VIŠJA STROKOVNA ŠOLA ACADEMIA
MARIBOR

VPELJAVA PODATKOVNEGA CEVOVODA Z UPORABO PYTHONA IN POSTGRESQL ZA PODJETJE KG-EKO, D. O. O.

Kandidat: Denis Narat
Vrsta študija: študent izrednega študija
Študijski program: Računalništvo
Mentor predavatelj: mag. Ervin Schaff
Mentor v podjetju: Franci Jerman
Lektor: Janoš Železnik, prof. slov. jez. in knj.

Maribor, 2026

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani, Denis Narat, sem avtor diplomskega dela z naslovom Vpeljava podatkovnega cevovoda z uporabo Pythona in PostgreSQL za podjetje KG-EKO, d. o. o., ki sem ga napisal pod mentorstvom mag. Ervina Schaffa.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli kot moje lastne – kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor, skladno z njenimi pravili,
- skladno s 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Trbovlje, februar 2026 Podpis študenta:

ZAHVALA

Iskreno se zahvaljujem svojemu mentorju, profesorju mag. Ervinu Schaffu, za strokovno vodenje, usmeritve in potrpežljivost pri nastajanju diplomskega dela. Njegovo znanje in izkušnje so mi bili v veliko pomoč pri raziskovanju področja podatkovnega inženiringa.

Posebna zahvala gre tudi mentorju v podjetju, g. Franciju Jermanu iz podjetja KG-EKO, d. o. o., za dragocen vpogled v delovanje podjetja in poslovne procese. Njegova pojasnila so mi omogočila boljše razumevanje gibanja podatkov v realnem okolju.

Iskreno se zahvaljujem tudi svoji partnerki in družini za podporo, razumevanje in spodbudo skozi celoten študij in pri pripravi diplomskega dela. Vaša prisotnost in vera vame sta mi pomenili ogromno.

Hvala vsem, ki ste mi stali ob strani na tej poti.

POVZETEK

Diplomsko delo obravnava vpeljavo podatkovnega cevovoda v manjšem podjetju KG-EKO, d. o. o. z namenom izboljšanja učinkovitosti obdelave in zanesljivosti podatkov ter avtomatskega generiranja poročil. Predstavljeno je obstoječe stanje, pri katerem podjetje pri vsakodnevnem poslovanju uporablja ročni vnos in obdelavo podatkov v programu Microsoft Excel, kar se ob večjih količinah podatkov izkaže kot časovno potratno in omejeno za nadaljnje analize.

Analiza obstoječega stanja je pokazala, da podatki zaradi nekonsistentnih zapisov, manjkajočih vrednosti in neustrezne strukture niso ustrezno pripravljeni za avtomatsko obdelavo. Takšno stanje povečuje verjetnost napak, zmanjšuje zanesljivost poročanja ter omejuje sledljivost podatkov, kar predstavlja dodatno tveganje pri poslovnih procesih, zlasti pri izdajanju računov. Ročna obdelava podatkov je poleg tega časovno neučinkovita in ne omogoča enostavne uvedbe avtomatiziranega poročanja.

V okviru diplomskega dela je bil zasnovan in implementiran podatkovni cevovod na osnovi vzorca ETL (Extract, Transform, Load). Podatki se zajemajo prek kontroliranega vnosnega obrazca, razvitega v programskem jeziku Python, pri čemer se že ob vnosu izvajajo validacija, standardizacija in logične omejitve. Prečiščeni podatki se nato uporabijo za avtomatizirano generiranje poročil, hkrati pa močno skrajšajo čas obdelave.

Izvedena je bila primerjava treh pristopov obdelave podatkov: ročne obdelave v Excelu, obdelave s knjižnico Pandas in obdelave v relacijski podatkovni bazi PostgreSQL. Rezultati so pokazali, da avtomatizacija prinaša občutni časovni prihranek, pri čemer je PostgreSQL primernejši za večje količine podatkov, Pandas pa za manjše projekte in razvojne faze.

Pomemben del naloge je namenjen tudi vizualizaciji podatkov. Razvite so bile nadzorne plošče v orodjih Microsoft Power BI in Tableau, pri čemer se je Power BI na podlagi praktične uporabe izkazal kot primernejša rešitev za manjše podjetje.

Zaključni del naloge obravnava varnost podatkovnih cevovodov s poudarkom na SQL injection ranljivostih ter potrjuje pomen dosledne uporabe uveljavljenih varnostnih praks. Poseben poudarek je namenjen pravilni obravnavi vhodnih podatkov in uporabi parametriziranih poizvedb, ki bistveno zmanjšujejo tveganje za zlorabe, ter pomenu doslednega upoštevanja varnostnih smernic v vseh fazah obdelave podatkov.

Ključne besede: *podatkovni cevovod, ETL, avtomatizacija poročil, nadzorna plošča, SQL injection*

ABSTRACT

IMPLEMENTATION OF A DATA PIPELINE USING PYTHON AND POSTGRESQL FOR THE COMPANY KG-EKO, D. O. O.

This bachelor's thesis addresses the implementation of a data pipeline in the small enterprise KG-EKO, d. o. o. with the aim of improving data processing efficiency, automated report generation, and data reliability. The existing state is presented, in which the company relies on manual data entry and data processing using Microsoft Excel, which proves to be time-consuming and limits further analysis when larger volumes of data are involved.

The analysis of the existing state showed that due to inconsistent records, missing values, and inadequate data structure, the data are not properly prepared for automated processing. Such conditions increase the likelihood of errors, reduce reporting reliability, and limit data traceability, which introduces additional risks in business processes, particularly in invoicing. In addition, manual data processing does not support the straightforward introduction of automated reporting.

As part of the thesis, a data pipeline based on the ETL (Extract, Transform, Load) pattern was designed and implemented. Data is collected through a controlled input form developed in Python programming language, where validation, standardization, and logical constraints are applied at the point of entry. The cleansed data is then used for automated report generation, significantly reducing processing time.

A comparison of three data processing approaches was introduced: manual processing in Excel, processing using the Pandas library, and processing in a relational PostgreSQL database. The results showed that automation leads to significant time savings, with PostgreSQL being more suitable for larger datasets, while Pandas is more appropriate for smaller projects and development phases.

An important part of the thesis is also dedicated to data visualization. Dashboards were developed using Microsoft Power BI and Tableau, with Power BI proving to be the more suitable solution for a small enterprise.

The final part of the thesis addresses data pipeline security, focusing on SQL injection vulnerabilities, and confirms the importance of consistent application of established security practices, particularly proper input handling and the use of parameterized queries.

Keywords: *data pipeline, ETL, report automation, dashboard, SQL injection*

KAZALO VSEBINE

1. UVOD 10

1.1 Opis področja in opredelitev problema 11

1.2 Namen, cilji in osnovne trditve 11

1.3 Predpostavke in omejitve 12

1.4 Uporabljene raziskovalne metode 13

2. SMISELNOST UVEDBE PODATKOVNEGA CEVOVODA V MANJŠA PODJETJA .. 14

2.1 Časovna primerjava med Excelom in avtomatizirano obdelavo 14

2.2 Analiza stroškovne učinkovitosti 17

2.3 Pregled modernih implementacij podatkovnih cevovodov 20

2.3.1 Batch Processing Pipeline (MacDonald, 2024) 20

2.3.2 Streaming Pipeline (MacDonald, 2024) 21

2.3.3 Lambda Architecture (MacDonald, 2024) 22

2.3.4 Kappa Architecture (MacDonald, 2024) 23

2.3.5 ETL (Extract, Transform, Load) (MacDonald, 2024) 23

2.3.6 ELT (Extract, Load, Transform) (MacDonald, 2024) 24

2.3.7 Data Mesh Pattern (MacDonald, 2024) 25

2.3.8 Data Lake Pattern (MacDonald, 2024) 26

2.3.9 RAG Pattern (MacDonald, 2024) 26

2.4 Primer uporabe s simulacijo poslovnega procesa 29

3. PRIMERJAVA OBDELAVE PODATKOV: PANDAS (EXCEL) PROTI POSTGRESQL(SQL) 39

3.1 Hitrost poizvedovanja in obdelave podatkov 39

3.2 Uporabnost in fleksibilnost pri manjših projektih 42

3.3 Omejitve pri uporabi relacijskih baz v praksi 43

4. PRIMERJAVA ORODIJ ZA VIZUALIZACIJO REZULTATOV (POWER BI PROTI TABLEAU) 45

4.1 Uporabniška izkušnja in enostavnost uporabe 45

4.2 Prilagodljivost in integracija z obstoječimi sistemi 52
4.3 Končna ocena za uporabo v manjšem podjetju 54
5. VARNOST PRI UPORABI PODATKOVNIH CEVOVODOV 56
5.1 Pregled SQL injection ranljivosti 56
5.1.1 SQL injection na ravni vnosnega obrazca 57
5.1.2 Obhod avtentikacije in eskalacija privilegijev (E. Caselli, 2020) 58
5.1.3 Blind SQL injection (E. Caselli, 2020) 60
5.1.4 Zbiranje informacij in pridobivanje sheme z uporabo UNION poizvedb (E. Caselli, 2020) 60
5.1.5 Zakasnjeni SQL injection (Follin, 2025) 61
5.1.6 Časovnica razvoja SQL injection ranljivosti 63
5.2 Novosti in pristopi v preprečevanju napadov v zadnjih 5 letih 66
5.3 Praktični primer varne implementacije 68
6. SKLEP 73
7. VIRI IN LITERATURA 75

KAZALO SLIK

Slika 1: Primer podatkovne strukture v podjetju za enega investitorja 15
Slika 2: Primer podatkovne strukture v podjetju za enega investitorja 2 16
Slika 3: Primer poročila za računovodjo 16
Slika 4: Primer poročila, pripravljenega za računovodjo 20
Slika 5: Batch Processing vzorec 21
Slika 6: Stream Processing vzorec 22
Slika 7: Lambda vzorec 22
Slika 8: Kappa vzorec 23
Slika 9: ETL vzorec 24
Slika 10: ELT vzorec 25
Slika 11: Data Mesh vzorec 25
Slika 12: Data Lakehouse vzorec 26
Slika 13: RAG vzorec 27

Slika 14: Primer celotnega vnosnega obrazca 29
Slika 15: Izbira datuma s tkcalendar 30
Slika 16: Izbira vrste artikla 30
Slika 17: Primer filtriranja artiklov glede na izbrano vrsto 31
Slika 18: Neustrezna količina 32
Slika 19: S pomočjo funkcije na standardizirano numerično vrednost popravljena količina .. 32
Slika 20: Spustni seznam investitorjev 33
Slika 21: Spustni seznam gradbišč 33
Slika 22: Zaklenjena gradnika Prevoz in Črpanje 34
Slika 23: Primer odklenjenega polja Prevoz 34
Slika 24: Izbirna polja za dodatke 35
Slika 25: Rezultat štoparice 36
Slika 26: Primer poročila za enega investitorja 36
Slika 27: Štoparica za generiranje poročil 37
Slika 28: Primer poročila pri avtomatizirani generaciji 38
Slika 29: Diagram relacij tabel v DataGripu 40
Slika 30: Rezultati štoparice pri PostgreSQL rešitvi 41
Slika 31: Primer enega poročila 42
Slika 32: Nadzorna plošča za dnevni pregled v Power BI 45
Slika 33: Nadzorna plošča za mesečni pregled v Power BI 46
Slika 34: Nadzorna plošča za dnevni pregled v programu Tableau 48
Slika 35: Nadzorna plošča za mesečni pregled v programu Tableau 49
Slika 36: Primer uporabniškega vnosa 57
Slika 37: Primer SQL injection za izbris baze 58
Slika 38: Primer prijave 58
Slika 39: Primer napadalčevega vnosa 59
Slika 40: Primer zakasnjenega SQL injection napada 62
Slika 41: Primer nepravilne implementacije 69
Slika 42: Primer pravilne implementacije 70
Slika 43: Primer parametrizirane poizvedbe 71
Slika 44: Primer nepravilne poizvedbe 71
Slika 45: Pravilna implementacija poizvedbe razvijalca 72

KAZALO TABEL

Tabela 1: Prednosti in slabosti Batch Processing cevovoda 21
Tabela 2: Prednosti in slabosti Streaming cevovoda 21
Tabela 3: Prednosti in slabosti Lambda cevovoda 22
Tabela 4: Prednosti in slabosti Kappa cevovoda 23
Tabela 5: Prednosti in slabosti ETL cevovoda 24
Tabela 6: Prednosti in slabosti ELT cevovoda 24
Tabela 7: Prednosti in slabosti Data Mesh cevovoda 25
Tabela 8: Prednosti in slabosti Data Lake cevovoda 26
Tabela 9: Prednosti in slabosti RAG cevovoda 26

1. UVOD

S porastom podatkov, ki se ustvarjajo vsak dan, je v porastu tudi njihovo upravljanje. Vsak dan se ustvari 402,74 milijona terabajtov podatkov (Duarte, 2025).

Ogromna podjetja in korporacije se dobro zavedajo pomembnosti podatkov in že dobro vpeljujejo načine, kako te podatke shranjevati, obdelovati, vizualizirati in zaščititi pred nepooblaščenimi dostopi. Tudi manjša podjetja menijo, da potrebujejo učinkovit sistem za spreminjanje podatkov v koristne informacije, ker le tako lahko ostajajo pred konkurenco. Velika verjetnost je, da imajo konkurenti že vpeljan nek sistem, a na vrhu bo ostal samo tisti, ki bo imel najučinkovitejšega.

Rečeno je, da so podatki nova nafta, kar nakazuje njihova vrednost na trgu. Osebno menim, da so podatki vredni veliko več kot nafta in se morda njihove prave vrednosti še ne zavedamo popolnoma. Nafta, zlato in ostale dobrine so na našem planetu omejeni in jih imamo toliko, kot jih imamo. Podatki pa so neomejeni in se generirajo vsako sekundo pravzaprav iz ničesar. Prava vrednost podatkov pa je v tem, kako jih lahko uporabimo. Surovi podatki sami po sebi nimajo velike vrednosti, a s pravimi pristopi jih podatkovni analitiki, podatkovni znanstveniki in podatkovni inženirji spreminjamo v koristne informacije tako za osebne kot poslovne namene. Podatke v koristne informacije na žalost spreminjajo tudi nepridipravi in hekerji, ki podatke kradejo, jih prodajajo naprej ali z različnimi metodami brišejo in spreminjajo našo podatkovno infrastrukturo, zato je zelo pomembno, da se znamo proti takšnim napadom ustrezno zaščititi.

Za diplomsko delo sem si izbral raziskovanje področja podatkovnega inženiringa, ker menim, da je vredno raziskati, kako podatke čimbolj pripraviti do tega, da brezhibno potujejo od vira do poročila, in razviti učinkovit sistem, ki to omogoča. Na voljo nam je kar nekaj orodij za grajenje ustreznega podatkovnega cevovoda, vendar se moramo ob tem vprašati, kakšne informacije sploh želimo pridobivati iz naših podatkov. Nesmiselno in predrago bi bilo oblikovati kompleksen sistem cevovoda, relacijskih baz podatkov itd. za npr. podjetnika, ki ga resnično zanima samo dobiček in nič drugega.

Krovna tema za leto 2024/2025 je kibernetska varnost, ki je zelo pomembna tema tudi, ko se pogovarjamo o občutljivih osebnih in poslovnih podatkih. Tudi če naš računalnik, ki hrani občutljive podatke, ni priklopljen na internet in se vse dogaja lokalno, se lahko pojavi fizična kraja računalnika, po kateri lahko nepooblaščena oseba dostopa do podatkov in jih zlorabi.

Vemo, da se v današnjem času na internetu vse dogaja na ravni raznih strežnikov in oblakov. Dejstvo je, da se moramo takoj, ko se priklopimo na internet**,** ustrezno zavarovati, zato bom to temo raziskal tudi v svojem diplomskem delu in jo skušal smiselno povezati z našim primerom.

1.1 Opis področja in opredelitev problema

Primarno se bom v svojem diplomskem delu ukvarjal predvsem z raziskovanjem in grajenjem najučinkovitejšega sistema za delo s podatki za potrebe manjšega gradbenega podjetja KG-EKO, d. o. o. Želim nadgraditi njihov sistem – od vnašanja naročil do generacije poročil, ki jih sedaj v celoti opravljajo v programu Microsoft Excel. Proces jim sicer ustreza, vendar je časovno potraten in podvržen napakam. Dokazoval bom, da je vpeljava podatkovnega cevovoda dobra izbira tudi za manjše gradbeno podjetje, pri čemer lahko domnevamo, da ga niti ne potrebujejo.

Podatkovni cevovod je sestavljen nabor tehnologij in postopkov, ki omogočajo zbiranje, obdelavo, shranjevanje in vizualizacijo podatkov iz različnih virov. Tak sistem običajno vključuje tudi analitična orodja, ki pridejo v poštev bolj v zaključnih fazah podatkovnega cevovoda in so namenjena samo npr. pregledovanju rezultatov poslovanja, ki se je zgodilo v nekem časovnem obdobju. Takšni orodji sta Tableau in Power BI, ki ju bom v nadaljevanju tudi podrobneje pregledal in testiral njuno uporabnost pri navigaciji in pregledu za končnega uporabnika.

Na koncu se bom ukvarjal še s kibernetsko varnostjo takšnega sistema. Glede na to, da bom zgradil sistem, prek katerega se bodo pretakali občutljivi podatki, moram razmisliti o obrambi pred napadi tipa SQL injection in ostalimi vrstami nepooblaščenih dostopov.

1.2 Namen, cilji in osnovne trditve

Namen diplomskega dela je proučiti in prikazati, kako lahko vpeljava podatkovnega cevovoda izboljša učinkovitost obdelave podatkov, avtomatizacijo poročanja in varnost podatkov v manjšem podjetju. Izhodišče dela predstavlja problem ročne obdelave podatkov v programu Microsoft Excel, ki se v podjetju KG-EKO, d. o. o. izkazuje kot časovno potratna, nagnjena k napakam in slabo prilagodljiva rastočemu obsegu podatkov. Namen dela je pokazati, da so sodobni pristopi podatkovnega inženiringa uporabni in smiselni tudi v manjših organizacijah ter da lahko prinesejo konkretne in merljive koristi v vsakodnevnem poslovanju.

Cilji diplomskega dela so:

- analizirati obstoječi način obdelave in poročanja podatkov v podjetju KG-EKO, d. o. o.,
- zasnovati in implementirati podatkovni cevovod po vzorcu ETL z uporabo programskega jezika Python,
- primerjati ročno obdelavo podatkov v Excelu z avtomatizirano obdelavo s pomočjo knjižnice Pandas in relacijske podatkovne baze PostgreSQL,
- ovrednotiti časovno učinkovitost avtomatiziranega generiranja poročil,
- primerjati vizualizacijski orodji Microsoft Power BI in Tableau z vidika uporabniške izkušnje v manjšem podjetju,
- analizirati varnostne vidike pregleda, obdelave in spreminjanja podatkov s poudarkom na SQL injection ranljivostih,
- preveriti, ali so se v zadnjih petih letih pojavili novi pristopi za preprečevanje SQL injection napadov.

Osnovne trditve (hipoteze), ki jih diplomsko delo preverja, so:

- **H1:** Vpeljava podatkovnega cevovoda skrajša čas izdelave poročila za vsaj 90 % v primerjavi z ročnim postopkom v Microsoft Excel.
- **H2:** Pandas DataFrame omogoča hitrejšo obdelavo podatkov iz podatkovne baze v Excelu v manjših projektih kot poizvedovanje s SQL v PostgreSQL relacijski podatkovni bazi.
- **H3:** Microsoft Power BI je pri pregledu rezultatov poslovanja za končnega uporabnika primernejše orodje kot Tableau.
- **H4:** V zadnjih petih letih ni novih pristopov za preprečitev napadov tipa SQL injection.

1.3 Predpostavke in omejitve

Ena izmed predpostavk diplomskega dela je, da obravnavani poslovni procesi in podatki reprezentativno odražajo dejansko stanje v manjšem podjetju. Čeprav so bili pri praktičnih primerih uporabljeni testni podatki, ti še vedno razkrivajo način dela podjetja KG-EKO, d. o. o., zato direktor podjetja ne dovoljuje javne objave diplomskega dela, saj bi lahko razkritje strukture podatkov in procesov predstavljalo poslovno tveganje.

Na področju vizualizacije podatkov predstavlja omejitev tudi uporaba orodja Tableau, ki je plačljivo. Avtor je imel dostop do orodja v okviru 13-dnevne preizkusne licence, kar je omejilo čas za podrobnejše testiranje naprednejših funkcionalnosti. Kljub temu je bil čas dostopa

zadosten za izvedbo primerjave z orodjem Microsoft Power BI in za preverjanje zastavljene hipoteze.

Omejitev predstavlja tudi obseg izvedenih intervjujev. Zaradi ciljev raziskave je bil intervju izveden z avtorjem diplomskega dela in direktorjem podjetja. Takšen izbor sogovornikov je bil ocenjen kot ustrezen, saj direktor predstavlja končnega uporabnika poročil in glavnega odločevalca, kar je zadostovalo za preverjanje uporabnosti rešitve in potrjevanje hipoteze.

1.4 Uporabljene raziskovalne metode

Pri izdelavi diplomskega dela je bilo uporabljenih več raziskovalnih metod, ki so omogočile celovito obravnavo raziskovalnega problema ter povezovanje teoretičnih izhodišč s praktično implementacijo.

