komponente. Predvsem računalnike, ki so lahko v vlogi končnih delovnih postaj (tudi notesnikov) in strežnikov.

Računalniki se lahko okužijo, lahko jih tudi razkužimo in lahko jim dodajamo protivirusno zaščito. Računalnike lahko tudi vklapljamo ali izklapljamo. Dodajamo lahko nove strežnike, delovne postaje in tudi napadalce. Mrežo lahko (tudi med potekom simulacije) poljubno širimo.

Računalnike lahko izberemo s klikom miške in jih prestavljamo po zaslonu, morda zaradi boljšega pregleda.

V animacijah je sled sporočil običajno pobarvana modro, po koncu sporočila pa ostane še nekaj časa vidna svetlomodro. Simulacije kažejo vnaprej konfigurirano mrežo strežnikov, delovnih postaj (uporabniških računalnikov, notesnikov ipd) in napadalca.

Delovne postaje med seboj komunicirajo s sporočili. Posebnost strežnikov je, da na vsako prejeto sporočilo (zahtevek) pošljejo odgovor (izvedejo storitev, svoje sporočilo) .

Napadalci so tudi računalniki, ki pa le pošiljajo sporočila, ne morejo pa jih prejemati (so uporabnikom omrežja neznani)

Med kiberleti najdemo tudi simulacijo drugih pojavov, vezanih na kibernetsko varnost: napad moža

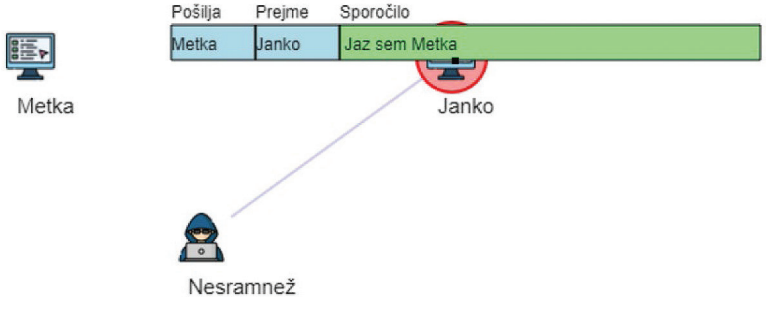


Figure 3: **Utrinek iz simulacije napada moža v sredini.**

Figure 4: **Utrinek iz simulacije šifriranih kanalov.**

v sredini, prenos podatkov preko šifriranih kanalov, vloga požarnega zidu in še kaj.

Naslednji zgled je na primer napad moža v sredini, kot ga kaže slika 3

Gre le za utrinek s simulacije, saj je celotna animacija bolj zgovorna. Napadalec je prestregel komunikacijo med Jankom in Metko in se v sporočilu Janku pretvarja, da je Metka. V resnici je format sporočila bolj zapleten. V tej simulaciji pa je napadalec spremenil glavo sporočila (in tudi samo besedilo), v katerem sta med drugim tudi podatka o pošiljatelju in prejemniku sporočila.

Pa še en utrinek iz simulacije šifriranih, torej zaščitenih kanalov. Prikazuje ga slika 4.

V tej simulaciji si Janko in Metka najprej izmenjala javna ključa. Nato pa si pošiljata sporočila, ki pa ga lahko vsakdo dešifrira le s svojim privatnim ključem. Uporabljeno je bilo resnično asimetrično šifriranje v JavaScript.

Med bolj kompleksimi simulacijami najdemo simulacijo pisarne (ali manjšega podjetja), ki mora z vidika kibernetske varnosti skrbeti za stanje svoje zaupnosti, celovitosti in razpoložljivosti. Na voljo ima zaposlene in računalniško opremo s svojimi lastnostmi, pa tudi politiko varovanja. Spodnja slika kaže utrinek s take simulacije, v okviru katere prihaja do različnih dogodkov, na primer poskusa vdora v strežnik, kraje podatkov ipd.

Kiberleti se ne izognejo niti bolj sodobni problematiki, kot je na primer kibernetska varnost pametnega doma, pametnega vozila ali oseb, opremljenih s pametnimi napravami, Spodnja slika kaže utrinek s take animacije.