Prva metoda je bila uporaba osebnih zapiskov, ki so nastajali ob razvoju rešitve in študiju podatkovnega inženiringa. Zapiski so zajemali tehnične ugotovitve, opažanja o obnašanju podatkov in izkušnje pri implementaciji podatkovnega cevovoda. Ta metoda je omogočila refleksijo razvojnega procesa in sistematično primerjavo različnih pristopov obdelave podatkov.

Druga raziskovalna metoda je bila tudi analiza podatkov, pri kateri so bili podatki obdelani z uporabo različnih orodij in tehnologij, vključno z Microsoft Excelom, knjižnico Pandas in relacijsko podatkovno bazo PostgreSQL. S pomočjo te metode je bila izvedena primerjava časovne učinkovitosti in uporabnosti posameznih pristopov, kar je omogočilo preverjanje zastavljenih hipotez.

Tretja uporabljena metoda je bil intervju, ki je bil izveden z direktorjem podjetja KG-EKO, d. o. o. Namen intervjuja je bil pridobiti povratne informacije o dveh razvitih nadzornih ploščah in oceniti njuno uporabnost z vidika končnega uporabnika. Intervju je bil osredotočen predvsem na preglednost prikazanih podatkov, razumljivost vizualizacij in splošno uporabniško izkušnjo.

Poleg tega je bila uporabljena analiza učnega in strokovnega gradiva, ki je zajemala pregled literature s področja podatkovnih cevovodov, ETL procesov, vizualizacije podatkov in SQL injection ranljivosti. Pregled gradiva je omogočil teoretično utemeljitev raziskovalnega problema.

2. SMISELNOST UVEDBE PODATKOVNEGA CEVOVODA V MANJŠA PODJETJA

Implementacija podatkovnega cevovoda zahteva jasno opredeljene faze in premišljeno zaporedje korakov. Njegov osnovni namen je zagotoviti zanesljiv in ponovljiv pretok podatkov od vira do končnega poročila. Učinkovit podatkovni cevovod je sestavljen iz faz zajema podatkov, njihovega čiščenja in transformacije ter končnega nalaganja v ciljno podatkovno strukturo, v kateri so podatki pripravljeni za analizo in poročanje. Poseben poudarek je na zagotavljanju kakovosti podatkov že v zgodnjih fazah cevovoda, saj se tako prepreči širjenje napak v nadaljnjih korakih obdelave. Avtomatizacija teh procesov omogoča večjo zanesljivost, manj ročnega dela in lažje prilagajanje rastočim količinam podatkov, kar je še posebej pomembno v okoljih, v katerih se podatki redno posodabljajo in uporabljajo za operativne ter analitične namene (Matillion, 2025).

2.1 Časovna primerjava med Excelom in avtomatizirano obdelavo

V podjetju KG-EKO, d. o. o. trenutno uporabljajo Excel na način, ki je v praksi zelo zamuden in neučinkovit. Namesto enotnega stolpca *Artikel*, v katerega bi se vpisal naziv artikla, ima Excelova datoteka dejanske stolpce poimenovane po posameznih artiklih. To pomeni, da je vsak artikel predstavljen kot svoj lasten stolpec v tabeli, medtem ko se količine ali druge vrednosti vpisujejo tako, da zaposleni najdejo pravilen stolpec, v katerega ročno vnesejo podatek.

Takšen pristop je časovno potraten, ker mora zaposleni pri vsakem vnosu najprej vizualno poiskati ustrezen stolpec med množico artiklov. Poleg tega je takšen sistem podvržen napakam – napačen klik v napačen stolpec lahko pomeni napačno obračunane stroške ali napačno prikazana poročila. Ta oblika Excel baze tudi onemogoča normalizacijo podatkov, avtomatsko obdelavo in hitro poizvedovanje, kar se kaže v dodatnih ročnih korakih, ki jih direktor opravlja pri pripravi mesečnih poročil.

V diplomskem delu bom takšen dejanski primer iz podjetja tudi simuliral: v Excelu bom pripravil bazo testnih podatkov z identično strukturo, kot jo uporabljajo v praksi, z vsemi stolpci in obstoječim načinom vnosa. Namen tega dela je prikazati, kako neustrezna podatkovna struktura podaljša čas dela, otežuje avtomatizacijo ter jasno izraža potrebo po uvedbi podatkovnega cevovoda in pravilno zasnovane podatkovne baze.

Dodatno bom za namen analize pripravil tudi testno Excel bazo z enako strukturo, kot jo uporablja podjetje v praksi. V njej bom ustvaril 10 vrstic podatkov za enega investitorja. Na podlagi tako strukturiranih podatkov bom nato ročno izračunal vse potrebne vrednosti za mesečno poročilo, enako kot to trenutno dela direktor podjetja. Tako bom prikazal dejansko časovno zahtevnost, kompleksnost in možnost napak pri ročni obdelavi takšne oblike podatkov ter vzpostavil osnovo za kasnejšo avtomatizirano obdelavo.

	A	B	C	D	E	F	G	H	I	J	K
	Datum	Vrsta	Beton,...	Količina	Investitor	Gradbišče	XQ/15/94	QX/57/82	YM/88/39	XQ/94/3	XY/75/4
	15.11.25	Beton	XQ/94/3 PREVOZ	15,30	Perko d.d.	Trbovlje	/	/	/	15,30	/
	15.11.25	Beton	XQ/94/3 + prevoz 50	7,60	Perko d.d	trbovlje	/	/	/	7,60	/
	16.11.2025	Beton	XQ/94/3 + velika črpanje	40,10	Perko d.d.	Živinozdravska ulica 25K	/	/	/	40,10	/
	16.11.2025	Beton	XQ/94/3 + kakovost,črpanje sr.	11,90	Perko d.d.	Živinozdravska ulica 25K	/	/	/	11,90	/
	16.11.2025	Peskokop	Mivka	40,00	Perko	Živinozdravska ulica 25K	/	/	/	/	/
	16.11.2025	Peskokop	Mivka	50,90	PERKO d.d.	Živinozdravska	/	/	/	/	/
	17.11.2025	Peskokop	Mivka	30,60	Perko d.d.	Živinozdravska Ulica 25	/	/	/	/	/
	17.11.2025	Peskokop	Mivka	15,50	Perko d.d.	Živinozdravska ulica 25K	/	/	/	/	/
	17.11.2025	Peskokop	Mivka	55,00	Perko d.d.	Živinozdravska ulica 25K	/	/	/	/	/

Slika 1: Primer podatkovne strukture v podjetju za enega investitorja

Vir: lastni vir

Slika prikazuje dejansko strukturo podatkov, ki jo prejme direktor. Takšna baza je uporabna izključno za ročno delo človeka, saj računalniško obdelavo praktično onemogoča. Direktor sicer dobro ve, da gre povsod za istega investitorja in isto gradbišče, vendar je zaradi velike nekonsistentnosti vnosov podatkov prisiljen poročila sestavljati ročno.

V podatkih je jasno vidno, da so datumi zapisani v različnih formatih, pri artiklih pa zaposleni pogosto dopisujejo dodatne opise (npr. "PREVOZ", "velika črpanje", "prevoz 50" …). Za človeka so to smiselne opombe, za računalnik pa popolnoma različni vnosi, ki se ne morejo združevati ali filtrirati.

Enako velja za investitorja in gradbišča. Čeprav gre v vseh primerih za isto podjetje in isto lokacijo, se pojavljajo različice zapisa, kot so "*Perko D.D.*" in "*Perko D.D*" ali zapis gradbišča po kraju in ulici. Direktorju, ki na pamet pozna vse ulice in mesta gradbišč, kjer poslujejo, to ne predstavlja težav. Računalniku pa to predstavlja popolnoma ločen podatek.

Zaradi takšnih odstopanj Excel datoteka še ni primerna za avtomatsko obdelavo ali pripravo poročil – podatki se ne ujemajo in jih ni mogoče pravilno agregirati, zato je trenutni sistem uporaben le za ročni proces, pri katerem direktor podatke prebere, interpretira in prepiše v poročilo, kar pa je časovno potratno in podvrženo napakam.

X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL
Odpadna zemlja	Odpadni asfalt	Mivka	Obsipni	Tampon mali	Tampon srednji	Tampon veliki	Pesek mali	Pesek veliki	Betonski material	Prevoz	Črpanje M	Črpanje S	Črpanje V	
/	/	/	/	/	/	/	/	/	/	/	/	/	/	
/	/	/	/	/	/	/	/	/	/	Prevoz do 50KM (do 50m3)	/	/	/	
/	/	/	/	/	/	/	/	/	/	/	/	/	40,10	
/	/	/	/	/	/	/	/	/	/	/	/	11,90	/	
/	/	40,00	/	/	/	/	/	/	/	/	/	/	/	
/	/	50,90	/	/	/	/	/	/	/	/	/	/	/	
/	/	30,60	/	/	/	/	/	/	/	/	/	/	/	
/	/	15,50	/	/	/	/	/	/	/	/	/	/	/	
/	/	55,00	/	/	/	/	/	/	/	/	/	/	/	

Slika 2: Primer podatkovne strukture v podjetju za enega investitorja 2

Vir: lastni vir

Na drugem primeru baze je težava še očitnejša. Vsaka entiteta v tabeli ima vse možne stolpce, saj so stolpci poimenovani po vseh artiklih, ki jih podjetje ponuja. To pomeni, da vsaka vrstica vsebuje desetine ali celo stotine praznih stolpcev ne glede na to, kateri artikel je bil v resnici uporabljen. Ker so ti stolpci razporejeni daleč v desno, se mora direktor vsakič premikati čez celotno širino baze in iskati pravi stolpec. To močno povečuje možnost napak. Hitro se lahko vnese količina v napačen stolpec ali se ustrezen stolpec spregleda.

TESTNI INVESTITOR

TESTNO GRADBIŠČE 1

Šifra	Artikel	Količina	Cena / m3	Skupna cena
1000	Artikel 1	50,00	90,00 €	4.500,00 €
1001	Artikel 2	90,00	60,00 €	5.400,00 €
1002	Artikel 3	100,00	40,00 €	4.000,00 €
1003	Artikel 4	30,00	20,00 €	600,00 €
1004	Artikel 5	40,00	100,00 €	4.000,00 €
			Skupaj	18.500,00 €

TESTNO GRADBIŠČE 2

Šifra	Artikel	Količina	Cena / m3	Skupna cena
1000	Artikel 1	130,30	10,00 €	1.303,00 €
1001	Artikel 2	50,40	20,00 €	1.008,00 €
1002	Artikel 3	40,10	50,00 €	2.005,00 €
			Skupaj	4.316,00 €

SKUPAJ VSA GRADBIŠČA	**22.816,00 €**

Slika 3: Primer poročila za računovodjo

Vir: lastni vir

Na prikazani sliki je razvidno, kakšno poročilo mora direktor pripraviti za vsakega investitorja za izbrano obdobje. Glavni cilj direktorja je, da za posameznega investitorja dobi pregled vseh

gradbišč v enem mesecu, pri čemer morajo biti agregirane vse količine za isti artikel, uporabljena mora biti enotna cena iz cenika, izračunana skupna vrednost po posameznem gradbišču, dodana mora biti šifra za vsak artikel in na koncu še seštevek cene za vsa gradbišča skupaj. Direktor je zaradi strukture baze prisiljen vse količine za isti artikel ročno poiskati, sešteti in prenesti v končno poročilo. Prav tako mora ročno preveriti ceno po ceniku in izračunati skupno vrednost, pri čemer mora ločeno obravnavati vsak artikel na vsakem gradbišču. Ker so v vhodni Excel datoteki stolpci poimenovani po vseh možnih artiklih, mora direktor prebirati pripise, ročno iskati ustrezne stolpce in vnesti pravilne količine.

Vse to povzroči, da je izdelava enega samega poročila časovno zelo obsežna, nepregledna in izjemno dovzetna za človeške napake, avtomatska obdelava pa zaradi nekonsistentnosti podatkov sploh ni mogoča. Trenutni proces predstavlja velik izziv in jasno kaže na potrebo po uvedbi strukturirane baze podatkov in podatkovnega cevovoda.

2.2 Analiza stroškovne učinkovitosti

V podjetju imajo zaposleni vzpostavljen osnovni Excel makro, ki omogoča hitrejši vnos podatkov, vendar ta rešitev ne odpravlja največjih težav. Datum, investitorja in gradbišče morajo zaposleni še vedno vpisovati ročno, kar pomeni, da se napake pojavljajo že pri osnovnih podatkovnih poljih (različni zapisi imen, različni formati datumov, različni zapisi lokacij). Spustni seznam za artikle obstaja, vendar praksa kaže, da občasno zaposleni uporabljajo artikel, ki ga še ni na ceniku, zato ga preprosto vpišejo ročno, kar povzroča dodatno nekonsistentnost.

Z uporabo ustreznega vnosnega obrazca bi bilo mogoče človeško napako skoraj popolnoma izničiti. Obrazec bi imel elemente, ki prisilijo pravilnost vnosa:

- koledar za izbor datuma (ni več napačnih formatov);
- spustni seznam za artikle, na katerem so vsi artikli enotni ob konsistentnem posodabljanju cenika;
- spustni seznam investitorjev in gradbišč brez možnosti vpisovanja po lastni presoji;
- dodajanje novega investitorja ali gradbišča prek ločenega obrazca, ki ga bo pravilno shranil v bazo;
- obvezna validacija vnosa (npr. količina mora biti številka, artikel mora obstajati, gradbišče mora biti povezano z investitorjem itd.).

Takšen pristop odpravi ročne vnosne napake, omogoča stoodstotno konsistentnost podatkov in zmanjša čas vnosa, poleg tega pa omogoča, da nadaljnja transformacija podatkov v podatkovnem cevovodu poteka hitro, pravilno in popolnoma avtomatizirano.

Napake pri ročnem vnosu podatkov predstavljajo enega glavnih virov neučinkovitosti v poslovnih procesih, saj se njihovi učinki ne omejujejo samo na napačne zapise, ampak se prenašajo skozi celoten informacijski sistem in vplivajo na poročanje, odločanje ter stroške poslovanja. Tudi majhna stopnja napak pri večjih količinah podatkov pomeni veliko absolutno število napačnih vnosov, kar povzroča dodatno delo, popravke in zamude. Članek še navaja, da se zaradi tega vse pogosteje poudarja doseganje čim višje, idealno popolne točnosti vnosa podatkov, kar je z ročnimi postopki težko dosegljivo. Uporaba avtomatiziranih in nadzorovanih vnosnih mehanizmov omogoča večjo konsistentnost zapisov, zmanjšuje potrebo po kasnejših ročnih popravkih ter povečuje zanesljivost podatkov, ki se uporabljajo v nadaljnjih fazah obdelave in analize (Smith, 2020).

Za potrebe analize bom izvedel tudi praktični preizkus ročne izdelave poročila. Uporabil bom testno Excel bazo z 10 vrsticami podatkov za enega investitorja, strukturirano enako kot dejanska baza v podjetju. Pri izdelavi poročila bom sledil enakemu postopku, kot ga uporablja direktor, in dokumentiral vse korake, ki jih mora opraviti: pregled pripisov pri artiklih, iskanje ustreznih stolpcev, ročno seštevanje količin, izbor cen s cenika in ročno dodeljevanje šifer artiklom.

Za celoten proces bom zagnal štoparico in izmeril dejanski čas izdelave ročnega poročila. S tem bom pridobil realen vpogled v časovno zahtevnost trenutnega sistema, tudi kadar je obseg podatkov minimalen. Rezultat tega preizkusa bo služil kot referenčna osnova za primerjavo z avtomatizirano obdelavo podatkov v podatkovnem cevovodu, v katerem bom enako poročilo ustvaril s pomočjo Pythona in knjižnice FPDF v popolnoma avtomatizirani obliki.

Pri ročnem postopku sem izmeril naslednje časovne in procesne komponente:

1. **Čas priprave okolja** (14 sekund)

- odprtje Excel datoteke z vhodnimi podatki
- nastavitev filtrov po investitorju in obdobju

2. **Čas pregleda podatkov** (1 minuta in 20 sekund)

- pregled vseh vrstic in identifikacija relevantnih zapisov
- preverjanje pripisov pri artiklih
- preverjanje gradbišč in njihove pravilnosti

3. **Čas iskanja struktur podatkov** (50 sekund)

- določanje števila gradbišč
- določanje števila artiklov
- prepoznavanje podvojenih in neenotnih zapisov
- določanje pravilnih podatkovnih tipov stolpcev, v katerih se nahajajo številke.

4. **Čas priprave poročila** (2 minuti in 16 sekund)

- ustvarjanje glave poročila
- ustvarjanje praznih tabel za gradbišča
- dodajanje naslovov in stolpcev

5. **Čas ročnega prepisovanja podatkov** (4 minute in 5 sekund)

- iskanje ustreznih stolpcev v vhodni tabeli (premikanje levo–desno)
- prepisovanje količin v poročilo
- seštevanje količin za iste artikle

6. **Čas pridobivanja cen** (1 minuta in 11 sekund)

- odprtje Excel cenika
- iskanje cen za uporabljene artikle
- vnos cen v poročilo

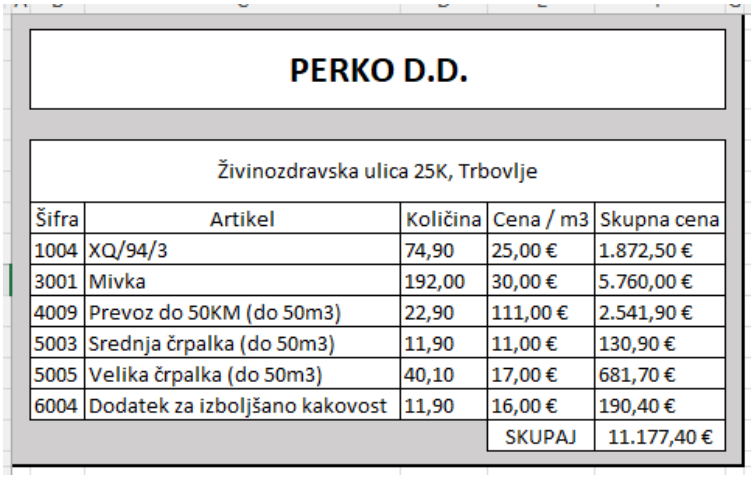
7. **Čas izračunov** (40 sekund)

- izračun skupne cene na artikel (količina × cena)
- izračun skupne vrednosti na gradbišče
- izračun končne, kumulativne vrednosti za investitorja

8. **Čas preverjanja pravilnosti** (36 sekund)

- ponovni pregled vseh količin
- preverjanje pravilnosti cen
- pregled končnih izračunov

Skupni čas ročnega postopka za 10 vrstic, za enega investitorja in eno gradbišče je torej **11 minut in 12 sekund**. To je razmeroma majhen in preprost primer, ki ne predstavlja dejanske kompleksnosti poslovanja podjetja.



PERKO D.D.

Živinozdravska ulica 25K, Trbovlje

Šifra	Artikel	Količina	Cena / m3	Skupna cena
1004	XQ/94/3	74,90	25,00 €	1.872,50 €
3001	Mivka	192,00	30,00 €	5.760,00 €
4009	Prevoz do 50KM (do 50m3)	22,90	111,00 €	2.541,90 €
5003	Srednja črpalka (do 50m3)	11,90	11,00 €	130,90 €
5005	Velika črpalka (do 50m3)	40,10	17,00 €	681,70 €
6004	Dodatek za izboljšano kakovost	11,90	16,00 €	190,40 €
			SKUPAJ	11.177,40 €

Slika 4: Primer poročila, pripravljenega za računovodjo

Vir: lastni vir

V realnem poslovnem procesu se direktor sooča z veliko večjim obsegom podatkov. Po njegovih besedah ima podjetje mesečno približno 90 različnih investitorjev, skupno okoli 800 vrstic podatkov, in približno 180 različnih gradbišč, ki se pojavljajo v istem obdobju. Zaradi nekonsistentnosti podatkov, ročnega prepisovanja, iskanja pravih stolpcev, interpretacije pripisov in ročnih izračunov se obseg dela eksponentno poveča.

Direktor navaja, da za pripravo vseh mesečnih poročil porabi približno en teden dela vsak mesec, kar predstavlja velik časovni pritisk in jasno kaže na neučinkovitost obstoječega sistema. Ta primer potrjuje, da ročni način izdelave poročil ni primeren za večje količine podatkov in da lahko uvedba podatkovnega cevovoda prinese ogromne prihranke časa ter višjo natančnost in kakovost rezultatov.

2.3 Pregled modernih implementacij podatkovnih cevovodov

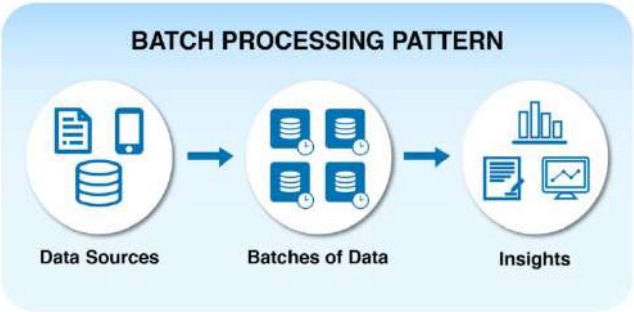
2.3.1 Batch Processing Pipeline (MacDonald, 2024)

Batch obdelava je najstarejši in najpogosteje uporabljen vzorec. Podatki se najprej zberejo v večji "paket" (npr. enkrat dnevno, tedensko ali mesečno) in se nato procesirajo vsi naenkrat. Uporablja se za obračune, poročila, računovodske izpise in večje količine podatkov, ki ne zahtevajo sprotne obdelave.