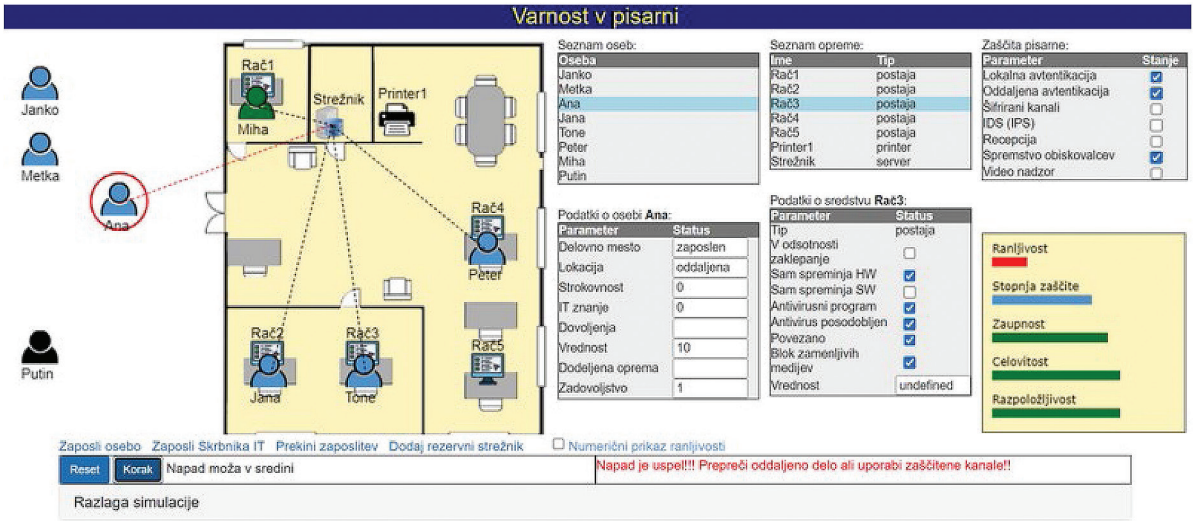


Figure 5: **Simulacija kibernetske varnosti v pisarni ali manjšem podjetju**

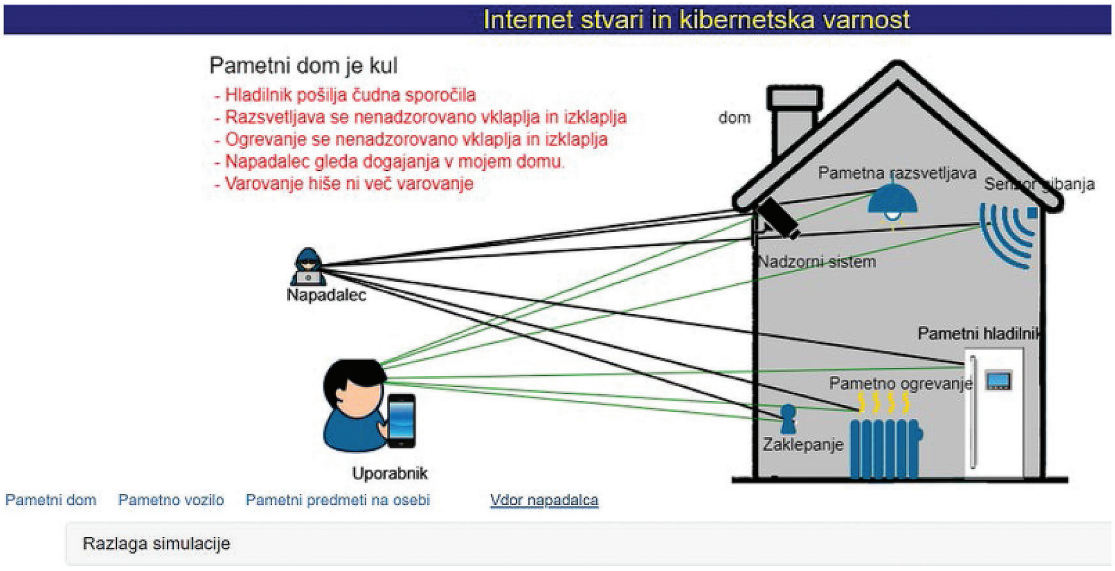


Figure 6: **Internet stvari in kibernetska varnost**

3.4 Razvoj novih kiberletov

Jedro kiberletov je zasnovano modularno in objektno usmerjeno tako, da si lahko izkušen računalnikar tudi sam izmišlja sorodne scenarije in sam sestavlja svoje simulacije. Seveda mora imeti za kaj takega primerno predznanje HTML in JavaScript.

Trenutna, še razvojna različica se nahaja na prej omenjenem naslovu. Celotna koda skupaj z vsemi JavaScript datotekami in slikami je zaenkrat prosto dostopna in seveda brezplačna pri avtorju. Zanjo velja licenca »CopyLeft«.

4 Zaključki

Kiberleti so uporabni tudi v sklopu spletnih učnih gradiv, saj so pisani v JavaScript in HTML. Že v trenutni obliki predvidevajo dvojno uporabo: Pri odprtju take strani je sama razlaga simulacije skrita. To je bolj primerno za predavanja, kjer kažemo le simulacijo. S klikom pa se razpre razlaga simulacije, kar je primerno bodisi pri pripravi predavatelja bodisi pri samoučenju.

Čeprav so navedene simulacije bolj ali manj preproste, lahko služijo popestritvi predavanj o kibernetski varnosti. Kot je pri takih simulacijah značilno, niso vse primerne za vsa možna pedagoška okolja. Tako so verjetno večinoma pretirane za uporabo na srednješolskem nivoju, vsaj kar se tiče predmeta Informatika v klasičnih gimnazijah. V srednjih šolah s področja računalništva pa bi bile lahko koristne. Tudi na univerzitetnem nivoju so uporabne. Ker so odprtokodne in prosto dostopne omogočajo pa tudi nadgradnjo v obliki seminarskih nalog.

Zanimiva je tudi misel, da bi kiberlete nadgradili z možnostjo vključevanja vanje preko drugih, v omrežje navezanih računalnikov. Po analogiji s spletnimi mnogouporabniškimi igricami bi lahko uvedli nekakšne sodelavne simulacije. Tako bi lahko bila taka mnogouporabniška izkušnja še bolj realistična.

5 LITERATURA

[1] Wolfgang Christian, Mario Belloni, Anne Cox, Melissa Dancy: Physlets; https://www.physport.org/curricula/physlets/
[2] Saša Divjak: Fizika s fizleti; http://sasa.musiclab.si/fizleti/
[3] MIT mathlets; https://mathlets.org/
[4] Chemlets (chemistry applet animations): https://lead.mst.edu/mathscicon/chemistry/chemlets/
[5] Cyber Security Awareness Training with Automated Phishing Simulations; https://attacksimulator.com/
[6] FourCore ATTACK Security Control Validation, https://fourcore.io/
[7] Hardik Manocha: Top 10 Awesome Open-Source Adversary Simulation Tools; https://fourcore.io/blogs/top-10-open-source-adversary-emulation-tools
[8] firedrill: A malware simulation harness; https://github.com/FourCoreLabs/firedrill
[9] Simulate, Validate, and Mitigate with the Infection Monkey; https://www.akamai.com/infectionmonkey
[10] CyberCIEGE: https://nps.edu/web/c3o/cyberciege
[11] Saša Divjak: Kibernetski napadi; http://sasa.musiclab.si/KIBERLETI/
[12] Kennedy Mwangi: Implementing Public Key Cryptography in JavaScript; https://www.section.io/engineering-education/implementing-public-key-cryptography-in-javascript/

▪

Saša Divjak je zaslužni profesor Univerze v Ljubljani, Fakultete za računalništvo in informatiko. Bil je Vodja odseka za avtomatiko, robotiko in biokibernetiko in kasneje načelnik oddelka za elektroniko na Institutu Jozef Stefan, pomočnik direktorja Iskre Delte, prodekan za raziskovalno delo na Fakulteti za elektrotehniko in računalništvo, prodekan za raziskovalno delo na Fakulteti za računalništvo in informatiko, dekan na Fakulteti za računalništvo in informatiko Univerze v Ljubljani , gostujoči profesor na Fakulteti za informatiko Univerze v Vidmu, Predstojnik Katedre za programsko opremo na Fakulteti za računalništvo in informatiko v Ljubljani. Predsednik Slovenske sekcije IEEE. Predstojnik Laboratorija za računalniško grafiko in multimedije na Fakulteti za računalništvo in informatiko, odgovoren za več projektov s področja multimedijskih tehnologij, Predsednik računalniške sekcije v sklopu slovenskega društva IEEE, član Izvršnega odbor ACM Slovenija, Senior member IEEE. Predsednik mednarodnega združenja CoLoS (Conceptual learning of Science). Predsednik generalne skupščine mednarodnega združenja HSci (Hands on Science), član in predsednik Evropske akademije znanosti (www.eurasc.org). Nosilec več projektov, predvsem s področja simulacije in avtomatizacije različnih tehnoloških procesov. Koavtor programske opreme prvih slovenskih robotov, sodelavec na italijanskem izobraževalnem projektu »Tovarne prihodnosti«. Nosilec več domačih in mednarodnih projektov s področja multimedijskih tehnologij v izobraževanju.