Tabela 1: Prednosti in slabosti Batch Processing cevovoda

PREDNOSTI	SLABOSTI
enostaven za implementacijo	Podatki niso ažurni, ker jih obdelujemo periodično.
"poceni" za procesiranje	neprimeren za "real-time" procese

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 5: Batch Processing vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/data-pipeline-design-batch-processing-1024x576.png

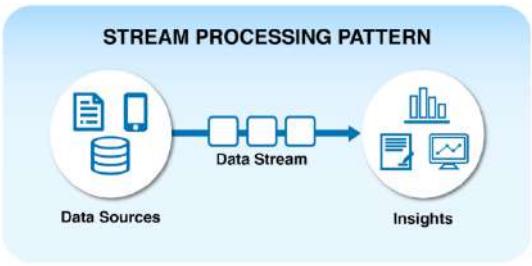
2.3.2 Streaming Pipeline (MacDonald, 2024)

Streaming Pipeline obdeluje podatke sproti, takoj ko prispejo v sistem. Zasnovan je za scenarije, v katerih je pomembno, da so informacije na voljo v realnem času. Uporablja se za monitoring sistemov, IOT senzorje in finančne transakcije v realnem času

Tabela 2: Prednosti in slabosti Streaming cevovoda

PREDNOSTI	SLABOSTI
minimalna zakasnitev	kompleksnejša infrastruktura (Kafka, Spark, Streaming)
takojšnja analiza	Zahteva stalno delovanje sistema.

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 6: Stream Processing vzorec

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/

2.3.3 Lambda Architecture (MacDonald, 2024)

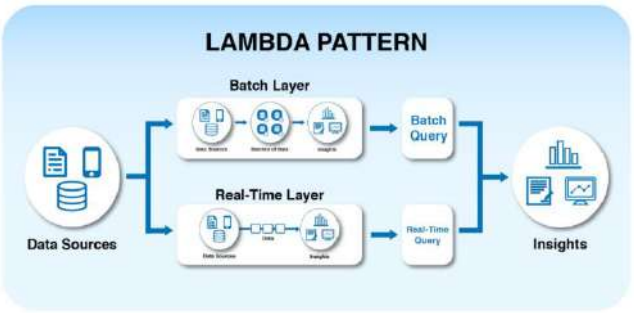
Lambda arhitektura združuje Batch in Streaming cevovod. Ena veja obdeluje podatke v realnem času, druga pa periodično zagotavlja točnost in celovitost podatkov. Sestavljen je iz treh plasti:

- speed layer – hitri, sprotni podatki,
- batch layer – počasni, popolni podatki,
- serving layer – združevanje obeh pogledov.

Tabela 3: Prednosti in slabosti Lambda cevovoda

PREDNOSTI	SLABOSTI
kombinacija hitrosti in natančnosti	podvajanje logike (batch + stream)
Primerna za analitiko, ki potrebuje sprotne podatke.	višja kompleksnost

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 7: Lambda vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/data-pipeline-design-lambda-pattern-1024x576.png

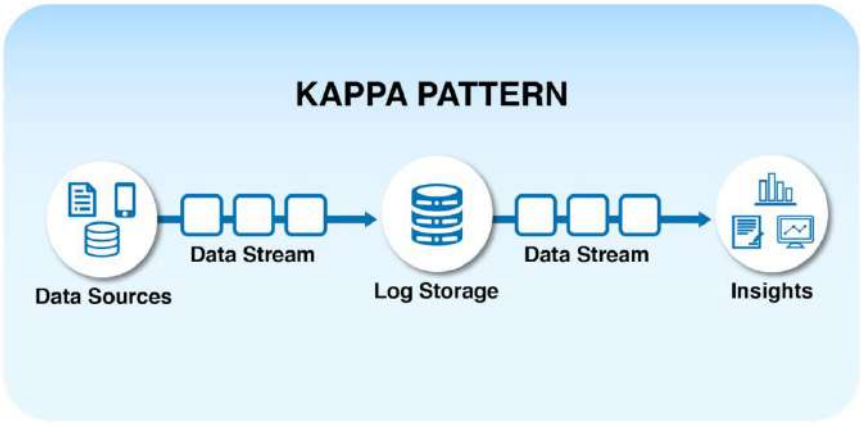
2.3.4 Kappa Architecture (MacDonald, 2024)

Kappa arhitektura je poenostavljena različica LAMBDA. Uporablja le Streaming Pipeline, vendar omogoča ponovno procesiranje zgodovinskih podatkov.

Tabela 4: Prednosti in slabosti Kappa cevovoda

PREDNOSTI	SLABOSTI
enostavnejša kot Lambda (ni batch plasti)	Ni idealna za podatke, ki ne prihajajo sproti.
idealna za sisteme z visokim tokom podatkov	Potrebuje streaming infrastrukturo.

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 8: Kappa vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/data-pipeline-design-kappa-1024x576.png

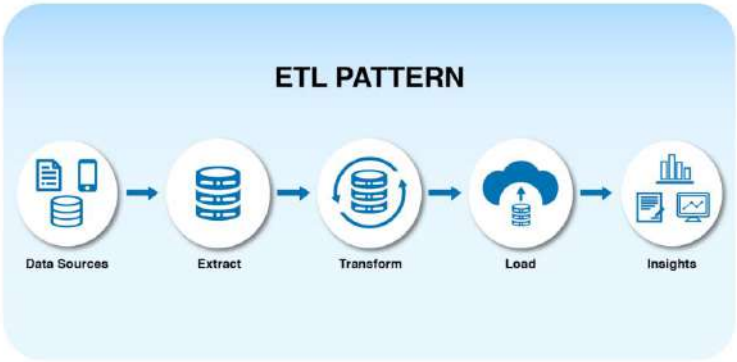
2.3.5 ETL (Extract, Transform, Load) (MacDonald, 2024)

Klasični vzorec, pri katerem se podatki najprej pridobijo, nato preoblikujejo in šele nato naložijo v bazo ali skladišče. Uporablja se pri aplikacijah z ročnim vnosom podatkov.

Tabela 5: Prednosti in slabosti ETL cevovoda

PREDNOSTI	SLABOSTI
V bazi so že čisti podatki.	Transformacija pred shranjevanjem lahko doda zakasnitev.
Preprečuje napake pri analitiki.	Zahtevana je kompleksna logika prečiščevanja podatkov pred shranjevanjem v bazo.

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 9: ETL vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/data-pipeline-design-ETL-1024x576.png

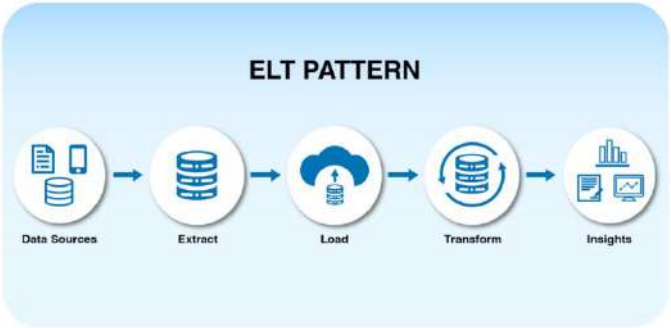
2.3.6 ELT (Extract, Load, Transform) (MacDonald, 2024)

Podatki se najprej naložijo v bazo v surovi obliki, transformacija pa sledi kasneje. Pogost v modernih oblačnih sistemih.

Tabela 6: Prednosti in slabosti ELT cevovoda

PREDNOSTI	SLABOSTI
zelo hitro nalaganje podatkov	Surovi podatki v bazi, kar pomeni slabo kvaliteto.
Izkorišča moč podatkovnih skladišč.	ni idealno za neenotne podatke ali ročne vnose

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 10: ELT vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/data-pipeline-design-ELT-1024x576.png

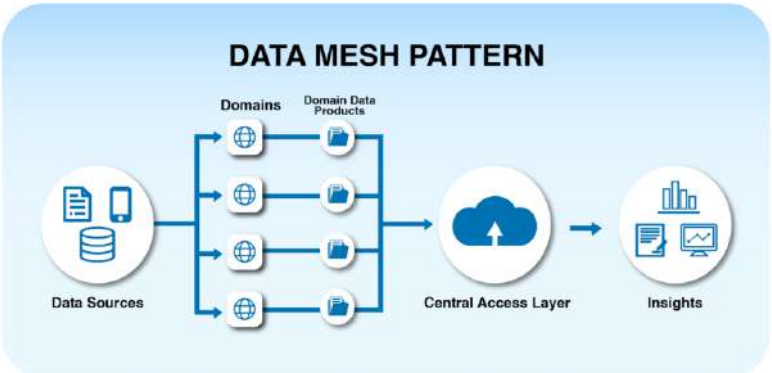
2.3.7 Data Mesh Pattern (MacDonald, 2024)

Cevovod, zasnovan na decentraliziranem upravljanju podatkov, pri katerem so podatki razdeljeni na domene (npr. finance, prodaja, operativa). Vsaka domena sama upravlja svoje podatkovne tokove, svoje modele in svojo kakovost podatkov.

Tabela 7: Prednosti in slabosti Data Mesh cevovoda

PREDNOSTI	SLABOSTI
visoka skalabilnost v velikih organizacijah	zelo kompleksna implementacija
jasne odgovornosti po domenah	Zahteva več ekip in stroge procese.

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 11: Data Mesh vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/data-pipeline-design-data-mesh-1024x576.png

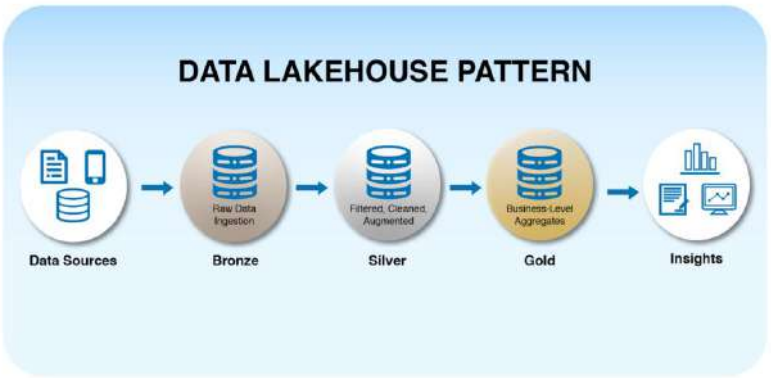
2.3.8 Data Lake Pattern (MacDonald, 2024)

Centralizirana shramba, namenjena hranjenju vseh vrst podatkov (strukturiranih, polstrukturiranih in nestrukturiranih) v surovi obliki. Transformacije se izvajajo šele ob branju ("schema-on-read").

Tabela 8: Prednosti in slabosti Data Lake cevovoda

PREDNOSTI	SLABOSTI
zelo primeren za velike količine podatkov	kompleksno vzdrževanje
velika fleksibilnost za podatkovne znanstvenike	Hitro se lahko spremeni v "data swamp", kar pomeni, da se podatki nalagajo brez pravil, ni validacij, standardov poimenovanja itd.

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 12: Data Lakehouse vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/data-pipeline-design-data-lakehouse-1024x576.png

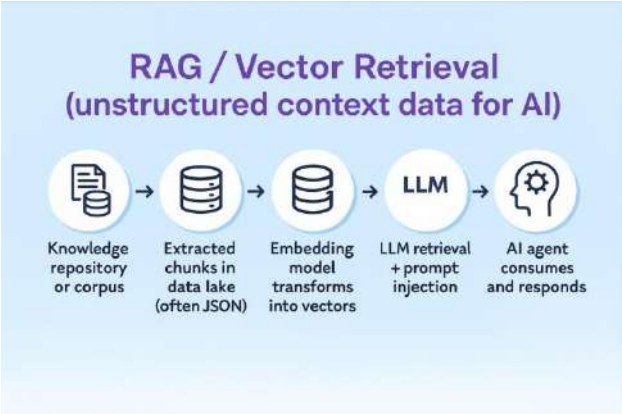
2.3.9 RAG Pattern (MacDonald, 2024)

Vzorec za delo z nestrukturiranimi podatki, pri katerem se dokumenti ali besedila pretvorijo v vektorske zapise in nato uporabljajo skupaj z AI modeli za iskanje in generiranje odgovorov.

Tabela 9: Prednosti in slabosti RAG cevovoda

PREDNOSTI	SLABOSTI
Omogoča uporabo dokumentov znotraj AI sistemov.	Zahteva vektorsko bazo in embedder model.
Izboljša natančnost generiranih odgovorov.	Ni primeren za klasično analitiko ali strukturirane podatke.

Vir: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/



Slika 13: RAG vzorec

Vir: https://montecarlodata.com/wp-content/uploads/2024/11/RAG-retrieval-pipeline-1024x683.png

Kandidata, med katerima izbiram in sta arhitekturno najbolj smiselna, sta ETL in ELT. Glavna razlika med njima je, kot sem omenil že zgoraj, v zaporedju transformacije in shranjevanja podatkov.

Pri ETL se podatki najprej pridobijo, nato očistijo, preoblikujejo in standardizirajo, šele na koncu pa se naložijo v ciljno podatkovno bazo ali skladišče, kar pomeni, da so podatki ob shranjevanju že kakovostni in pripravljeni za uporabo. Nasprotno se pri ELT podatki najprej naložijo v ciljni sistem v surovi obliki, transformacije pa se izvajajo naknadno z uporabo podatkovnega skladišča. ETL je zato primernejši za okolja, v katerih je kakovost podatkov pomembna že ob vnosu in v katerih so viri podatkov nagnjeni k napakam, medtem ko je ELT bolje prilagojen večjim količinam podatkov ter fleksibilni analitiki in bolje izkorišča moč podatkovnih skladišč. ELT omogoča hitrejši zajem podatkov in večjo prilagodljivost pri kasnejših transformacijah, vendar to hkrati pomeni, da se nekakovostni podatki začasno shranjujejo v sistemu, kar lahko poveča kompleksnost upravljanja podatkov. ETL s transformacijo pred shranjevanjem zmanjšuje tveganje širjenja napak in zagotavlja bolj

nadzorovan podatkovni tok, vendar lahko zahteva več procesnih virov v fazi obdelave pred nalaganjem (Bartley, 2025).

Za obravnavani primer podjetja KG-EKO, d. o. o. bo potem najprimernejši vzorec podatkovnega cevovoda ETL (Extract, Transform, Load). To je pristop, ki na našem primeru omogoča, da se podatki očistijo in standardizirajo, še preden se dokončno shranijo v Excel, iz katerega se nato generirajo poročila. To je pomembno v okolju, v katerem so podatki vneseni ročno in v katerem obstoječa Excel struktura vsebuje veliko nekonsistentnih vnosov.

Prva faza – extract (pridobivanje podatkov)

V tej fazi podatke vnese zaposleni preko kontroliranega vnosnega obrazca. Obrazec vsebuje spustne sezname za artikle, investitorje in gradbišča, koledar za datum ter validacijo številsko omejenih vnosov. S tem se že pri izvoru preprečijo napačni in neenotni vnosi, ki so značilni za obstoječi ročni sistem.

Druga faza – transform (transformacija podatkov)

Transformacija poteka že ob oddaji obrazca, kar pomeni:

- standardizacijo zapisov (enotno ime investitorja, gradbišča, artikla),
- odpravo pripisov pri artiklih (npr. “prevoz”, “črpanje”), ker aplikacija vse dodatke obravnava ločeno,
- validacijo datuma in količin,
- dodajanje kod ali internih oznak artiklov,
- zagotavljanje konsistentne strukture podatkov.

Posebnost tega primera je, da transformacija nadomesti vse ročne popravke, ki jih sedaj opravi direktor, saj se podatki “očistijo”, še preden končajo v Excelu.

Tretja faza – load (shranjevanje)

Končni, prečiščeni in standardizirani podatki se zapišejo v strukturirano Excel datoteko, ki predstavlja:

- enoten vir resnice ("single source of truth"),
- podatkovno osnovo za nadaljnjo obdelavo v Pythonu,
- bazo za avtomatsko generiranje poročil.

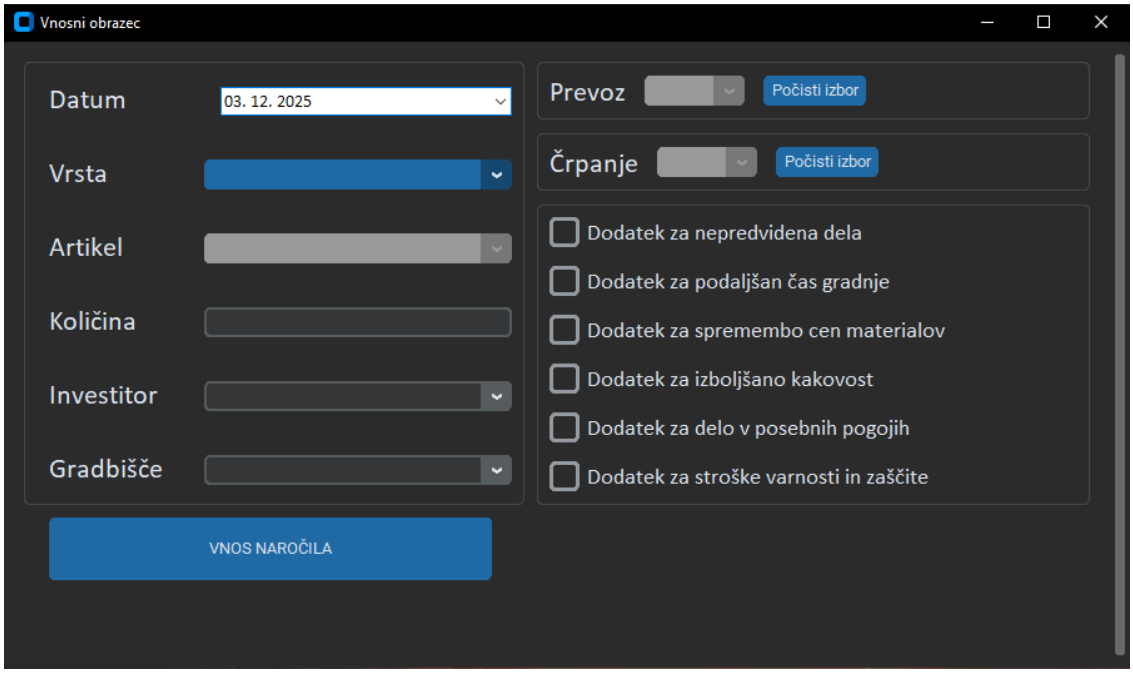
Ker so podatki že vnaprej pravilni, Excel služi le kot shramba in izvoz, ne pa kot orodje za ročno popravljanje napak.

Sklep, zakaj je ETL najprimernejši vzorec za naš primer:

- Podatki se vnašajo ročno, zato je nujno, da se preverijo in očistijo že ob vnosu.
- Obstoječa Excel baza povzroča velike težave zaradi neenotnih zapisov. ETL to v celoti odpravi.
- Excel ostane primarno orodje, zato ni potrebe po uvajanju kompleksnih arhitektur (Data Mesh, Data Lake, Lambda, Kappa).
- Transformacija ob vnosu zagotavlja, da se poročila ustvarjajo na popolnoma konsistentnih podatkih.
- Proces omogoča avtomatsko generiranje poročil, kar zmanjša čas in možnost napak.

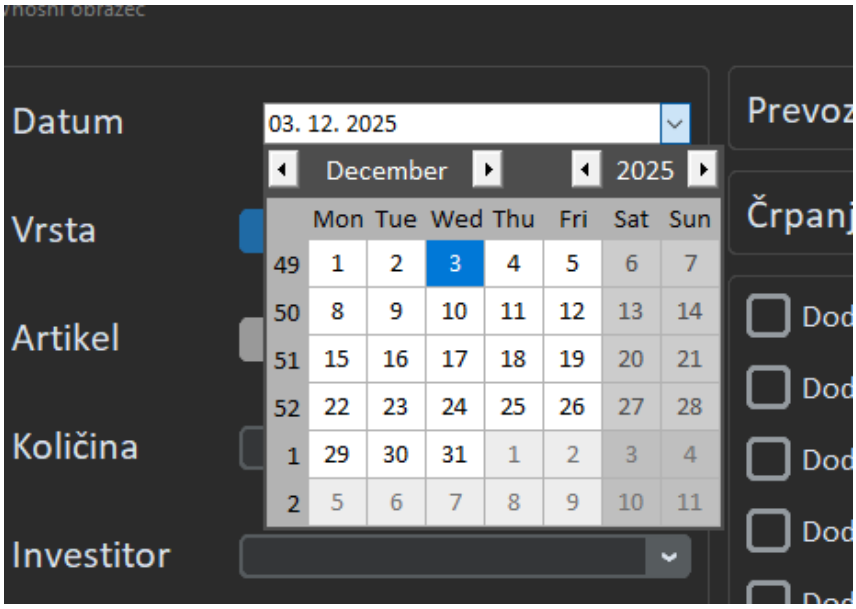
2.4 Primer uporabe s simulacijo poslovnega procesa

Preden lahko začnemo s prvim korakom našega ETL cevovoda, moramo razviti vnosni obrazec. V programu PyCharm sem s pomočjo Pythona in knjižnic Tkinter in Customtkinter razvil vnosni obrazec, prek katerega zaposleni vnašajo naročilo. Vsi podatki so testni in izdelani s pomočjo knjižnice Faker. Poslovni proces poteka tako, da stranka naroči beton, odvoz odpadkov ali peskokop, podjetje pa dodatno ponuja še storitve, kot so prevoz, črpanje in šest dodatkov. Vsi trije so obračunani glede na količino, ki jo stranka naroči.