50 let od uvedbe predmeta računalništvo v srednje šole: poskusni pouk in učbenik

Ivan Bratko[1], Iztok Lajovic[2], Vladislav Rajkovič[3]
[1] Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Ljubljana, Slovenija
[2] Kreativni sistemi, Ljubljana, Slovenija
[3] Univerza v Mariboru, Fakulteta za organizacijske vede
bratko@fri. uni-lj.si, Iztok.Lajovic@kres-ks.si, Vladislav.Rajkovic@gmail.com

Izvleček

Letos mineva 50 let od uvedbe poskusnega pouka računalništva v slovenske srednje šole. V tem prispevku navajamo nekaj zgodovinskih dejstev o projektu uvedbe pouka računalništva pred 50 leti ter opišemo naše lastne spomine na to, kako smo sodelovali pri poskusnem pouku računalništva in kako je nastal učbenik za ta predmet.

Ključne besede: srednješolski pouk računalništva, poskusni pouk, učni načrt, učbenik

Introducing Informatics in secondary schools 50 years ago: Experimental teaching and text book

Abstract

Fifty years ago, experimental teaching of informatics was introduced in secondary schools in Slovenia. In this paper, we present some historical facts about the introduction of the informatics course 50 years ago and describe our memories of our own involvement in the teaching of informatics at that time, and the writing of the textbook for this course.

Keywords: Teaching informatics in secondary schools in Slovenia, experimental teaching, informatics curriculum, informatics textbook

1 UVOD

Letos mineva 50 let od uvedbe poskusnega pouka računalništva v slovenske srednje šole. Projekt uvedbe tega pouka je začel Zavod za šolstvo Republike Slovenije l. 1971. V nekaj letih, do š.l. 1974/75, je predmet Računalništvo zajel 65 srednjih šol in 2500 srednješolcev, izšel je učbenik, usposobljenih je bio 75 učiteljev za ta predmet.

S tem projektom je Slovenija močno prehitela druge republike tedanje Jugoslavije in bila po nekaterih ocenah v tem pogledu med vodilnimi v Evropi. Boris Lipužič, tedanji direktor Zavoda za šolstvo, je l. 2010 v opisu tega projekta zapisal [8]: »Poročilo Generalnega direktorata za izobraževanje in kulturo Evropske komisije na sedežu EU v Bruslju za leto 2000/01 navaja, da je Slovenija začela vpeljevati pouk računalništva v srednjih šolah že leta 1974, celo pred Zvezno republiko Nemčijo – ta je uvajanje zastavila šele v poznih sedemdesetih letih (*Basic Indicators on the incorporation of ICT into European Education Systems, Facts and figures, Eurydice 2001, str. 17, Brussels*).« Omenimo, da gornji citat iz Lipužičevega članka ni povsem dobeseden prevod navedbe v originalnem dokumentu Evropske komisije. Tudi ni povsen

* Prispevek je bil že objavljen v okviru konference Informacijska družba 2021. V reviji ga po-objavljamo zaradi vsebinske pomembnosti prispevka.

jasno, kaj je točno mišljeno z letnico 1974. Vendar tudi originalno besedilo nedvomno uvršča pouk v Sloveniji med najbolj zgodnje v Evropi.

V pričujočem prispevku navajamo nekaj zgodovinskih dejstev o projektu uvedbe pouka računalništva pred 50 leti ter opišemo naše lastne spomine na to, kako smo sodelovali pri poskusnem pouku računalništva in kako je nastal učbenik za ta predmet.

2 PROJEKT UVEDBE SREDNJEŠOLSKEGA POUKA RAČUNALNIŠTVA

V tem razdelku so navedena dejstva o projektu uvedbe pouka računalništva v srednje šole v Sloveniji.

5. 4. 1971 je bilo poslano prvo vabilo z Zavoda za šolstvo RS Slovenije za sestanek za pripravo projekta za postopno uvajanje pouka o računalništvu v srednje šole.

13. 4. 1971 je potekal v direktorjevi pisarni na Zavodu sestanek o pouku računalništva na srednjih šolah. S strani Zavoda sta sodelovala Milan Adamič in Branko Roblek, s strani Instituta »Jožef Stefan« Vladislav Rajkovič, Cveto Trampuž in Mira Volk, Fakulteto za elektrotehniko je zastopal Jernej Virant, Republiški računski center Janez Lesjak, INFIM Egon Zakrajšek in Višjo tehniško šolo Maribor Milan Kac. Sprejeta sta bila dva sklepa:

V šolskem letu 1971/72 se uvede poskusni pouk računalništva v štirih izbranih šolah. Pouk naj bi se odvijal v obliki izbirnega predmeta v okviru ur za praktična znanja.

Treba je izdelati učni načrt in z njim v skladu pripraviti ustrezni učbenik.