Slika 14: Primer celotnega vnosnega obrazca

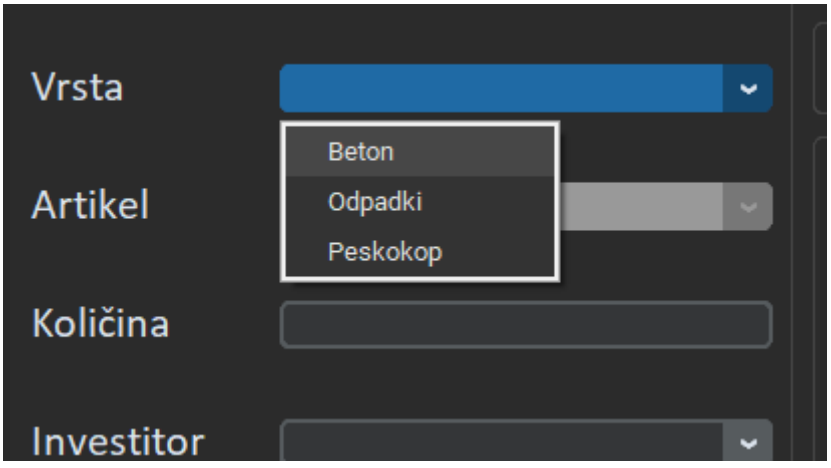
Vir: lastni vir



Slika 15: Izbira datuma s tkcalendar

Vir: lastni vir

Uporaba tkcalendar namesto ročnega vnosa datuma izboljša kakovost podatkov in zmanjšuje možnost napak. Pri ročnem vnosu obstaja tveganje, da uporabnik datum vnese v napačnem formatu (npr. *3.12.25, 03/12/2025, 2025-12-03*), kar kasneje povzroča težave pri filtriranju, sortiranju in obdelavi podatkov. Gradnik "Datepicker" omogoča, da uporabnik datum izbere z enostavnim klikom v koledar, pri čemer je format datuma vedno enoten in sistemsko nadzorovan.

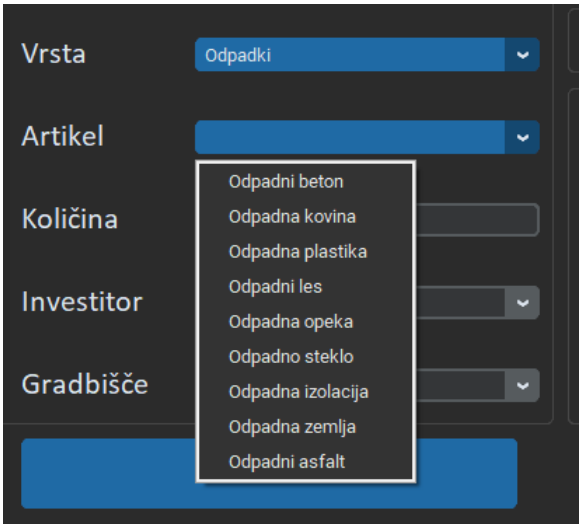


Slika 16: Izbira vrste artikla

Vir: lastni vir

Na vnosnem obrazcu je potrebno najprej izbrati vrsto (npr. Beton, Odpadki, Peskokop). Dokler uporabnik vrste ne izbere, je polje Artikel namerno zaklenjeno. Razlog za to je, da se seznam artiklov dinamično filtrira glede na izbrano vrsto, s čimer se uporabniku prikažejo izključno tisti artikli, ki dejansko spadajo v izbrano kategorijo.

Takšen način vnosa preprečuje tipične napake, ki se pojavljajo pri ročnem sistemu. V Excelu se lahko zgodi, da zaposleni najprej ročno vnesejo vrsto (npr. Peskokop), nato pa zaradi nepreglednosti ali pomote vpišejo artikel, ki ne sodi v to vrsto, kot je npr. artikel "YQ/99/4", ki spada pod beton.



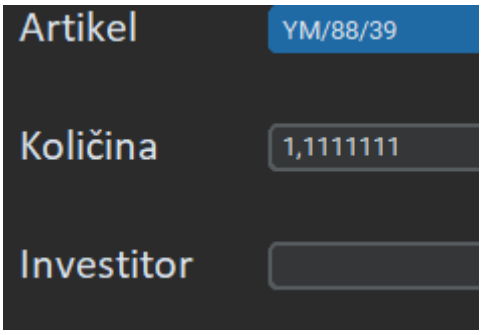
Slika 17: Primer filtriranja artiklov glede na izbrano vrsto

Vir: lastni vir

Pri vnosu podatkov se transformacija ne izvaja le pri izbiri vrste in artikla, ampak tudi pri vnosu količine. Ker zaposleni pogosto uporabljajo različne zapise decimalnih vrednosti (npr. *1,1, 1.1, 1,10*), lahko takšni vnosi v Excelu povzročijo napačne izračune ali napačno interpretacijo podatkov. V vnosnem obrazcu je zato implementirana pomožna funkcija *popravi_zapis_kolicina(widget),* ki ob vsakem vnosu količino avtomatsko:

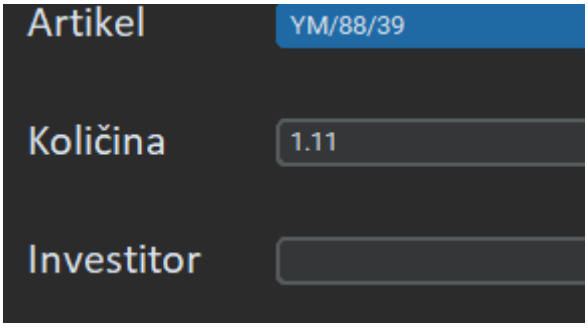
- pretvori v ustrezno številsko vrednost,
- popravi decimalno ločilo,
- oblikuje zapis v standardizirani obliki 0.00,
- zagotovi, da je v podatkovnem cevovodu vedno uporabljena pravilna numerična vrednost.

S tem se popolnoma odpravi ena najpogostejših napak ročnega sistema, ko je zaradi neenotnih zapisov količin prihajalo do napačnih izračunov in dodatnega ročnega popravljanja podatkov.



Slika 18: Neustrezna količina

Vir: lastni vir



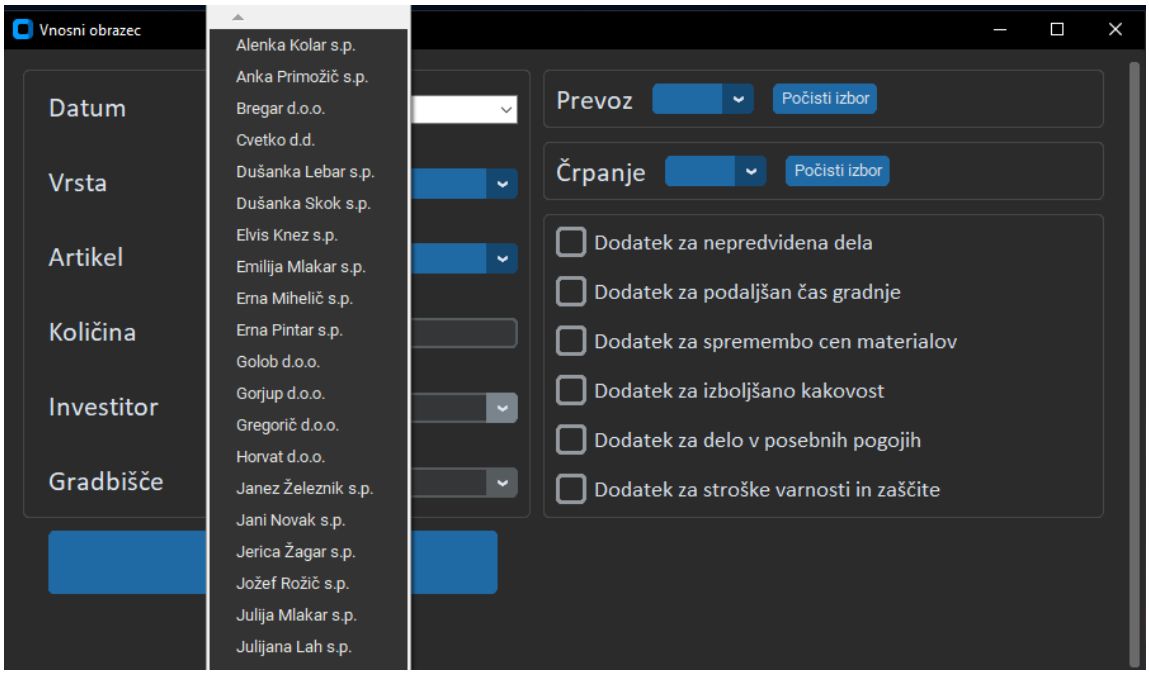
Slika 19: S pomočjo funkcije na standardizirano numerično vrednost popravljena količina

Vir: lastni vir

Pri izbiri investitorja in gradbišča v vnosnem obrazcu uporabnik ničesar ne vpisuje ročno. Program ob zagonu samodejno prebere seznam vseh investitorjev in gradbišč iz Excel tabele, iz katere nato ustvari spustne sezname. Zaposleni tako enostavno izbere ustrezno vrednost s klikom, kar popolnoma izključi možnost ročnega tipkanja in s tem povezanih napak.

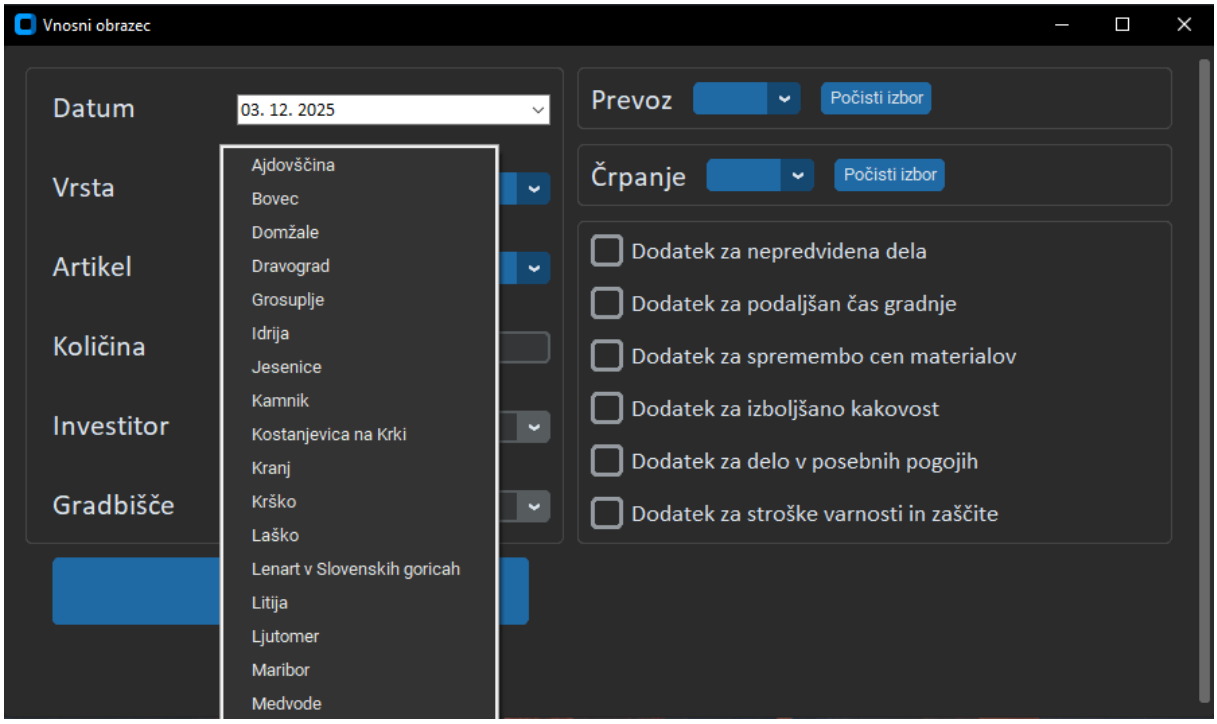
Če želi zaposleni pri vnosu naročila dodati novega investitorja ali novo gradbišče, bi bilo v aplikaciji smiselno razviti ločen vnosni obrazec, namenjen upravljanju teh osnovnih podatkov. Takšen obrazec bi omogočal kontroliran vnos, validacijo in takojšnjo vključitev novih zapisov v sistem. V okviru te diplomske naloge ta funkcionalnost ni bila implementirana, zato se predpostavlja, da so investitorji in gradbišča že vnaprej vneseni v podatkovno strukturo, iz katere aplikacija ob zagonu samodejno prebere in prikaže razpoložljive možnosti v spustnih seznamih.

Takšen pristop bi v celoti odpravil težave, ki so bile prisotne v obstoječem sistemu, v katerem so se zaradi ročnega tipkanja pojavljale različne in nekonsistentne variante istih imen, kot na primer *D.D., D.D, d. o. o., D. o. o.*, zapisi z manjkajočimi pikami ali razlikami v velikosti črk.



Slika 20: Spustni seznam investitorjev

Vir: lastni vir



Slika 21: Spustni seznam gradbišč

Vir: lastni vir

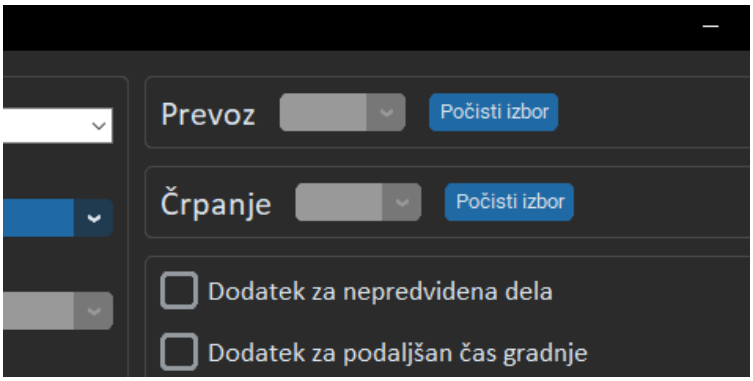
Na desni polovici obrazca sta polja Prevoz in Črpanje privzeto zaklenjena in obarvana sivo. Uporabnik ju lahko vključi šele, ko so izpolnjeni vsi trije osnovni pogoji: izbrani morajo biti vrsta, določen artikel in vnesena količina. Razlog za to je usklajenost s poslovnimi procesi

podjetja, v katerem prevoz ali črpanje nikoli ne predstavljata samostojnih postavk, ampak sta vedno vezana na konkreten artikel in določeno količino materiala. Pri vnosu naročila sem dodatno uvedel tudi logično omejevanje spustnih seznamov za prevoz in črpanje, ki se v praksi obračuna po vneseni količini materiala. Podjetje ima v ceniku obe storitvi ovrednoteni glede na količinski razred ("do 50 m³" in "nad 50 m³"), zato sistem samodejno prilagodi možnosti izbire.

Ko uporabnik vnese količino, program preveri njen obseg:

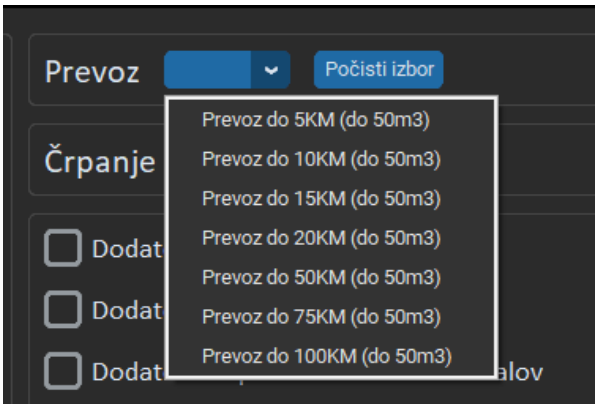
- če je količina manjša od 50 m³, se v spustnem seznamu prikažejo izključno postavke "do 50 m³",
- če količina doseže ali preseže 50 m³, se pokažejo postavke "nad 50 m³".

Na ta način aplikacija s filtriranjem že vnaprej prepreči napačne izbire in zagotovi, da zaposleni vedno izberejo ustrezno možnost glede na dejansko količino materiala. Takšen pristop predstavlja dodatno avtomatsko varovalo, ki zmanjša možnost napak ter hkrati izboljša natančnost in zanesljivost podatkov, vnesenih v sistem.



Slika 22: Zaklenjena gradnika Prevoz in Črpanje

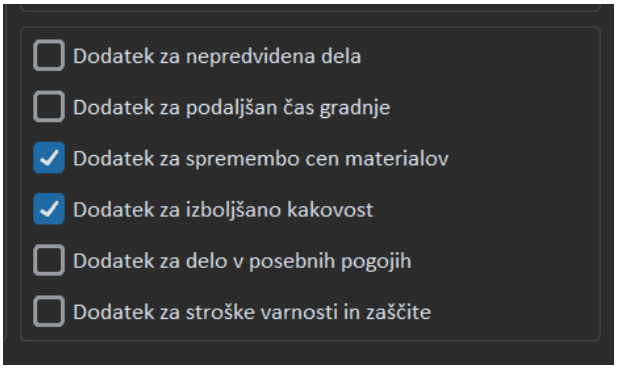
Vir: lastni vir



Slika 23: Primer odklenjenega polja Prevoz

Vir: lastni vir

Polje Dodatki je v vnosnem obrazcu izvedeno v obliki izbirnih polj, kar pomeni, da uporabnik posamezne dodatne storitve preprosto označi s klikom. Na ta način zaposleni ne vpisujejo več dodatnih opisov ali pripisov neposredno k artiklu, kot se je to dogajalo v prejšnjem Excel sistemu (npr. “beton + prevoz 50 km”, “beton črpanje”, “velika črpalka” ipd.).



Slika 24: Izbirna polja za dodatke

Vir: lastni vir

Cena posameznega artikla se v vnosnem obrazcu sedaj samodejno pridobi s cenika, ki ga program ob zagonu prebere in naloži v spomin. Takoj ko uporabnik izbere artikel, sistem ustrezno ceno poveže z izbrano postavko brez potrebe po dodatnem ročnem poseganju. V prejšnjem postopku je moral direktor cene za vsak artikel ročno poiskati v ceniku in jih vnesti v poročilo. Z avtomatskim pridobivanjem cen se celoten proces pospeši, obenem pa se zagotovita popolna natančnost in skladnost s cenikom podjetja.

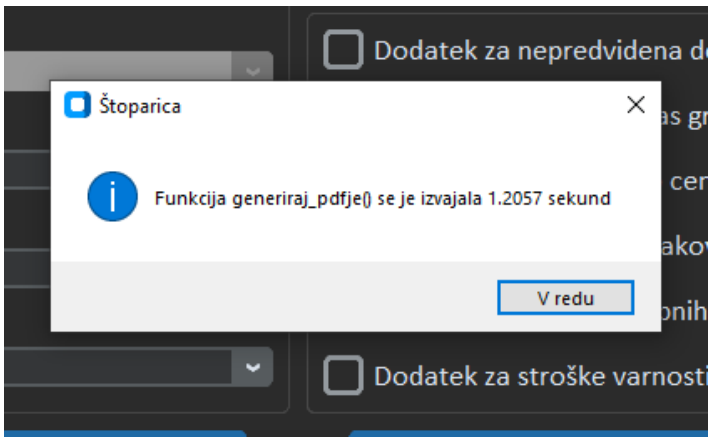
V nadaljevanju naloge bom na podlagi izvedenih meritev, analiz in primerjav začel dokazovati zastavljeno hipotezo, pri čemer bom rezultate ročnega postopka primerjal z avtomatizirano obdelavo podatkov v razvitem podatkovnem cevovodu.

V okviru diplomske naloge je bila logika za generiranje poročil zasnovana tako, da se baza podatkov ob kliku na gumb naloži samodejno. Takšen pristop je primeren za namen raziskave, saj omogoča enotne pogoje testiranja in primerjavo med ročnim in avtomatiziranim postopkom.

V dejanski poslovni aplikaciji bi morala biti logika drugačna. Direktor bi moral imeti možnost samostojno izbrati in naložiti katerokoli bazo podatkov, za katero želi pripraviti poročila. V raziskovalni različici aplikacije ta funkcionalnost namenoma ni implementirana, saj je poudarek izključno na demonstraciji delovanja podatkovnega cevovoda in preverjanju hipoteze, ne pa na končni produkcijski rešitvi.

Za namen analize smo vzpostavili primer testne baze z 10 vrsticami podatkov za istega investitorja, pri čemer je ročni postopek izdelave poročila trajal 11 minut in 12 sekund. Ta rezultat predstavlja izhodiščno vrednost za primerjavo z avtomatizirano obdelavo podatkov.

V nadaljevanju bom časovno učinkovitost avtomatskega postopka izmeril neposredno v programu. Na začetek funkcije za generiranje PDF poročila bom dodal ukaz *start = time.perf_counter()*, na koncu funkcije pa bom dodal ukaz *end = time.perf_counter()*, nato bom rezultat samo še izpisal.



Slika 25: Rezultat štoparice

Vir: lastni vir

GRADBENO PODJETJE

Poročilo za investitorja: Perko d.d.

Gradbišče: TRBOVLJE				
Šifra	Naziv	Količina	Cena	Skupaj
1004	XQ/94/3	74.90 m3	25.00 €	1872.50 €
3001	Mivka	192.00 m3	30.00 €	5760.00 €
4009	Prevoz do 50KM (do 50m3)	22.90 m3	111.00 €	2541.90 €
5003	Srednja črpalka (do 50m3)	11.90 m3	11.00 €	130.90 €
5005	Velika črpalka (do 50m3)	40.10 m3	17.00 €	681.70 €
6004	Dodatek za izboljšano kakovost	11.90 m3	16.00 €	190.40 €
			Skupaj	11177.40 €

Skupaj za plačilo:	11177.40 €

Slika 26: Primer poročila za enega investitorja

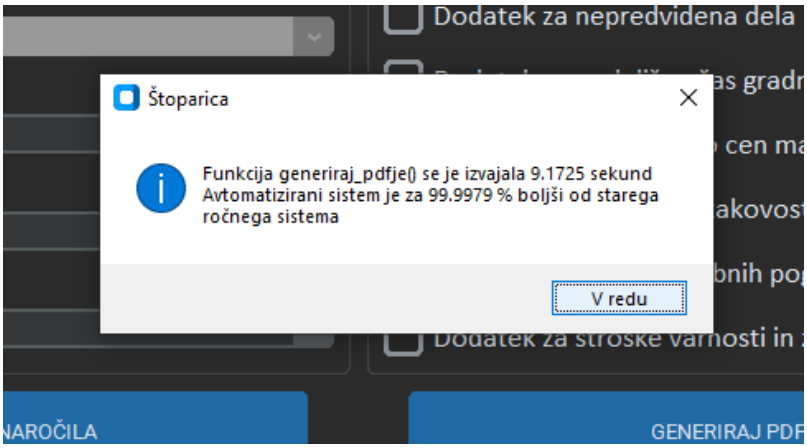
Vir: lastni vir

Rezultat merjenja časa avtomatizirane izdelave poročila je znašal 1,2057 sekunde, kar je veliko hitreje od ročnega postopka, ki je za enake podatke trajal 11 minut in 12 sekund. Obe meritvi sta bili izvedeni na isti testni bazi z desetimi vrsticami za enega investitorja, kar omogoča neposredno primerjavo med obema pristopoma.

Za nadaljnje dokazovanje hipoteze bom izhajal iz predpostavke dela z dejansko realno bazo, ki v povprečju vsebuje približno 800 vrstic na mesec. Za naš primer bomo vzeli točno 800 vrstic na mesec s petdesetimi različnimi investitorji. Direktor za obdelavo takšne količine podatkov in pripravo poročil običajno porabi približno pet dni dela oziroma 432.000 sekund. V avtomatizirani različici bom izmeril, koliko časa potrebuje Python za generiranje poročil za več investitorjev in več gradbišč hkrati, pri čemer bo uporabljena enaka logika kot pri testnem primeru. Na koncu bom na podlagi ročnih in avtomatiziranih časov izračunal, za koliko odstotkov se je poslovni proces izboljšal ter ugotovil, ali velja prva hipoteza.

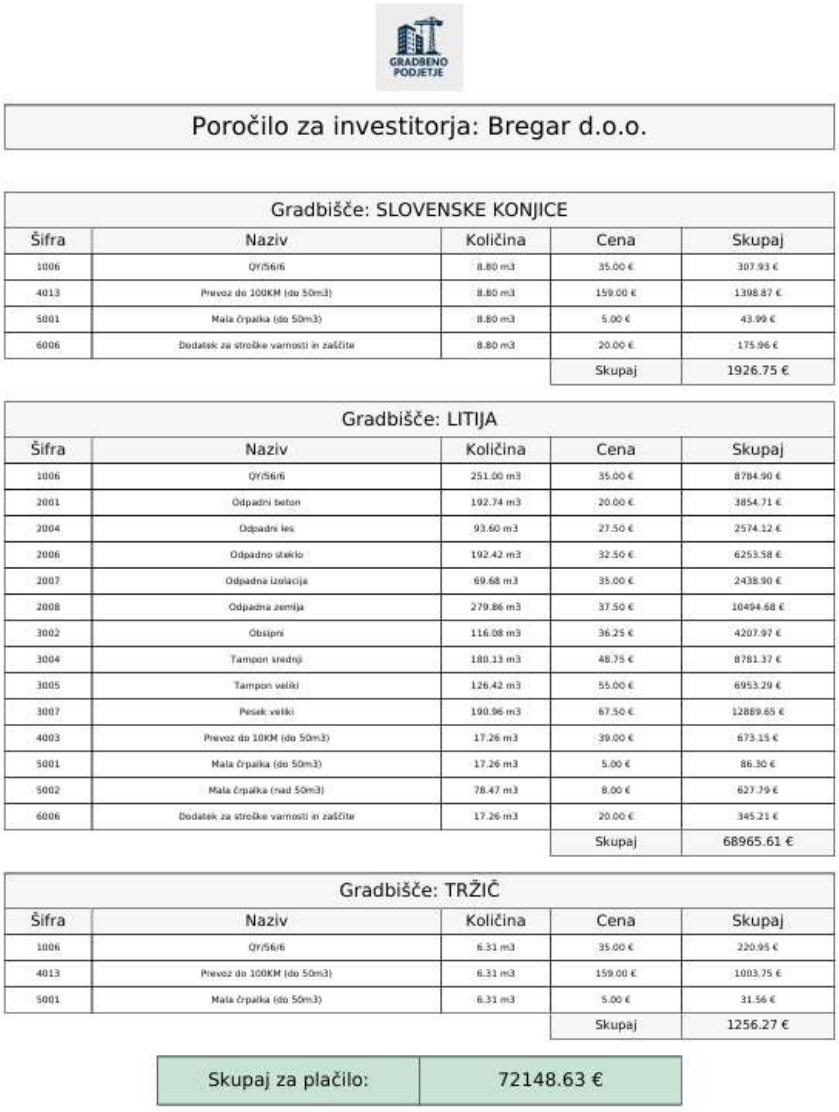
Za izračun sem uporabil formulo:

$$Izboljšanje\ (\%) = \frac{stari\ čas - novi\ čas}{stari\ čas} * 100 \quad (2.1).$$



Slika 27: Štoparica za generiranje poročil

Vir: lastni vir



Slika 28: Primer poročila pri avtomatizirani generaciji

Vir: lastni vir

Na podlagi izvedenih meritev lahko zanesljivo potrdim hipotezo H1, ki navaja, da vpeljava podatkovnega cevovoda skrajša čas izdelave poročila za vsaj 90 % v primerjavi z ročnim postopkom v Microsoft Excelu.

Avtomatizirani sistem je generiral 50 poročil v 9,1725 sekunde, medtem ko direktor za njihovo izdelavo porabi 5 dni. Izračunano izboljšanje znaša približno 99,9979 %, kar več kot presega zahtevani prag iz hipoteze. Rezultati jasno kažejo, da avtomatizacija z uporabo podatkovnega cevovoda ne zmanjšuje le časa, ki je potreben za izdelavo poročil, ampak tudi odpravlja napake, izboljšuje konsistentnost podatkov in v celoti optimizira poslovni proces.

3. PRIMERJAVA OBDELAVE PODATKOV: PANDAS (EXCEL) PROTI POSTGRESQL(SQL)

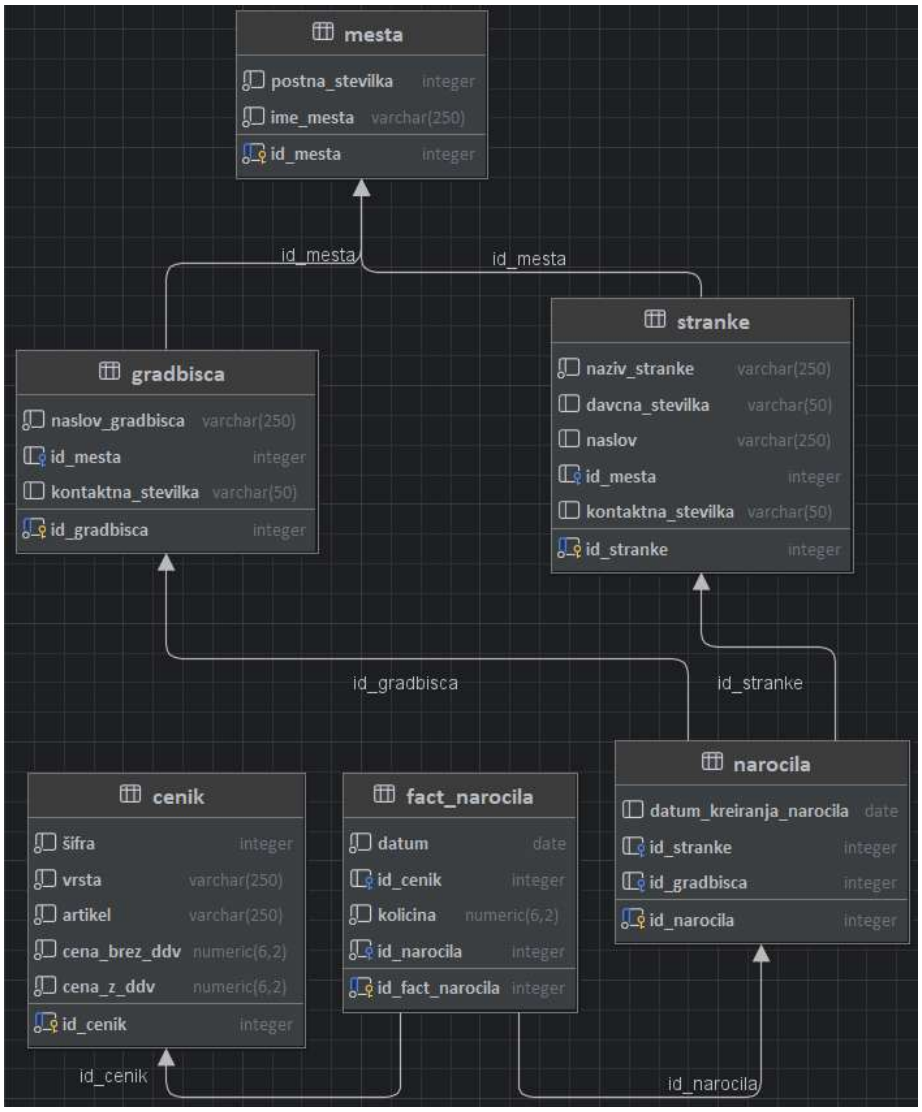
3.1 Hitrost poizvedovanja in obdelave podatkov

V prejšnjem poglavju je bila predstavljena primerjava med ročno izdelavo poročil in njihovim avtomatiziranim generiranjem s pomočjo podatkovnega cevovoda. V nadaljevanju sem to avtomatizirano rešitev razširil še na okolje relacijske baze PostgreSQL, da sem lahko neposredno primerjal delovanje Pandas DataFrame pristopa z delovanjem podatkov, ki so shranjeni v relacijski bazi. Za izvedbo te primerjave je bilo potrebno najprej zasnovati strukturo podatkovne baze, kar sem izvedel v orodju Toad Data Modeler, nato pa model implementiral v okolju DataGrip. Na podlagi tako vzpostavljene baze lahko v tem poglavju sistematično ocenim razlike med obema pristopoma in preverim, kateri izmed njiju je primernejši za obravnavani poslovni primer.

V okviru primerjave med Pandas DataFrame pristopom in relacijsko bazo PostgreSQL je bilo potrebno vzpostaviti tudi enako podatkovno strukturo, kot jo uporablja obstoječi Excel sistem. Za potrebe testiranja sem zato v PostgreSQL zgradil popolnoma enako bazo podatkov, pri čemer sem strukturiral tabele za investitorje, gradbišča, cenik, naročila in posamezne postavke naročil.

Pri tem se je izkazalo, da obdelava podatkov v relacijski bazi zahteva dodatne korake, ki v Excelu niso potrebni. Ker Excel deluje kot "dvodimenzionalna" tabela, lahko uporabnik v eno vrstico vnese tako glavne podatke o naročilu kot vse pripadajoče artikle. V relacijskem modelu to ni mogoče, saj mora biti naročilo najprej zapisano kot samostojen zapis v tabeli *Narocila*, šele nato pa se lahko k temu zapisu dodajajo posamezne postavke v tabelo *Fact_narocila*. To pomeni, da sem moral najprej generirati glave naročil, nato pa na podlagi teh primarnih ključev (id_narocila) ustvariti dodatne vrstice za vsak artikel posebej.

Ta postopek jasno pokaže razliko med Pandas DataFrame pristopom in relacijskim modelom: Pandas omogoča takojšnjo obdelavo podatkov iz ene same tabele, medtem ko PostgreSQL zahteva strogo normalizacijo in ločevanje entitet, kar pomeni implementacijo dodatne logike pri uvozu in obdelavi podatkov. Ravno zato lahko pri manjšemu obsegu podatkov Pandas podatkovni okvir deluje hitreje in bolj fleksibilno, saj ne zahteva vnaprej definiranih relacijskih struktur, ampak podatke obdela neposredno.



Slika 29: Diagram relacij tabel v DataGripu

Vir: lastni vir

Na prikazani shemi je predstavljena podatkovna baza, zasnovana v orodju Toad Data Modeler in implementirana v okolju DataGrip. Model prikazuje relacijsko strukturo, ki združuje vse pomembne entitete poslovnega procesa: stranke (investitorje), gradbišča, mesta, cenik, naročila ter dejanske postavke naročil (*Fact_narocila*). Vsaka tabela je povezana preko primarnih in tujih ključev.

Tabela *Stranke* hrani osnovne podatke o investitorjih, medtem ko tabela *Gradbisca* opisuje lokacije, na katerih se izvajajo dela. Obe tabeli sta povezani s tabelo *Mesta*, kar omogoča centralizirano upravljanje geografskih informacij. Tabela *Cenik* vsebuje artikle in pripadajoče cene, ki se nato preko tujih ključev povezujejo s tabelo *Fact_narocila*, v kateri so zabeležene posamezne postavke naročila (artikel, količina, datum). Tabela *Narocila* deluje kot glava dokumenta in povezuje posamezne postavke z investitorjem in gradbiščem.

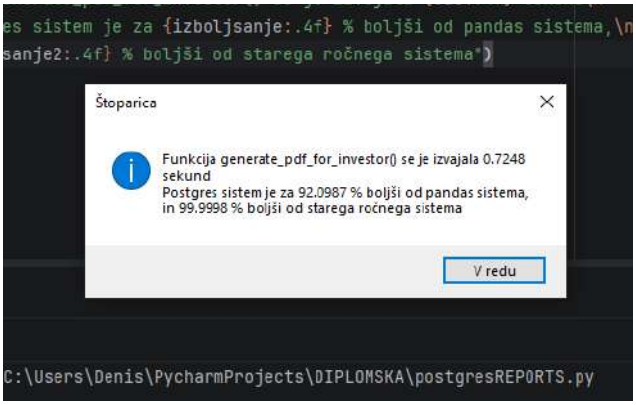
Takšen podatkovni model omogoča strukturirano shranjevanje podatkov in jasno razmejene relacije med entitetami, kar je značilno za relacijske baze podatkov. Model je zasnovan tako, da podpira robustno poizvedovanje, agregiranje in analizo podatkov, hkrati pa omogoča natančno kontrolo nad integriteto in skladnostjo informacij.

Za potrebe primerjave učinkovitosti različnih pristopov generiranja poročil sem pripravil tudi rešitev, ki uporablja Python v kombinaciji s knjižnico *psycopg2* za dostop do podatkov iz PostgreSQL baze in knjižnico *ReportLab* za ustvarjanje PDF poročil. S tem pristopom sem iz podatkovnega modela, ki ga uporabljam tudi v Excel okolju, generiral enaka poročila iz relacijske baze.

Najprej sem se z uporabo knjižnice *psycopg2* povezal na PostgreSQL podatkovno bazo, iz katere sem pridobil seznam unikatnih parov *investitor–gradbišče*. Na podlagi teh parov sem izvedel parametriziran SQL poizvedbeni stavek, ki združuje vse transakcije (postavke naročil) posameznega investitorja in gradbišča ter izračuna skupne vrednosti posameznih artiklov.

Za generiranje PDF dokumentov sem uporabil knjižnico *ReportLab*. V ta namen sem implementiral funkcijo *generate_pdf_for_investor()*, ki oblikuje naslovno vrstico, tabelo postavk naročil ter celotni znesek, ki ga je posamezni investitor ustvaril. Zaradi uporabe slovenskih znakov sem moral dodatno registrirati Unicode pisavo *DejaVu Sans*, saj privzete PDF pisave ne podpirajo šumnikov.

Na koncu sem meril tudi čas izvajanja funkcije in ga primerjal s prejšnjim pristopom, ki je poročila generiral iz Pandas podatkovnih okvirov. Rezultati so pokazali, da je generiranje poročil s PostgreSQL hitrejše predvsem zaradi učinkovitejšega izvajanja agregacij že na ravni podatkovne baze.



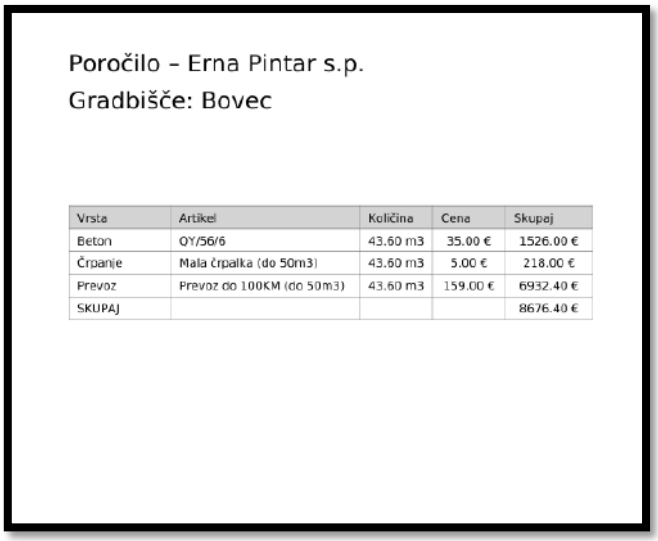
Slika 30: Rezultati štoparice pri PostgreSQL rešitvi

Vir: lastni vir

Na sliki je prikazan rezultat merjenja časa izvajanja funkcije *generate_pdf_for_investor()*, ki se izvaja nad podatki v PostgreSQL podatkovni bazi. Funkcija se je izvedla v približno 0,72 sekunde, kar jasno kaže, da je obdelava podatkov v relacijski podatkovni bazi hitrejša v primerjavi z uporabo Pandas DataFrama ter ročnega postopka v Excelu. Uspeli smo torej generirati 50 poročil oziroma 50 PDF datotek v času 0,72 sekunde, medtem ko ročni postopek izdelave poročil traja približno 5 delovnih dni.

Na podlagi natančno izvedenih meritev je PostgreSQL:

- 92,10 % hitrejši od obdelave istih podatkov v Pandas DataFramu,
- 99,9998 % hitrejši od starega ročnega postopka v Excelu.


Poročilo – Erna Pintar s.p.
Gradbišče: Bovec

Vrsta	Artikel	Količina	Cena	Skupaj
Beton	QY/56/6	43.60 m3	35.00 €	1526.00 €
Črpanje	Mala črpalka (do 50m3)	43.60 m3	5.00 €	218.00 €
Prevoz	Prevoz do 100KM (do 50m3)	43.60 m3	159.00 €	6932.40 €
SKUPAJ				8676.40 €

Slika 31: Primer enega poročila

Vir: lastni vir

3.2 Uporabnost in fleksibilnost pri manjših projektih

Pandas je pri manjših in srednje velikih projektih zelo fleksibilno orodje za obdelavo podatkov, saj omogoča hitro filtriranje, preoblikovanje, čiščenje in analizo podatkov brez potrebe po strogih, vnaprej definiranih shemah. Delovanje knjižnice Pandas omogoča takojšnje izvajanje ukazov in enostavno preverjanje vmesnih rezultatov, kar je zelo primerno za postopno, iterativno delo. Zaradi tega je Pandas pogosto primeren v okoljih, v katerih se podatki pogosto spreminjajo ali v katerih uporabnik potrebuje prilagodljivost pri preoblikovanju strukture podatkov. V takšnih primerih je Pandas mogoče razumeti kot "food court" okolje, v katerem je mogoče hitro preizkušati različne možnosti, se sproti odločati in takoj videti rezultate.

Relacijske baze, kot je PostgreSQL, bazirajo na strogo določenih shemah in zahtevajo natančno opredeljeno strukturo podatkov že na samem začetku. Tak pristop je idealen za večje, kompleksne sisteme, v katerih je nujno zagotavljati kvaliteto podatkov. Zelo dobro podpirajo kompleksne relacije med tabelami, transakcijsko varnost in velike količine podatkov, vendar so zaradi svojih omejitev manj primerne za hitro eksperimentiranje in stalne spremembe strukture podatkov. V tej metafori SQL deluje kot Michelin restavracija – formalno, strukturirano okolje, v katerem je vse strogo določeno, rezultat pa vrhunsko nadzorovan, a manj prilagodljiv in zahteva več priprave.

Pri obsegu približno 800 vrstic podatkov, kot je obravnavano v tej nalogi, dodatna kompleksnost relacijske baze ne prinaša praktičnih prednosti. Podatki so dovolj majhni, da jih Pandas v celoti obdela, kar omogoča izjemno hitro izvajanje operacij brez povezovanja na strežnik ali izvajanja SQL-poizvedb. Relacijska baza bi sicer omogočala strožje nadzorovane vnose podatkov in dodatne omejitve integritete, vendar enak učinek v tem projektu dosežemo že z validacijskimi mehanizmi v uporabniškem obrazcu. Zaradi tega se Pandas izkaže kot učinkovitejša in uporabniku prijazna izbira za manjše projekte, medtem ko SQL pride do izraza predvsem v večjih in kompleksnejših sistemih.