Koncem aprila 1971 je Zavod pripravil Projekt uvajanja pouka o računalništvu v srednje šole. Projekt je vodil Branko Roblek. Predvideno je bilo postopno uvajanje s sprotno evalvacijo v šolskih letih od 1971/72 do 1975/76.

Postopno naj bi se povečevalo število šol. Posebej je bil izpostavljen problem učnega kadra. V začetku naj bi poučevali računalniški strokovnjaki iz okolja, ob sprotnem usposabljanju učiteljev iz šol. V ta namen je bil organiziran tečaj za učitelje in pripravljeno gradivo »Računalništvo« sedmih avtorjev (slika 1).

12. 7. 1971 je Zavod razposlal vabilo petim gimnazijam in dvema tehniškima šolama za pričetek poskusnega izvajanja pouka računalništva.

S pričetkom projekta se je pričela tudi priprava učnega načrta predmeta. Pri tem smo se v večji meri opirali na priporočila IFIP-a, ki je leta 1970 organiziral 2. svetovno konferenco Computers in Education. Posebno vzpodbudo je predstavljal tudi IFIP-71, svetovni računalniški kongres, ki je potekal v Ljubljani.

Učni načrt je obsegal 52 ur. Od tega je bilo namenjenih 8 ur pripravi problema in rešitve ter 22 ur programskemu jeziku fortran. Predvideno je bilo praktično delo na računalniku: izdelava in testiranje programa.

Leta 1974 je izšel učbenik Uvod v računalništvo, ki je bil ponatisnjen še sedemkrat v več kot 20.000 izvodih [3].

Iz leta v leto se je povečevalo število šol, kjer se je poučeval predmet računalništvo, število učencev pa tudi število učiteljev, saj se izobraževanje na že omenjenem posebnem tečaju za učitelje. Ti podatki so prikazani v tabeli 1.

Rezultati projekta so bilo objavljeni tudi na IFIP 2nd World Conference Computers in Education leta 1975 [6].

Leta 1977 so se pričela tudi tekmovanja srednješolcev iz računalništva. Leta 1981 je v 3.000 izvodih izšla zbirka nalog [1].

Leta 1980 je izšel učbenik »Osnove tehnike in proizvodnje« v okviru skupnih izobraževalnih osnov v srednjih šolah. V tem učbeniku je bilo tudi poglavje »Informatika in računalništvo« na 38 straneh [4]. Učbenik je bil še dvakrat ponatisnjen v skupnem številu 52.000 izvodov.

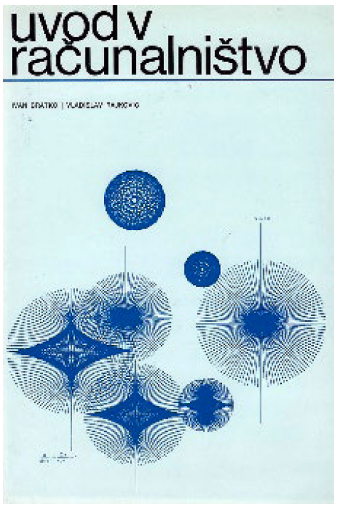
Leta 1981 je založba Univerzum izdala mapo s prosojnicami in diapozitivi v pomoč učiteljem pri poučevanju informatike in računalništva.

Že koncem sedemdesetih let se je začel poučevati programski jezik pascal. Temu je sledil novi učbenik

Tabela 1: **Rast obsega pouka po šolskih letih**

Š. leto	Šol	Razredov	Dijakov	Učiteljev
1971/72	7	12	200	-
1972/73	20	30	500	25
1973/74	40	75	1800	50
1974/75	65	100	2500	75

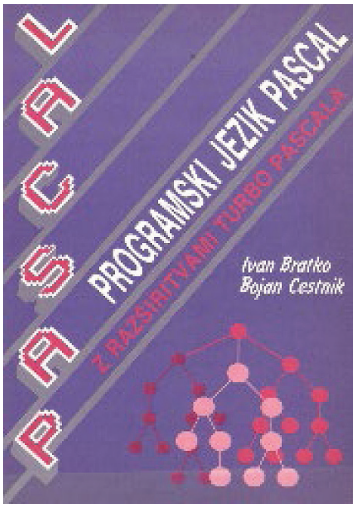
Slika 1: **Gradivo za tečaj za učitelje**



Slika 2: **Prvi učbenik**

Slika 3: **Učbenik za Osnove tehnike in proizvodnje**

Slika 3: **Učbenik iz l. 1984**



Slika 4: **Učbenik iz l. 1990**

Računalništvo s programskim jezikom pascal, ki je izšel 1984 [5]. Ta učbenik je bil ponatisnjen še štirikrat v skupni nakladi 24.000 izvodov. Zadnjič leta 1989.