3.3 Omejitve pri uporabi relacijskih baz v praksi

Relacijske baze podatkov so izjemno močno orodje zlasti pri obdelavi velikih količin podatkov, ko pridejo do izraza njihove prednosti, kot so transakcijska konsistenca, indeksiranje, optimizacija poizvedb in skalabilnost, vendar pa imajo v praksi tudi določene omejitve, ki postanejo opazne pri manjših ali srednje velikih projektih, kakršen je obravnavani primer z mesečno bazo približno 800 vrstic. Pri takšnem majhnem obsegu podatkov stroga relacijska struktura, zahteve po definiranju sheme, vzpostavitev strežnika in upravljanje z uporabniki in dostopi pogosto predstavljajo nepotrebno dodatno kompleksnost. V takšnih primerih se Pandas DataFrame izkaže kot preprostejša in razvojno hitrejša izbira, saj omogoča lokalno obdelavo podatkov brez omrežnega dostopa, brez konfiguracije strežnika in z veliko fleksibilnostjo pri transformaciji podatkov.

Pri majhnih naborih podatkov, kot je 800 vrstic, ni težav s skalabilnostjo, zaklepanjem vrstic ali zahtevnimi paralelnimi operacijami – kar so tipične okoliščine pri relacijskih bazah. Pandas omogoča, da so operacije filtriranja, združevanja in agregiranja izvedene v milisekundah, medtem ko PostgreSQL v takšnem okolju uvede dodatne korake, kot so komunikacija s strežnikom, izvajanje SQL-poizvedb in mrežna latenca. Poleg tega uporaba PostgreSQL

zahteva vzpostavitev varne povezave, upravljanje gesel in zaščito strežnika pred morebitnimi vdori, medtem ko Pandas obdeluje podatke lokalno in predstavlja manjše varnostno tveganje.

Če bi se hipoteza osredotočala na hitrost razvoja ali prilagodljivost analitičnega procesa, bi Pandas v resnici predstavljal boljšo izbiro. Kljub temu pa so rezultati eksperimenta pokazali, da hipoteza H2, da Pandas DataFrame omogoča hitrejšo obdelavo podatkov kot SQL-poizvedbe v PostgreSQL, ne drži, ko govorimo o dejanski hitrosti obdelave podatkov. PostgreSQL namreč večino operacij, kot so agregacije, filtriranje in JOIN-i, izvede neposredno na podatkovnem strežniku, na katerem so podatki optimizirani, indeksirani in shranjeni v strukturah, ki omogočajo izjemno hitro procesiranje. Meritev v okviru naloge je pokazala, da PostgreSQL poizvedbe zaključuje hitreje kot enakovredne operacije Pandas.

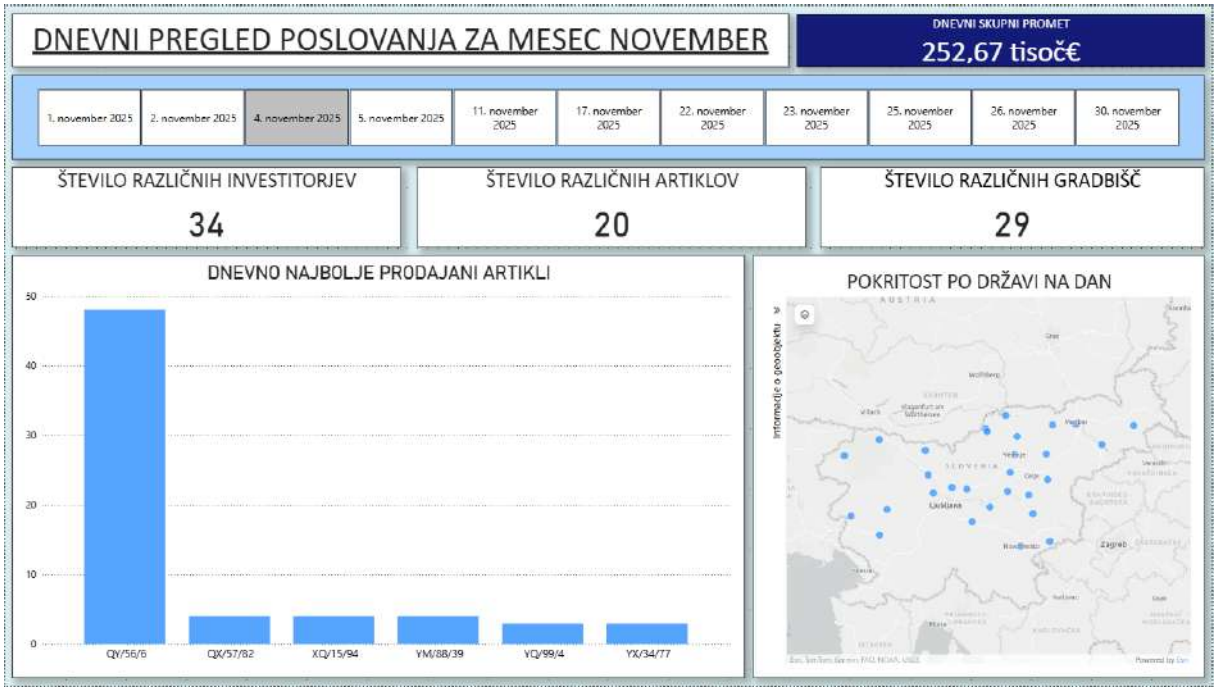
V okoljih, v katerih je podatkovni cevovod že vzpostavljen, so vključeni vnosni obrazci, transakcijski sistem in v katerih se podatki redno posodabljajo ter uporabljajo za nadaljnje poročanje, PostgreSQL prevlada. Pandas mora podatke najprej prenesti iz baze v pomnilnik, kar doda dodatne stroške pomnilnika, medtem ko PostgreSQL izvede celotno logiko neposredno na strežniku, zato se pri realnih obremenitvah in rednih poročilih PostgreSQL izkaže za hitrejšo rešitev.

Ugotovimo lahko torej, da je hipoteza H2 ovržena. Čeprav je Pandas v manjših projektih enostavnejši in razvojno hiter, je v smislu dejanske hitrosti obdelave podatkov PostgreSQL učinkovitejši. Pandas je tako primernejši za prototipiranje in razvoj, PostgreSQL pa za produkcijsko obdelavo in ponovljivo analitiko.

4. PRIMERJAVA ORODIJ ZA VIZUALIZACIJO REZULTATOV (POWER BI PROTI TABLEAU)

4.1 Uporabniška izkušnja in enostavnost uporabe

V okviru praktičnega dela diplomskega dela bom v orodjih Microsoft Power BI in Tableau razvil dve podobni nadzorni plošči, katerih namen je prikaz poslovnih podatkov na pregleden, razumljiv in interaktiven način. Smisel dveh ločenih nadzornih plošč je v ločevanju operativnega in strateškega pogleda na poslovanje, kar omogoča ciljno usmerjeno analizo podatkov ter preprečuje preobremenjenost uporabnika z informacijami.



Slika 32: Nadzorna plošča za dnevni pregled v Power BI

Vir: lastni vir

Nadzorna plošča "Dnevni pregled poslovanja za mesec november" je zasnovana kot operativni pregled, namenjen spremljanju poslovanja za posamezen dan. Njena glavna naloga je omogočiti hiter in pregleden vpogled v dnevno dogajanje ter uporabniku zagotoviti osnovne informacije poslovanja.

V zgornjem delu nadzorne plošče sta jasno izpostavljena naslov, ki opredeljuje časovno obdobje analize, in kazalnik skupnega dnevnega prometa, prikazan v obliki poudarjene kartice (zgoraj desno). Ta kazalnik se dinamično spreminja glede na izbran datum in uporabniku omogoča takojšen vpogled v višino prometa za posamezni dan.

Pod naslovom se nahaja razčlenjevalnik po dnevih, ki je predstavljen v obliki interaktivnih kartic z datumi. Uporabnik lahko z izbiro posameznega dne filtrira vse prikazane vizualizacije na nadzorni plošči, kar omogoča natančno analizo dnevnih podatkov brez potrebe po ročnem spreminjanju filtrov.

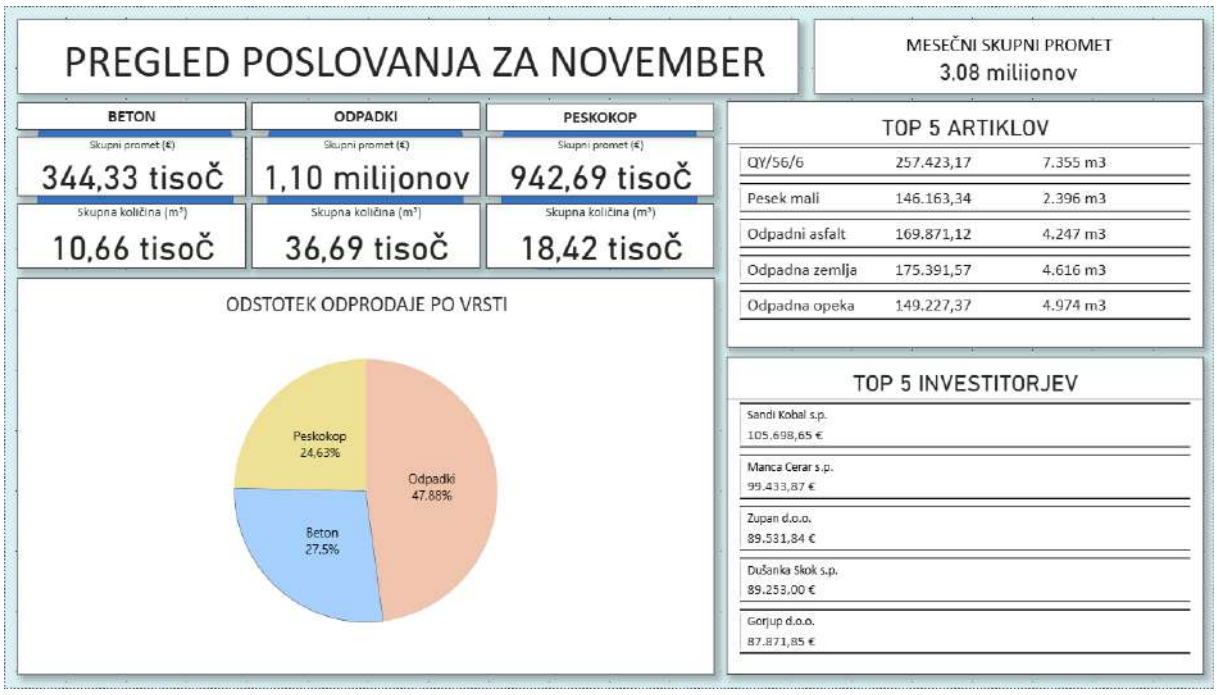
V osrednjem delu so prikazane tri pomembne KPI kartice:

- število različnih investitorjev,
- število različnih artiklov,
- število različnih gradbišč.

Ti kazalniki prikazujejo obseg in raznolikost poslovanja na izbrani dan ter omogočajo hitro oceno, ali je to dan z večjo ali manjšo poslovno aktivnostjo.

Spodnji del nadzorne plošče je namenjen podrobnejši analizi. Na levi strani je prikazan stolpčni graf dnevno najbolje prodajanih artiklov, ki omogoča identifikacijo artiklov z največjo prodajo na izbrani dan.

Na desni strani je postavljena zemljevidna vizualizacija pokritosti po državi, ki prikazuje lokacije gradbišč na izbrani dan. Prostorski prikaz omogoča boljše razumevanje geografskega obsega poslovanja in vizualno poudari razpršenost ali koncentracijo aktivnosti.



Slika 33: Nadzorna plošča za mesečni pregled v Power BI

Vir: lastni vir

Nadzorna plošča "Pregled poslovanja za november" predstavlja strateški oziroma analitični pogled na poslovanje v celotnem mesečnem obdobju. Njena zasnova je usmerjena v agregacijo podatkov in analizo strukture poslovanja ter dopolnjuje dnevni pregled z dolgoročnejšimi vpogledi.

V zgornjem delu nadzorne plošče sta izpostavljena naslov in kartica s skupnim mesečnim prometom, ki povzema celotno vrednost poslovanja v mesecu.

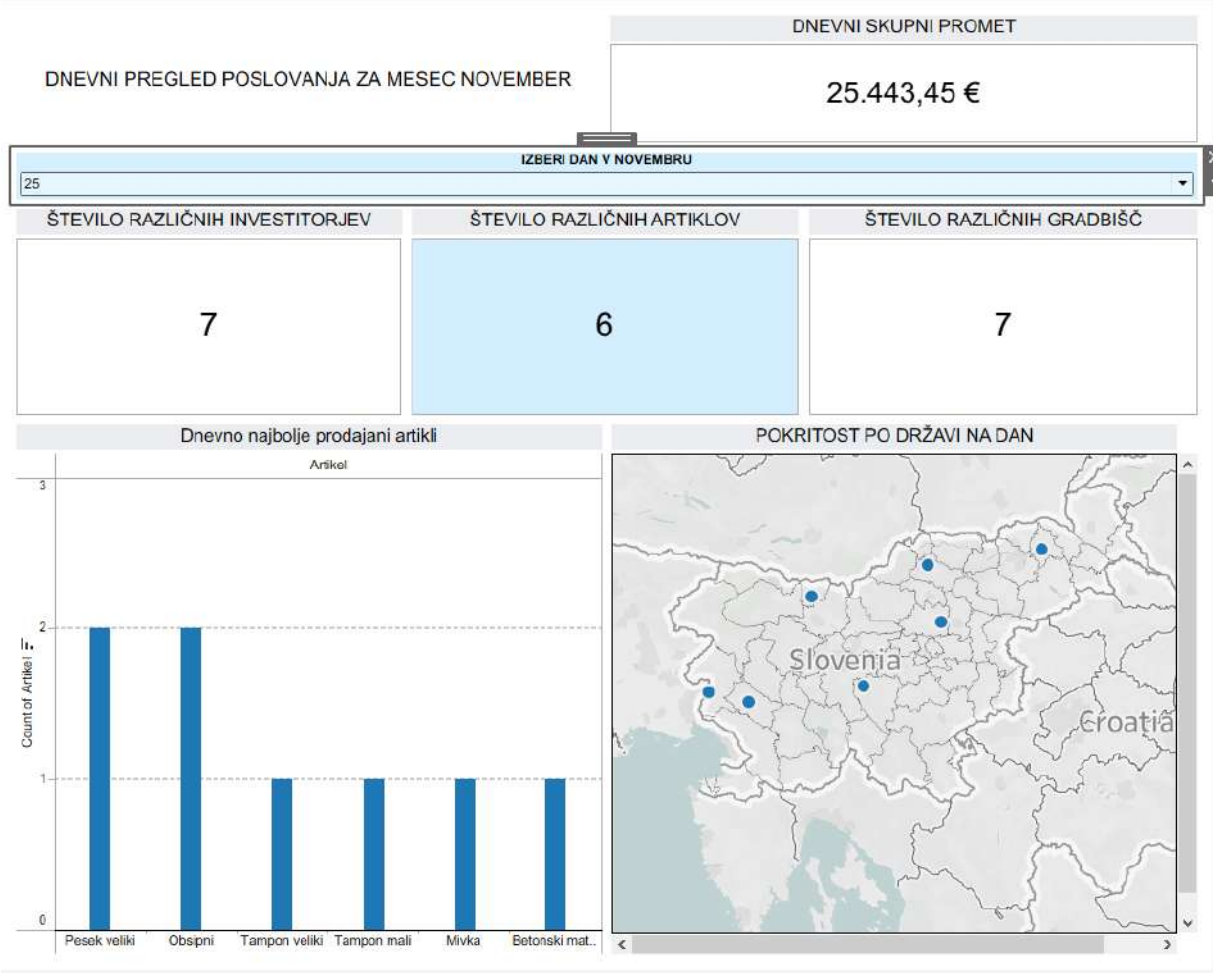
Srednji del plošče je razdeljen po vrstah poslovanja (beton, odpadki, peskokop), za vsako vrsto pa sta na kartici prikazana:

- skupni promet,
- skupna količina v kubičnih metrih (m^3).

V spodnjem levem delu je prikazan tortni diagram odstotka odprodaje po vrsti, ki grafično ponazarja deleže posameznih poslovnih segmentov v skupnem prometu. Ta vizualizacija omogoča razumevanje razmerij med prodanimi artikli, prikazanimi po vrsti v mesecu.

Desni del nadzorne plošče vsebuje dve vizualizaciji. Prva prikazuje top 5 artiklov, poleg imen artiklov pa sta prikazana tudi promet in prodana količina, kar omogoča identifikacijo najpomembnejših produktov v mesecu. Druga tabela prikazuje top 5 investitorjev, ki so razvrščeni glede na ustvarjeni promet, kar daje vpogled v najpomembnejše poslovne partnerje.

Mesečni pregled poslovanja je zasnovan za podporo analitičnemu razmišljanju, primerjavam in dolgoročnejšemu načrtovanju. V nasprotju z dnevnim pregledom ne poudarja časovne dinamike, pač pa strukturo in koncentracijo poslovnih rezultatov.



Slika 34: Nadzorna plošča za dnevni pregled v programu Tableau

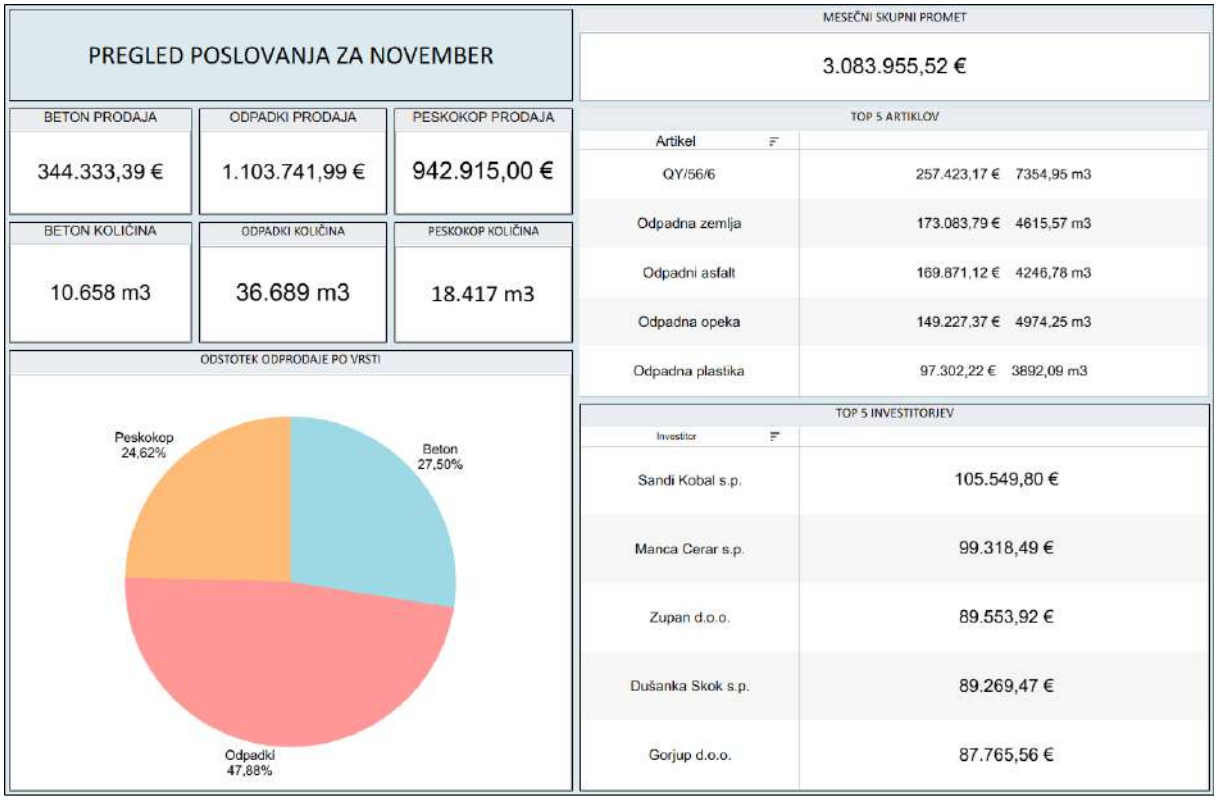
Vir: lastni vir

Nadzorna plošča za dnevni pregled poslovanja v programu Tableau je prav tako zasnovana kot celovit pregled kazalnikov za izbrani dan v mesecu novembru. V zgornjem delu je prikazan skupni dnevni promet, ki uporabniku omogoča hiter vpogled v finančni rezultat izbranega dne. Pod njega je umeščen interaktivni filter v obliki spustnega seznama za izbor dneva, s katerim se vsi prikazani elementi na nadzorni plošči samodejno posodobijo glede na izbrani datum.

Osrednji del nadzorne plošče sestavljajo trije kazalniki: število različnih investitorjev, število različnih artiklov in število različnih gradbišč za mesec. Vsak kazalnik je prikazan ločeno, kar izboljšuje berljivost in omogoča jasno primerjavo med posameznimi metrikami.

V spodnjem delu nadzorne plošče je prikazan stolpčni graf najbolje prodajanih artiklov na izbrani dan. Ta vizualizacija omogoča identifikacijo artiklov z največjim številom prodaj in s tem podpira poslovne odločitve. Desno od grafa je umeščen še zemljevid pokritosti gradbišč po državi, ki prikazuje geografsko razporeditev aktivnosti.