Z razvojem programskega jezika pascal, natančneje z razširitvami turbo pascala, se je pojavila tudi potreba po ustreznem učbeniku. Ta je izšel 1990 v nakladi 5.000 izvodov [2].

3 POSKUSNI POUK IN UČBENIK

Jeseni 1971 se je začel poskusni pouk računalništva na izbranih srednjih šolah. Na gimnazija Bežigrad smo učili soavtorji tega prispevka. Ta pouk nam je bil vsem trem v veliko veselje, pravzaprav tako kot skoraj vse, s čimer smo se takrat ukvarjali.

Bili smo eno leto po diplomi na Fakulteti za elektrotehniko v Ljubljani. Kar nas je posebej kvalificiralo za ta pouk, je bilo to, da smo se med študijem, pravzaprav bolj ob študiju, naučili tudi nekaj računalništva, med drugim programirati v algolu in fortranu. Prve programerske korake smo naredili na računalniku Zuse na Fakulteti za matematiko in fiziko. Toda za delo na tem računalniku si moral z našim statusom študenta vstati ob 6h zjutraj, sicer pa je bil računalnik zaseden. No, ko smo prešli na fortran in začeli programirati za nekatere znane profesorje na Univerzi in Institutu Jožef Stefan, so se nam še pred diplomo pogoji za delo na računalnikih močno popravili.

Ko smo začeli s poskusnim poukom računalništva, se nam je iztekalo prvo leto naše zaposlitve v Oddelku za elektroniko na Institutu Jožef Stefan. Ob zaposlitvi so nam dodelili skupno »pisarno«, osamljeno sobo na sicer povsem neobdelanem podstrešju. Takrat, ob navdušenju nad prvo zaposlitvijo v zanimivem raziskovalnem okolju skoraj niti nismo opazili, kako neprimeren delovni prostor je bil to. Pot do naše pisarne je vodila po neobljudenem podstrešju med tramovjem in ovirami, ki so naključno ležale po podstrešju. Hoja do pisarne je zato bila svojevrstna pustolovščina, posebej v temi ponoči. Naša soba je imela le majhno strešno okno, pravzaprav strešno lino. Poletne temperature v tem prostoru so bile neznosne. Toda nič od tega nas ni posebej motilo, saj smo bili tako zaposleni s svojim raziskovalnim delom in aplikativnim delom, nenehnimi pogovori o novih in novih idejah, ves čas se je dogajalo kaj zanimivega.

Del tega vzdušja so bile tudi naše priprave na pouk računalništva. Temu je bilo namenjeno dopoldne vsak torek v tednu, ko je imel tisto popoldne prvi od nas naslednjo uro pouka na gimnaziji. Takrat smo prediskutirali stanje pouka, izkušnje iz prejšnjega tedna ter naredili načrt, kaj bomo učili ta teden.

Vseskozi smo bili trdno odločeni, da se držimo nekaterih osnovnih načel: da bomo spodbujali aktivno delo učencev, učili reševanje problemov z računalniki z mnogimi primeri in da bomo veliko od tega dejansko sprogramirali ter kolikor bo možno tudi izvedli na računaniku. Za pouk nam je bil dosegljiv, sicer v zelo omejenem obsegu le računalnik IBM 1130 na Fakulteti za matematiko in fiziko na Jadranski cesti. Programski jezik je bil fortran. Mislim, da smo nazadnje vsi dosegli, da je vsak naš učenec napisal vsaj po en svoj program, ga spravil na luknjane kartice in izvedel na računalniku. Program na luknjanih karticah je izgledal kot paket kart, ki smo ga navadno speli z elastiko, da se kartice ne bi po nesreči pomešale. V tej zvezi se spomnimo zabavnega dogodka, ko je nek dijak v svoji raztresenosti vstavil v računalnik svoj paket lunkjanih kartic kar z elastiko vred. Čitalnik kartic je ob branju kartic elastiko takoj raztrgal in se ob tem pokvaril …

Med našimi učenci pri takratnem pouku tistega leta ali kakšno leto kasneje so bila tudi imena, ki so kasneje postala dobro prepoznavna, med drugimi: prof. dr. Matjaž Gams, dr. Marko Gričar, prof. dr. Marko Petkovšek, prof. dr. Franc Solina.

Poskusni pouk računalništva je bil deležen zelo pozitivnih odzivov in komisija za uvajanje pouka sklenila: zdaj potrebujem učbenik, kaj če bi poskusila Bratko in Rajkovič?

Učbenik je nastajal ob izvajanju poskusnega pouka računalništva in je bil oblikovan po našem izvedenem pouku v prvih dveh šolskih letih pouka.

Knjiga je bila razdeljena v dva dela: (I) Zgradba, delovanje in uporaba računalnika, (II) Programski jezik fortran. Poglavja v I. delu so bila: 1. Uvod, 2. Osnovni pojmi o informacijah in njih predstavitvi, (3) Zgradba računalnika, (4) Odvijanje programa v računalniku, (5) Programski jeziki, (6) Reševanje problemov z računalniki (vključno z značilnimi primeri konstruiranja algoritmov), (7) Uporaba računalnikov, (8) Računalniški sistemi. Poglavje 7 je vsebovalo daljši razdelek o umetni inteligenci. Poleg tega pa tudi razdelek »Ali računalnik ogroža človeka«. Ta razdelek se konča takole: »Morda je najbolj upravičena bojazen pred vmešavanjem računalnikov v človekove osebne stvari. Skrajno neprijetna je namreč zavest, da bi lahko obsežne datoteke z vsemi mogočimi podatki, omogočile dostop do vseh osebnih podatkov o komerkoli.«

Posebna skrb je bila v učbeniku namenjena računalniškemu izrazoslovju. Dolge so bile debate o prevodih posameznih pojmov, kot npr.: pomnilnik, računalniška beseda, naslov, adresa ipd. Marsikatero dilemo nam je pomagal razrešiti tudi jezikoslovec Tomo Korošec. Ko smo ga povprašali, kako naj zapišemo imena programskih jezikov, je dejal takole: »Če niste pogumni, jih pišite z velikimi tiskanimi črkami. Z malo poguma jih pišite z veliko začetnico, če pa imate dovolj poguma jih pišite z malo začetnico, kot pišemo imena jezikov v slovenščini«. In tako smo zapisali fortran, pascal in druge. Razen zelo redko uprabljanih jezikov in PL1, ki je kratica.

Z veseljem lahko ugotovimo, da je skrb za slovensko računalniško izrazoslovje tudi danes zelo živa, na primer v Slovenskem društvu INFORMATIKA, kot tudi na univerzah in inštitutih. Društvo redno vzdržuje *Islovar* računalniških pojmov, ki je prosto dostopen na spletu. Številni pojmi v današnjem Islovarju so tudi v pojmovnem kazalu učbenika. Tudi programski jeziki zapisani z malo začetnico.

4 ZAKLJUČEK

Projekt pred 50 leti je potekal hitro in učinkovito. Deležen je bil pozitivne ocene tudi v mednarodnem

merilu. Pozitivno ocenjujemo tudi težnjo po tem, da je v pouku bilo poudarjeno reševanje problemov z algoritmi ter razvijanje algoritmičnega razmišljanja.

Kaj se je dogajalo s poukom računalništva kasneje, ko sta postajala računalništvo in digitalizacija neprimerno bolj razširjena in pomembna in so se kazale nujne potrebe po spremembah učnih programov za računalništvo? Ne bi mogli reči, da so bile spremembe vedno posrečene in pravočasne. Posebej pa je težko razumeti, zakaj je danes veliko teže doseči spremembe učnega načrta, čeprav se o njihovi potrebnosti strinja praktično celotna računalniška stroka, doma in v svetu [7].

5 VIRI IN LITERATURA

[1] Janez Benkovič, Aleksander Cokan, Mark Martinec, Robert Reinhardt, Branko Roblek. Računalništvo: Zbirka nalog 1. Državna založba Slovenije, 1981.

[2] Ivan Bratko, Bojan Cestnik. Programski jezik pascal z razširitvami turbo pascala. Državna založba Slovenije, 1989.

[3] Ivan Bratko, Vladislav Rajkovič. Uvod v računalništvo, Državna založba Slovenije 1974.

[4] Ivan Bratko, Vladislav Rajkovič. Informatika in računalništvo. V učbeniku Osnove tehnike in proizvodnje, Tehniška založba Slovenije, 1982.

[5] Ivan Bratko, Vladislav Rajkovič. Računalništvo s programskim jezikom Pascal. Državna založba Slovenije, 1984.

[6] Ivan Bratko, Vladislav Rajkovič, Branko Roblek: What should econdary school studenta know about computers: Analysis of an experiment. IFIP 2nd World Conference: Computer in Educatio, 1975.

[7] Andrej Brodnik s soavtorji. Snovalci digitalne prihodnosti ali le uporabniki? Poročilo strokovne delovne skupine za analizo prisotnosti vsebin računalništva in informatike v programih osnovnih in srednjih šol ter za pripravo študije o možnih spremembah (RINOS). Ministrstvo za izobraževanje, 2018.