Funkcionalno in vsebinsko je ta nadzorna plošča enakovredna dnevnemu pregledu, ki je bil razvit v orodju Power BI. Kljub temu je bila implementacija v Tableau nekoliko časovno zahtevnejša, saj Tableau uporablja ločeno logiko delovnih listov (worksheets) in nadzornih plošč (dashboards), kar zahteva več vmesnih korakov pri pripravi posameznih vizualizacij. Power BI ima v primerjavi s tem nekoliko preglednejši pristop, saj so vizualizacije in postavitev obravnavane znotraj istega okolja, kar razvijalcu omogoča hitrejši razvoj nadzornih plošč.



Slika 35: Nadzorna plošča za mesečni pregled v programu Tableau

Vir: lastni vir

Nadzorna plošča za mesečni pregled poslovanja je tako kot v Power BI zasnovana kot razširjen analitični pregled poslovnih rezultatov za mesec november. V zgornjem delu je jasno izpostavljen skupni mesečni promet, ki predstavlja osnovni finančni kazalnik uspešnosti poslovanja v obravnavanem obdobju.

V levem osrednjem delu nadzorne plošče so prikazani kazalniki, razdeljeni po vrstah dejavnosti. Prikazani so promet in količine za beton, odpadke in peskokop, kar omogoča neposredno primerjavo tako finančnih kot količinskih vidikov posamezne dejavnosti. Takšna razdelitev omogoča boljše razumevanje prihodkov in obsega prodaje, saj visoki prihodki niso nujno vedno povezani z največjo količino.

Spodnji levi del nadzorne plošče vsebuje tortni diagram, ki prikazuje odstotek odprodaje po vrstah dejavnosti. Vizualizacija omogoča hiter vpogled v relativni prispevek posamezne vrste k skupnemu mesečnemu prometu in poudari dejavnosti, ki imajo največji vpliv na poslovni rezultat.

Desni del nadzorne plošče je namenjen identifikaciji poslovnih elementov. V zgornjem delu je prikazana tabela petih najbolje prodajanih artiklov v mesecu, skupaj s pripadajočim prometom in količino. Pod njo je prikazana še tabela petih investitorjev z največjim prometom, kar omogoča jasno identifikacijo najpomembnejših poslovnih partnerjev podjetja.

Celotna nadzorna plošča je interaktivna, saj se vsi prikazani elementi samodejno prilagajajo izbranim filtrom, kar omogoča dinamično analizo podatkov brez potrebe po ročnem preračunavanju. Funkcionalno je nadzorna plošča primerljiva z mesečnim pregledom, razvitim v orodju Power BI, vendar je bila implementacija v programu Tableau časovno zahtevnejša zaradi ločene logike delovnih listov in nadzornih plošč.

Za namen ocene uporabniške izkušnje in primerjave orodij Tableau in Microsoft Power BI sta bila izvedena dva strukturirana intervjuja. Prvi intervju je bil opravljen z mano kot razvijalcem nadzornih plošč, ker imam poglobljeno tehnično znanje in neposredne izkušnje z razvojem v obeh orodjih. Drugi intervju je bil opravljen z direktorjem podjetja, ki nastopa v vlogi končnega uporabnika nadzornih plošč.

Direktorju sta bili pred uporabo orodij predani obe nadzorni plošči (dnevni in mesečni pregled poslovanja), razviti v Tableau in Power BI, brez dodatnih navodil ali razlage delovanja. Namen tega pristopa je bil preveriti, kako intuitivno se uporabnik znajde v nadzornih ploščah in kako hitro je sposoben samostojno uporabljati funkcionalnosti, kot so filtri, pregled kazalnikov in interpretacija rezultatov.

Zavedati se je treba, da je vzorec uporabnikov v tej raziskavi majhen, vendar zadosten za potrebe diplomskega dela in za osnovno primerjavo uporabniške izkušnje med obema orodjema. Za bolj poglobljeno in statistično zanesljivo analizo bi bilo smiselno vključiti večje število uporabnikov, pripraviti obsežnejši in standardizirani vprašalnik ter v raziskavo vključiti tudi orodja za analizo uporabniškega vedenja, kot so UI testerji, ki bi omogočili identifikacijo delov nadzorne plošče, pri katerih uporabniki doživljajo največ težav ali frustracij.

Intervju 1: Izkušeni uporabnik (avtor rešitve)

1. **Katera orodja za vizualizacijo ste uporabljali pri pripravi nadzornih plošč in zakaj?** Pri razvoju nadzornih plošč sem uporabljal orodji Tableau in Microsoft Power BI, saj obe veljata za vodilni rešitvi na področju poslovne analitike. Namen uporabe obeh orodij je bil neposredna primerjava uporabniške izkušnje, hitrosti razvoja in fleksibilnosti pri implementaciji nadzornih plošč.
2. **Kako bi primerjali zahtevnost razvoja nadzornih plošč v Tableau in Power BI?** Pri razvoju v programu Tableau sem zaznal večjo kompleksnost, saj je potrebno najprej pripraviti posamezne delovne liste (worksheets), šele nato pa jih združevati v nadzorno ploščo. Power BI omogoča bolj neposreden pristop, pri katerem se vizualizacije oblikujejo neposredno na ravni nadzorne plošče, kar občutno pospeši razvoj in zmanjša število vmesnih korakov.
3. **Katero vizualizacijsko orodje vam je osebno bolj ustrezalo in zakaj?** Osebno se bolj nagibam k uporabi Power BI predvsem zaradi večje enostavnosti uporabe, bolj intuitivnega vmesnika in hitrejšega razvoja nadzornih plošč. Možnost neposrednega dodajanja in prilagajanja vizualizacij brez predhodne priprave delovnih listov mi predstavlja pomembno prednost.
4. **Kako ocenjujete preglednost in upravljanje filtrov v obeh vizualizacijskih orodjih?** Power BI omogoča preglednejši in uporabniku prijazen način upravljanja filtrov, saj so filtri jasno vidni in logično umeščeni. V Tableau je upravljanje filtrov bolj tehnično in zahteva dodatno razumevanje, na primer razliko med globalnimi filtri in filtri, vezanimi na posamezne delovne liste.
5. **Ali bi glede na izkušnje priporočili eno od orodij za uporabo v manjšem podjetju?** Na podlagi praktičnih izkušenj bi za manjše podjetje priporočil Power BI, saj omogoča hitrejši razvoj, lažje vzdrževanje in boljšo prilagoditev končnemu uporabniku, ki pogosto nima poglobljenega tehničnega znanja.

Intervju 2: Direktor podjetja (končni uporabnik)

1. **Katera orodja za vizualizacijo ste imeli priložnost uporabljati in v kakšnem kontekstu?**
   Kot končni uporabnik sem uporabljal nadzorne plošče, razvite v orodjih Tableau in Microsoft Power BI, ki so bile pripravljene za spremljanje dnevnega in mesečnega poslovanja podjetja.
2. **Kako bi ocenili preglednost in razumljivost nadzornih plošč v obeh orodjih?** Nadzorne plošče v Power BI so se mi zdele preglednejše in vizualno privlačnejše. Postavitev elementov je jasnejša, najpomembnejši podatki so bolj izpostavljeni, kar mi omogoča hitrejše razumevanje poslovnega stanja.
3. **Katero vizualizacijsko orodje vam je bilo lažje za uporabo in zakaj?** Lažje sem uporabljal Power BI, saj sem hitreje našel filtre, zlasti filter za izbiro datuma. Interakcija z nadzorno ploščo je bolj intuitivna, brez potrebe po dodatnem razlaganju ali tehničnem znanju.
4. **Kako pomembna se vam zdi enostavnost uporabe nadzornih plošč?** Enostavnost uporabe nadzorne plošče je zame najpomembnejša, saj bi nadzorne plošče uporabljal za hitro sprejemanje odločitev. Orodje mora omogočati takojšen dostop do informacij brez dodatnega učenja ali zapletenih postopkov.
5. **Katero vizualizacijsko orodje bi glede na izkušnje raje uporabljali v prihodnje?** Glede na izkušnje bi se odločil za Power BI, saj so nadzorne plošče vizualno bolj všečne, filtri dostopnejši, uporaba pa enostavnejša. To mi omogoča hitrejše in učinkovitejše spremljanje poslovanja.

4.2 Prilagodljivost in integracija z obstoječimi sistemi

Pri izbiri pravega orodja za poslovno inteligenco (BI) sta Microsoft Power BI in Tableau med najbolj uveljavljenimi platformami na trgu, vendar sta zasnovana z nekoliko različnimi pristopi in ciljnimi uporabniki.

Power BI je posebej zasnovan kot prijazen BI sistem za širšo uporabo, ki omogoča enostavno vizualizacijo podatkov in poročanje uporabnikom z različno stopnjo tehničnega znanja in je dobro integriran z ostalimi Microsoftovimi orodji, kot sta Excel in Azure.

Tableau je usmerjen bolj k strokovnim analitikom in podatkovnim znanstvenikom, ki potrebujejo napredne vizualizacijske funkcije in zmogljivejše možnosti za raziskovanje kompleksnih podatkovnih nizov.

Ena izmed razlik med Power BI in Tableau je tudi v ceni in dostopnosti. Power BI ponuja nižje stroške vstopa in vključuje brezplačno verzijo, kar omogoča manjšim podjetjem in posameznikom enostaven začetek z BI rešitvami. Plačane različice Power BI v obliki Pro in Premium omogočajo naprednejše funkcije avtomatizacije in sodelovanje v timih.

Tableau prav tako nudi več nivojev licenciranja, ki so običajno dražji kot Power BI, vendar ustrezajo različnim vlogam znotraj organizacije – od preprostega gledanja (Viewer) do popolnega ustvarjanja vsebin (Creator). Tako strukturirana cenovna politika omogoča prilagoditev stroškov glede na potrebe uporabnikov, vendar pomeni večje začetne stroške za celotno ekipo sodelavcev.

Power BI ima neprekinjeno integracijo z Microsoftovimi storitvami, kar olajša delo uporabnikom, ki uporabljajo Azure, Office 365 ali SQL Server. Te integracije omogočajo hitro dostopanje do podatkov in njihovo obdelavo v okviru že obstoječih sistemov brez kompleksnih povezav. Tableau ponuja širok nabor podatkovnih povezav, vključno z obsežnimi možnostmi povezovanja na različne podatkovne vire, kar omogoča analizo mnogih podatkovnih nizov, vendar pa je proces integracije pogosto bolj splošen in zahteva več ročnega pregleda ter razumevanja podatkovnih virov, kar je lahko prednost ali slabost glede na tehnološke potrebe organizacije.

Velika prednost Tableau je njegova osredotočenost na napredno vizualizacijo in raziskovanje podatkov. Tableau pogosto velja za orodje, ki omogoča ustvarjanje bolj osebnih in kompleksnih vizualnih prikazov, kar je glavna zahteva za analitike, ki želijo podrobno raziskati podatke in odkriti skrite vzorce. Power BI prav tako omogoča širok nabor vizualizacij in interaktivnih nadzornih plošč, vendar je njegov poudarek nekoliko bolj usmerjen k hitri implementaciji in operativnim poročilom, pri katerih sta pomembni učinkovitost in hitra interpretacija poslovnih kazalnikov.

Power BI je pogosto opažen kot lažje dostopen in prijaznejši do uporabnikov, ki se šele spoznavajo s poslovno inteligenco ali prihajajo iz okolij, kot je Excel. To pomeni, da se lahko uporabniki hitreje vključijo v delo z orodjem.

Tableau ima nekoliko strmejšo krivuljo učenja, še posebej za uporabnike, ki niso vajeni kompleksnih vizualnih analiz, kljub temu pa ponuja tudi zmogljive funkcije, ki nagradijo

natančno raziskovanje podatkov in omogočajo napredne analize, kar je prednost za izkušenejše analitike.

Če strnemo primerjavo iz vira (Johnivan, 2025):

- Power BI je cenovno dostopnejši in integriran z Microsoft ekosistemom, primeren za poslovne uporabnike in generalno rabo ter ima intuitiven vmesnik;
- Tableau je močan pri naprednih vizualizacijah in analitiki ter se bolje obnese pri kompleksnih podatkih, vendar s pridihom višjih stroškov in nekoliko strmejše krivulje učenja.

4.3 Končna ocena za uporabo v manjšem podjetju

Na podlagi praktične implementacije nadzornih plošč v orodjih Tableau in Microsoft Power BI ter izvedene ocene uporabniške izkušnje je mogoče podati končno oceno primernosti obeh orodij za uporabo v manjšem podjetju. Analiza je pokazala, da sta bili v obeh orodjih razviti funkcionalno enakovredni nadzorni plošči, ki omogočata dnevni in mesečni pregled poslovanja ter zagotavljata vse pomembne informacije, ki jih dano podjetje potrebuje za pregled.

Kljub primerljivosti pa so se v praksi pokazale pomembne razlike v enostavnosti uporabe in hitrosti razvoja. Power BI omogoča neposrednejši pristop k oblikovanju vizualizacij, saj se večina elementov razvija neposredno na ravni nadzorne plošče, brez potrebe po predhodnem ustvarjanju in upravljanju posameznih delovnih listov. Takšen pristop je posebej primeren za manjša podjetja, v katerih je čas razvoja omejen, vzdrževanje rešitev pa pogosto poteka brez namenskega BI strokovnjaka.

Ugotovitve intervjujev potrjujejo prednosti Power BI z vidika uporabniške izkušnje. Izkušeni uporabnik, ki je sodeloval pri razvoju nadzornih plošč, je izpostavil večjo preglednost okolja Power BI in hitrejši razvojni proces v primerjavi s Tableau. Končni uporabnik se je v Power BI nadzornih ploščah hitreje samostojno znašel, zlasti pri uporabi filtrov za izbor datuma, in je nadzorne plošče ocenil kot vizualno preglednejše in estetsko bolj dovršene.

Čeprav Tableau ponuja zelo zmogljive analitične funkcionalnosti in natančen nadzor nad podatkovno logiko, njegova ločena struktura delovnih listov in nadzornih plošč zahteva več tehničnega razumevanja in več vmesnih korakov, kar lahko predstavlja oviro za manj izkušene uporabnike. V okviru manjšega podjetja, v katerem je poudarek na enostavnosti, hitrosti in intuitivni uporabi, se je Power BI izkazal kot primernejša izbira.

Na podlagi navedenih ugotovitev in izvedene analize lahko potrdimo hipotezo H3: "Microsoft Power BI je pri pregledu rezultatov poslovanja za končnega uporabnika primernejše orodje kot Tableau."

5. VARNOST PRI UPORABI PODATKOVNIH CEVOVODOV

5.1 Pregled SQL injection ranljivosti

SQL (Structured Query Language) je standardni jezik za delo z relacijskimi podatkovnimi bazami. Uporablja se za shranjevanje, pridobivanje, posodabljanje in brisanje podatkov ter za upravljanje strukture podatkov v bazi. Z uporabo SQL lahko uporabnik natančno določi, katere podatke želi iz baze pridobiti, kako jih filtrirati, združevati ali analizirati. Je glavni vmesnik med aplikacijami in podatkovnimi bazami ter omogoča učinkovito in strukturirano delo z velikimi količinami podatkov.

SQL injection ranljivost se je pojavila že konec devetdesetih let kmalu potem, ko so se spletne aplikacije začele množično povezovati z relacijskimi podatkovnimi bazami. Prvi dokumentirani primeri segajo v leto 1998, ko so raziskovalci opozorili, da je mogoče z nepreverjenimi uporabniškimi vnosi neposredno vplivati na SQL poizvedbe v ozadju aplikacije. V zgodnjih fazah razvoja spleta, varnost ni bila v ospredju, zato so razvijalci SQL stavke pogosto sestavljali dinamično brez ustreznega filtriranja vhodnih podatkov, kar je omogočilo enostavno zlorabo.

Skozi leta se je SQL injection razvijal in prilagajal novim tehnologijam, vendar je osnovni princip ostal enak – zlonamerni SQL ukazi se vnesejo v aplikacijo in se izvršijo nad podatkovno bazo. Kljub razvoju varnostnih mehanizmov, kot so pripravljeni stavki, ORM orodja in varnostni pregledi, SQL injection ostaja ena najpogostejših in najnevarnejših ranljivosti spletnih aplikacij. Vir poudarja, da je razlog za to predvsem človeški faktor – napačna implementacija, pomanjkljivo razumevanje varnostnih praks in vzdrževanje stare kode. Posledice uspešnega napada so lahko zelo resne – od kraje občutljivih podatkov do popolnega uničenja podatkovne baze in sistema.

SQL kot jezik omogoča izvajanje operacij nad podatki, kot so pridobivanje, vstavljanje, spreminjanje in brisanje podatkov. Med osnovne SQL ukaze sodijo SELECT za pridobivanje podatkov, INSERT za vstavljanje zapisov, CREATE za ustvarjanje tabel ali podatkovnih baz, DROP za brisanje celotnih tabel ali baz, DELETE za brisanje posameznih zapisov in ALTER za spreminjanje strukture podatkovnih objektov. Med navedenimi ukazi ima pri SQL injection napadih najpomembnejšo vlogo ukaz SELECT, saj je najpogosteje uporabljen v spletnih aplikacijah in predstavlja glavno vstopno točko za večino napadov. Večina SQL injection napadov pa izkorišča prav SELECT poizvedbe, pri katerih napadalec v obstoječo poizvedbo vstavi dodatno SQL logiko in s tem pridobi nepooblaščen dostop do podatkov. Relacijski

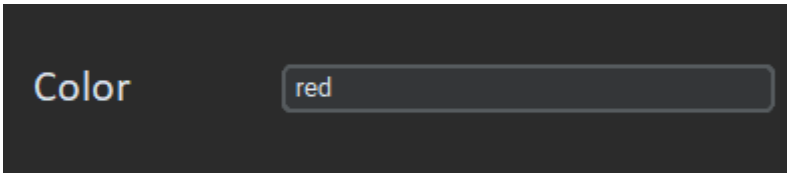
podatkovni model, ki omogoča povezovanje več tabel prek skupnih atributov, sicer povečuje zmogljivost in fleksibilnost obdelave podatkov, hkrati pa ob neustrezni zaščiti povečuje tudi obseg možnih zlorab. Poleg tega se posamezni SQL sistemi med seboj razlikujejo v podrobnostih implementacije, kot sta sintaksa in struktura, pri čemer je MySQL kot odprtokodna in široko uporabljena rešitev pogosto obravnavan kot referenčna implementacija SQL jezika.

Preden podrobneje pregledamo posamezne SQL ukaze in njihovo uporabo v aplikacijah, je treba najprej razumeti, zakaj sploh pride do možnosti vstavljanja poljubnih SQL ukazov. Eden najpomembnejših razlogov se skriva v neustrezno obravnavanem uporabniškem vnosu, pri katerem aplikacija ne izvaja nobenih preverjanj ali omejitev. Podobno kot pri drugih jezikih, ki jih interpretira računalnik, ima tudi SQL natančno določene posebne znake, ki imajo v poizvedbah specifičen pomen.

Med najpogosteje zlorabljenimi znaki sta enojni (') in dvojni (") narekovaj, saj se uporabljata za omejevanje besedilnih vrednosti v SQL poizvedbah. Poleg tega ima pomembno vlogo tudi podpičje (;), ki omogoča ločevanje več SQL ukazov znotraj ene poizvedbe. Posebej problematični so tudi znaki za komentiranje, kot je zaporedje --, saj omogočajo, da se preostanek poizvedbe ignorira, kar napadalcu omogoča obvoz sintaktičnih omejitev.

5.1.1 SQL injection na ravni vnosnega obrazca

Za ponazoritev lahko vzamemo preprost primer aplikacije, ki omogoča iskanje objektov glede na barvo, pri kateri uporabnik v vnosni obrazec vnese besedilo **red** (rdeča barva).



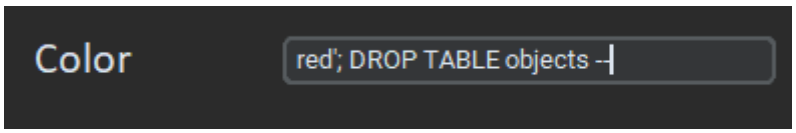
Slika 36: Primer uporabniškega vnosa

Vir: lastni vir

Pri nepravilni uporabi bi aplikacija sestavila SQL poizvedbo, pri čemer se vneseno besedilo v tem primeru direktno prenese na konec sestavljenega SQL stavka:

```
SELECT color, shape FROM objects WHERE color = 'red'.
```

Če uporabniški vnos ni ustrezno preverjen, lahko napadalec namesto pričakovane vrednosti **red** vnese zlonamerno vsebino, na primer:



Slika 37: Primer SQL injection za izbris baze

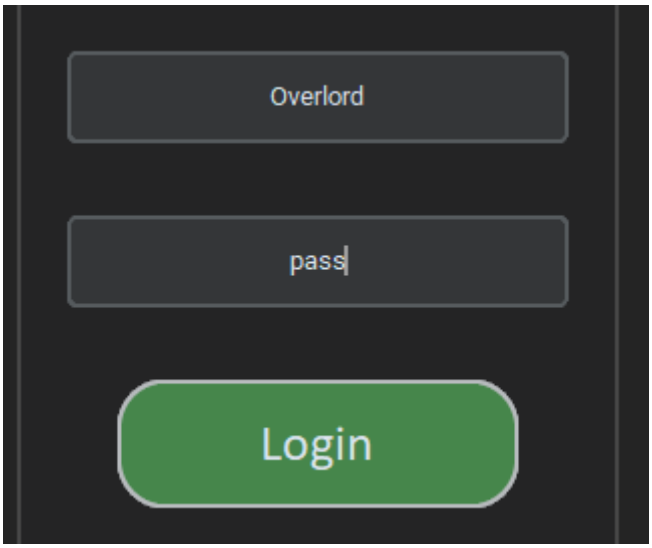
Vir: lastni vir

V tem primeru se poizvedba preoblikuje v zaporedje dveh SQL ukazov, pri čemer prvi del še vedno izpolnjuje pričakovano sintakso, drugi pa povzroči izbris celotne tabele. Z uporabo komentarja se prepreči napaka zaradi odvečnega narekovaja, ki bi ga aplikacija sicer samodejno dodala. Primer sestavljenega nevarnega SQL stavka:

> ***SELECT*** *color, shape* ***FROM*** *objects* ***WHERE*** *color = 'red';* ***DROP TABLE*** *objects --'.*

5.1.2 Obhod avtentikacije in eskalacija privilegijev (E. Caselli, 2020)

Tukaj je še en primer prijavnega obrazca, pri katerem aplikacija preverja, ali se kombinacija uporabniškega imena in gesla ujema z zapisom v podatkovni bazi.