[8] Boris Lipužič, 2010. Pionirski koraki: 40 let pouka računalništva. Šolski razgledi, številka 17/2010.

▪

Ivan Bratko je profesor računalništva na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Je začetnik raziskav iz umetne inteligence v Sloveniji. Njegovi znanstveni prispevki so med drugim na področjih strojnega učenja, induktivnega logičnega programiranja in učenja kvalitativnih modelov, z uporabo v medicini, ekologiji in robotiki. Njegova bibliografija vsebuje okrog 250 objav, med katerimi je najbolj znana knjiga Prolog Programming for Artificial Intelligence (Pearson Education). Je redni član SAZU, IAS (Inženirska akademija Slovenije), Academia Europaea in EurAI.

▪

Iztok Lajovic je bil samostojni razvijalec informacijskih rešitev. Pri tem se posebej posveča bazam podatkov in komunikacijam v okviru celovitih poslovnih informacijskih sistemov.

▪

Vladislav Rajkovič je zaslužni profesor Univerze v Mariboru. Njegovo področje so računalniški informacijski sistemi s posebnim poudarkom na uporabi metod umetne inteligence v porcesih odločanja.

Iz Islovarja

Islovar je spletni terminološki slovar informatike, ki ga že več kot 20 let ureja jezikovna sekcija Slovenskega društva INFORMATIKA. Slovar je javno dostopen za vpoglede in vnašanje novih izrazov. Slovar najdete na naslovu http://www.islovar.org.

fíksno omréžje -ega -a s *(angl. fixed network)* omrežje, ki uporablja kabelske povezave med vozlišči; prim. žično omrežje, brezžično omrežje, mobilno omrežje

jêdrno omréžje -ega -a s *(angl. core network)* osrednji del komunikacijskega omrežja, ki uporabnikom zagotavlja različne omrežne storitve; prim. omrežje za dostop, robna naprava

omréžje na čípu -a -- -- s *(angl. network on chip, network on a chip, NoC)* omrežni komunikacijski podsistem, najpogosteje med moduli na istem integriranem vezju; prim. sistem na čipu

omréžni naslòv -ega -óva **m** *(angl. network address)* enolični identifikator omrežnega gradnika; prim. domensko ime

omréžni promèt -ega -éta m *(angl. network traffic)* količina v časovni enoti prenesenih podatkov v omrežju; prim. mrežni promet

omréžni skrbník -ega -a m *(angl. network administrator)* kdor skrbi za upravljanje in delovanje računalniškega omrežja ter omrežnih gradnikov; sin. skrbnik omrežja

omréžno okólje -ega -a s *(angl. network environment)* okolje, ki omogoča uporabo omrežnih storitev; prim. omreženo okolje

širokopasôvno omréžje -ega -a *(angl. broadband network)* omrežje, ki z veliko pasovno širino (2) omogoča uporabniku stalno in interaktivno uporabo večpredstavnih vsebin in storitev

začásno omréžje -ega -a s *(angl. ad hoc network)* brezžično omrežje za omejeni čas uporabe, za katero ni potrebno vzpostaviti namenske omrežne infrastrukture

žično omrežje -ega -a ž *(angl. wired network)* fiksno omrežje, v katerem so povezave vzpostavljene po žicah; prim. optično omrežje

Izpitni centri ECDL

ECDL (European Computer Driving License), ki ga v Sloveniji imenujemo evropsko računalniško spričevalo, je standardni program usposabljanja uporabnikov, ki da zaposlenim potrebno znanje za delo s standardnimi računalniškimi programi na informatiziranem delovnem mestu, delodajalcem pa pomeni dokazilo o usposobljenosti. V Evropi je za uvajanje, usposabljanje in nadzor izvajanja ECDL pooblaščena ustanova ECDL Fundation, v Sloveniji pa je kot član CEPIS (Council of European Professional Informatics) to pravico pridobilo Slovensko društvo INFORMATIKA. V državah Evropske unije so pri uvajanju ECDL močno angažirane srednje in visoke šole, aktivni pa so tudi različni vladni resorji. Posebno pomembno je, da velja spričevalo v 148 državah, ki so vključene v program ECDL. Doslej je bilo v svetu v program certificiranja ECDL vključenih že preko 16 milijonov oseb, ki so uspešno opravile preko 80 milijonov izpitov in pridobile ustrezne certificate. V Sloveniji je bilo doslej v program certificiranja ECDL vključenih več kot 18.000 oseb in opravljenih več kot 92.000 izpitov. V Sloveniji sta akreditirana dva izpitna centra ECDL, ki imata izpostave po vsej državi.

V spomin

Strokovni prispevki

Informacije

ISSN 1318-1882
9 771318 188001