Slika 38: Primer prijave

Vir: lastni vir

Nepravilno sestavljena SQL poizvedba v takšnem primeru izgleda kot:

```
SELECT * FROM users WHERE username = 'Overlord' AND password = 'pass'.
```

Ta SQL stavek izvede poizvedbo nad tabelo *Users*, pri kateri sistem poišče vse zapise uporabnikov, pri katerih se uporabniško ime ujema z vrednostjo **overlord**, njegovo geslo pa je enako vrednosti **pass**. Nevarnost pri direktnem sestavljanju SQL poizvedb se pojavi takrat, ko napadalcu omogočimo, da v polje *uporabnik* vnese nekaj takšnega:



Slika 39: Primer napadalčevega vnosa

Vir: lastni vir

Sestavi se takšen stavek:

```
SELECT * FROM users WHERE username ='' OR '1'='1' -- ' AND password = ''.
```

V tem primeru je napadalec z zlonamernim vnosom spremenil logiko SQL poizvedbe tako, da preverjanje gesla sploh ni bilo več izvedeno. Z uporabo vedno resničnega pogoja ('1'='1') in operatorja OR je bil celoten pogoj WHERE izpolnjen ne glede na dejansko uporabniško ime ali

geslo. Dodatek komentarja z znaki "--" je povzročil, da je bil preostali del poizvedbe ignoriran. Posledično je podatkovna baza vrnila prvi zapis v tabeli uporabnikov, aplikacija pa je uporabnika obravnavala kot uspešno prijavljenega, čeprav ta sploh ni poznal uporabniškega imena in gesla.

5.1.3 Blind SQL injection (E. Caselli, 2020)

V večini realnih aplikacij uporabnik nima neposrednega vpogleda v rezultate SQL poizvedb, saj se te izvajajo v ozadju sistema. V takšnih primerih govorimo o slepem SQL injection napadu (blind SQL injection), pri katerem napadalec ne vidi izpisa podatkov, ampak lahko sklepa o obnašanju podatkovne baze na podlagi odziva aplikacije. Pri blind SQL injection napadih napadalec s pomočjo logičnih pogojev preverja, ali se vneseni SQL izrazi izvajajo. Najpogostejša oblika je boolean-based blind SQL injection, pri kateri se v poizvedbo vstavijo pogoji, ki so bodisi vedno resnični bodisi vedno neresnični. Če aplikacija na oba primera reagira različno, je to jasen znak, da je ranljiva za SQL injection.

Ko je ranljivost potrjena, lahko napadalec postopno pridobiva informacije iz podatkovne baze, na primer z uporabo funkcij za obdelavo nizov, kot je SUBSTRING(). Na ta način je mogoče znak za znakom rekonstruirati podatke, čeprav neposreden izpis ni mogoč. Zaradi počasnosti takšnega postopka se napadi običajno avtomatizirajo.

Druga oblika je time-based blind SQL injection, pri kateri napadalec v poizvedbo vstavi ukaze, ki povzročijo časovni zamik pri izvajanju. Če aplikacija ob določenem vnosu opazno počasneje odgovori, to pomeni, da se vbrizgani SQL ukaz izvaja. Različni sistemi za upravljanje podatkovnih baz za to uporabljajo različne funkcije, vendar je osnovno načelo enako.

Blind SQL injection je zahtevnejša, a zelo nevarna oblika napada, saj lahko ob dovolj potrpežljivosti in avtomatizacije napadalcu omogoči razkritje strukture in vsebine podatkovne baze, tudi kadar aplikacija ne prikazuje nobenih rezultatov ali napak.

5.1.4 Zbiranje informacij in pridobivanje sheme z uporabo UNION poizvedb (E. Caselli, 2020)

Ena izmed najpreprostejših metod prepoznavanja ciljnega sistema je sprožitev napake v SQL poizvedbi. Z namerno napačnim vnosom lahko slabo zasnovana aplikacija vrne podrobno sporočilo o napaki, iz katerega je pogosto razvidno, kateri sistem za upravljanje podatkovnih baz (DBMS – Data base management system) je v uporabi. Čeprav sodobne aplikacije takšne informacije večinoma skrivajo, so starejše ali slabo zaščitene rešitve še vedno izpostavljene tovrstnemu razkrivanju podatkov.

Ena najmočnejših tehnik SQL injection je uporaba ukaza UNION. Ta omogoča združevanje rezultatov več poizvedb v en sam izpis, kar napadalcu omogoča prikaz dodatnih podatkov poleg tistih, ki jih aplikacija običajno vrača. V osnovi se UNION uporablja za združevanje rezultatov poizvedb z enako strukturo stolpcev, vendar se ta lastnost v napadih pogosto zlorablja.

Poleg združevanja obstoječih podatkov je mogoče v UNION poizvedbah uporabiti tudi poljubne vrednosti. To napadalcem omogoča testiranje števila stolpcev v izvirni poizvedbi in prilagajanje strukture napada, da se izognejo sintaktičnim napakam. Ko je struktura enkrat znana, se lahko v rezultatih prikažejo tudi skriti ali občutljivi podatki.

Zelo pogosta tarča napadov je tudi pridobivanje informacij o različici podatkovne baze. Ukazi, kot je SELECT @@VERSION, omogočajo izpis različice DBMS in v nekaterih primerih tudi informacij o operacijskem sistemu. To je še posebej pomembno pri sistemih, kot je Microsoft SQL Server, pri katerem lahko razkritje različice pomeni tudi razkritje znanih varnostnih ranljivosti. Pri drugih sistemih, kot je MySQL, lahko ta ukaz razkrije okolje, v katerem teče baza, kar napadalcu dodatno pomaga pri izbiri ustreznega napada.

Pomemben del zbiranja informacij predstavlja tudi enumeracija shem, tabel in stolpcev v podatkovni bazi. V sistemih, kot je MySQL, je za ta namen posebej uporabna sistemska podatkovna baza information_schema, ki vsebuje metapodatke o celotni strukturi baze. Z ustrezno oblikovanimi UNION poizvedbami lahko napadalec pridobi seznam vseh shem, tabel in njihovih atributov. To je še posebej nevarno v okoljih, v katerih več aplikacij uporablja isto podatkovno bazo, saj lahko ranljivost ene aplikacije razkrije podatke drugih sistemov.

Opisane tehnike jasno pokažejo, da SQL injection ni omejen samo na preproste napade, pač pa omogoča sistematično raziskovanje in postopno razkrivanje celotne podatkovne strukture, zato predstavlja eno najresnejših varnostnih groženj pri uporabi podatkovnih baz in zahteva uporabo zaščitnih mehanizmov že pri sami zasnovi aplikacije.

5.1.5 Zakasnjeni SQL injection (Follin, 2025)

Zakasnjeni SQL injection ali second-order SQL injection je posebna oblika SQL injection napada, pri kateri se zlonamerni SQL ukaz ne izvede takoj ob vnosu, ampak se najprej shrani v podatkovno bazo kot navaden podatek. Do dejanskega napada pride šele kasneje, ko aplikacija ta shranjeni podatek ponovno uporabi v drugi SQL poizvedbi brez ustrezne zaščite. Zaradi tega se ta vrsta napada pogosto imenuje tudi zakasnjeni SQL injection.

Posebnost napada second-order SQL injection je, da je začetni vnos pogosto videti povsem nenevaren. Aplikacija lahko ob vnosu podatkov deluje pravilno, brez napak ali vidnih anomalij,

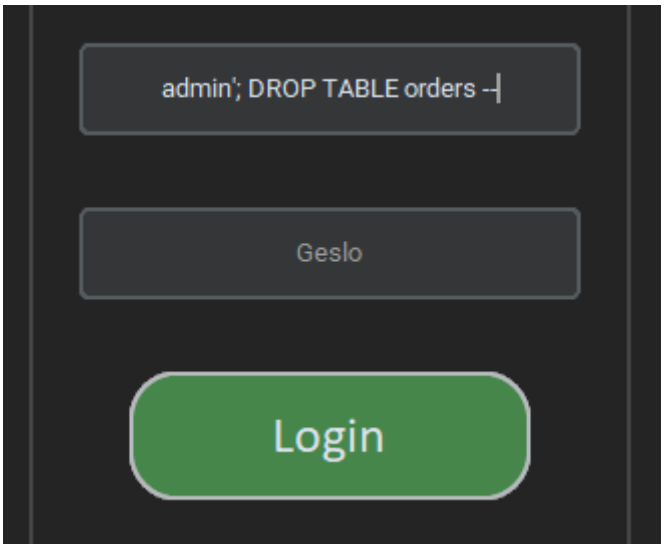
zato ranljivost pogosto ostane neopažena. Napad se sproži šele pri administrativnem pregledu podatkov, generiranju poročil ali izvajanju avtomatiziranih procesov.

Ta vrsta napada je še posebej nevarna v sistemih, v katerih se isti podatki uporabljajo na več mestih, kar je značilno za podatkovne cevovode in poslovne aplikacije. Če aplikacija ob prvem vnosu podatke pravilno shrani, vendar jih kasneje ponovno vstavi v SQL poizvedbo z uporabo dinamičnega sestavljanja nizov, lahko pride do izvajanja zlonamernega SQL ukaza.

Predpostavimo, da aplikacija uporabnikom omogoča urejanje profila in spreminjanje imena. V prvi fazi se izvede SQL stavek, ki uporabnika shrani v podatkovno bazo:

```
INSERT INTO users (username) VALUES ('ime_uporabnika').
```

Napadalec lahko v polje za uporabniško ime vnese naslednjo vrednost:



Slika 40: Primer zakasnjenega SQL injection napada

Vir: lastni vir

Aplikacija tega vnosa ne obravnava kot nevarnega in ga shrani v podatkovno bazo kot navaden niz, saj se ob vnosu ne izvede nobena dodatna SQL logika. V tej fazi do napada še ne pride.

Kasneje administrator sistema odpre stran za upravljanje uporabnikov, na kateri aplikacija dinamično sestavi SQL poizvedbo za prikaz ali obdelavo podatkov, na primer:

***SELECT** * **FROM** users **WHERE** username = 'admin'; **DROP TABLE** orders --'.*

Ker aplikacija v tej fazi ponovno vstavi že shranjeni uporabniški vnos neposredno v SQL poizvedbo, se zlonamerni del ukaza izvrši. Posledično se lahko izvede ukaz za brisanje tabele Orders, kar povzroči izgubo podatkov ali resno motnjo v delovanju sistema.

Pomembno je poudariti, da v tem primeru napadalec ni vnesel SQL injection neposredno v končno poizvedbo, ampak je izkoristil dejstvo, da se shranjeni podatki kasneje ponovno uporabijo. Prav zaradi tega je second-order SQL injection težje zaznati kot klasične oblike napadov.

Ko govorimo o podatkovnih cevovodih, je ta vrsta napada še posebej problematična, saj se podatki pogosto uporabljajo večkrat. Najprej se vnesejo, nato shranijo, kasneje pa obdelujejo, agregirajo ali uporabljajo za poročila. Če v katerikoli od teh faz pride do dinamičnega sestavljanja SQL poizvedb brez ustrezne zaščite, lahko shranjeni zlonamerni podatki sprožijo napad.

5.1.6 Časovnica razvoja SQL injection ranljivosti

SQL injection ranljivost se je pojavila konec devetdesetih let, ko so se spletne aplikacije začele množično povezovati z relacijskimi podatkovnimi bazami. V tem obdobju razvijalci niso posvečali večje pozornosti varnosti, saj je bil poudarek predvsem na funkcionalnosti in hitrosti razvoja aplikacij. Dinamično sestavljanje SQL poizvedb z neposrednim vključevanjem uporabniškega vnosa je bilo zelo razširjeno, kar je ustvarilo idealne pogoje za zlorabe.

1998 – Prvi dokumentirani primeri SQL injection (Invicti, 2013)

Leto 1998 velja za začetek prepoznavanja SQL injection ranljivosti. V tem času so raziskovalci prvič javno opozorili, da je mogoče z manipulacijo uporabniškega vnosa vplivati na SQL poizvedbe v ozadju aplikacij. Napadi so bili sprva preprosti, vendar so jasno pokazali, da lahko nepreverjen vnos vodi do razkritja ali spremembe podatkov v podatkovni bazi.

1999–2002 – Razširitev in prve resne zlorabe (Invicti, 2013)

V naslednjih letih se je SQL injection hitro razširil, saj so spletne aplikacije postajale vse kompleksnejše in vse pogosteje povezane s podatkovnimi bazami. Napadalci so začeli uporabljati SQL injection za pridobivanje uporabniških podatkov, gesel in drugih občutljivih

informacij. V tem obdobju se je ranljivost začela pojavljati tudi v večjih in bolj obiskanih spletnih sistemih.

2003–2005 – SQL injection postane ena glavnih spletnih groženj (Invicti, 2013)

V tem obdobju je SQL injection postal ena najbolj razširjenih in nevarnih spletnih ranljivosti. Napadi so se razširili tudi na večja podjetja in javne ustanove. Razvijalci so začeli prepoznavati problem, vendar so bili obrambni mehanizmi še vedno večinoma omejeni na osnovno filtriranje vhodnih podatkov, ki se je izkazalo za nezanesljivo.

2006–2009 – Organizirani napadi in avtomatizacija (Invicti, 2013)

Napadi SQL injection so v tem obdobju postali bolj organizirani in avtomatizirani. Napadalci so začeli uporabljati specializirana orodja za množično iskanje ranljivih spletnih aplikacij. SQL injection se ni več uporabljal samo za krajo podatkov, ampak tudi za popolno izpostavljenost sistemov, vključno z brisanjem podatkovnih baz in vstavljanjem zlonamerne kode.

2010–2013 – Kljub razvoju zaščit ostaja SQL injection aktualen (Invicti, 2013)

Čeprav so se v tem obdobju pojavili boljši varnostni mehanizmi in smernice za varen razvoj aplikacij, SQL injection ni izginil. Ranljivost se je še naprej pojavljala predvsem v starejših sistemih in aplikacijah, v katerih varnost ni bila pravilno implementirana. SQL injection se je utrdil kot stalnica med najpogostejšimi spletnimi ranljivostmi.

2014–2017 – SQL injection kot trajna grožnja (Invicti, 2013)

V tem obdobju je SQL injection kljub zrelosti spletnih tehnologij ostal ena najpogosteje zaznanih ranljivosti. Napadi so bili uspešni predvsem zaradi nepravilne uporabe obstoječih zaščitnih mehanizmov in vzdrževanja stare kode. Ranljivost se je redno pojavljala tudi na seznamih najnevarnejših spletnih ranljivosti.

2018–2013 (retrospektivni pregled 14 let) (Invicti, 2013)

Vir izpostavlja, da je SQL injection po več kot desetletju še vedno ena najnevarnejših spletnih ranljivosti. Kljub temu da so obrambni mehanizmi dobro znani, se napadi še vedno pojavljajo zaradi človeškega faktorja, napačnih implementacij in kompleksnosti sodobnih aplikacij.

Trenutno stanje SQL injection ranljivosti (Jackson, 2025)

Po poročilu Aikido Security o stanju SQL injection ranljivosti v letu 2024/2025 so SQL injection napadi kljub desetletjem obstoja obrambnih praks še vedno prisotni v velikem deležu

sodobnih digitalnih projektov. Aikido je zbral podatke o najdenih ranljivostih v odprtokodnih in zaprtokodnih projektih ter analiziral njihovo pojavnost.

Statistika iz leta 2024 (Jackson, 2025)

- 6,7 % vseh ranljivosti v odprtokodnih projektih predstavljajo SQL injection ranljivosti.
- 10 % vseh ranljivosti v zaprtokodnih projektih predstavljajo SQL injection ranljivosti.
- Predvidena rast števila SQL injection ranljivosti v odprtokodnih projektih: pričakuje se več kot 2400 primerov SQL injection ranljivosti do konca leta 2024.
- Opazimo 14 % zmanjšanje deleža SQL injection ranljivosti v odprtokodnih projektih v primerjavi z deležem vseh ranljivosti v obdobju od 2023 do 2024.
- Opazimo približno 17% zmanjšanje deleža SQL injection ranljivosti v zaprtokodnih projektih v primerjavi z deležem vseh ranljivosti v obdobju od 2023 do 2024.
- Več kot 20 % zaprtokodnih projektov je ranljivih za SQL injection ob prvem pregledovanju z varnostnimi orodji.
- Povprečno ima ranljivi projekt približno 30 različnih mest v kodi, v kateri se pojavlja SQL injection ranljivost.

SQL injection kljub svoji starosti ostaja ena najresnejših varnostnih groženj, saj problem ne izhaja iz pomanjkanja znanja ali neustreznih obrambnih tehnologij, ampak iz načina razvoja in vzdrževanja sodobnih aplikacij. Osnovni mehanizem napada je že desetletja dobro poznan, vendar se zaradi ponavljajočih se napak pri implementaciji v praksi še vedno pogosto pojavlja.

Sodobne aplikacije so vse kompleksnejše in sestavljene iz več slojev, API-jev, mikroservisov ter zunanjih integracij, kar povečuje število vstopnih točk za uporabniški vnos. Če zaščita pred SQL injection ni implementirana na vseh nivojih sistema, lahko že ena sama nezaščitena poizvedba povzroči resno varnostno ranljivost. Kompleksnost sistemov pogosto otežuje preglednost podatkovnih tokov, kar še dodatno povečuje tveganje za napačno obravnavo vhodnih podatkov.

Pomembno vlogo pri vztrajnosti SQL injection napadov ima tudi človeški faktor. Razvoj programske opreme je pogosto podvržen časovnemu pritisku na delovnem mestu, kar vodi v uporabo bližnjic, ponovno uporabo obstoječe kode in zanemarjanje varnostnih praks. Pogosta napaka je tudi predpostavka, da so podatki, pridobljeni iz internih virov ali drugih delov sistema, že varni in jih ni treba dodatno preverjati.

Kljub temu da sodobna ogrodja in knjižnice ponujajo učinkovite zaščitne mehanizme, kot so parametrizirane poizvedbe in ORM orodja, se te rešitve v praksi pogosto uporabljajo

nepravilno. SQL injection tako ne ostaja aktualen zaradi pomanjkanja tehnologije, pač pa zaradi napačne uporabe obstoječih rešitev. Dodatno tveganje predstavljajo starejši in slabo vzdrževani sistemi, ki so v produkcijskih okoljih še vedno v uporabi. Takšni sistemi pogosto vsebujejo znane ranljivosti, niso redno posodobljeni in niso vključeni v sodobne varnostne procese, kar jih naredi posebej privlačne za napadalce.

5.2 Novosti in pristopi v preprečevanju napadov v zadnjih 5 letih

Najzanesljivejši način zaščite pred SQL injection napadi je ločitev SQL kode od uporabniških podatkov. Do SQL injection ranljivosti namreč pride takrat, ko aplikacija obravnava uporabniški vnos kot del SQL ukaza. Glavni cilj zaščite je zagotoviti, da se uporabniški podatki nikoli ne interpretirajo kot izvršljiva SQL logika.

Najpomembnejši in hkrati najučinkovitejši zaščitni mehanizem so parametrizirane poizvedbe (prepared statements). Pri tem pristopu je struktura SQL poizvedbe vnaprej določena, uporabniški podatki pa se posredujejo ločeno kot parametri. Podatkovna baza tako jasno loči med SQL ukazi in vrednostmi, kar učinkovito prepreči možnost vbrizgavanja zlonamerne SQL kode. Ta pristop velja za osnovni standard varnega razvoja aplikacij, ki uporabljajo podatkovne baze.

Drugi pomembni zaščitni mehanizem je uporaba ORM (Object–Relational Mapping) ogrodij, ki omogočajo delo s podatkovno bazo brez neposrednega pisanja SQL poizvedb. ORM orodja samodejno generirajo varne poizvedbe in pravilno obravnavajo uporabniški vnos. Kljub temu je pomembno poudariti, da ORM ne zagotavlja popolne varnosti, če razvijalec ročno vstavi SQL kodo ali vgrajene mehanizme uporablja nepravilno.

Dodatno zaščito predstavlja validacija uporabniškega vnosa. Ta metoda je prisotna pri preverjanju, ali vnos ustreza pričakovani obliki, tipu in dolžini. Priporočljiva je uporaba t. i. bele liste (whitelist), pri kateri so dovoljene le vnaprej določene vrednosti ali vzorci. Validacija sama po sebi ne zadošča za popolno zaščito.

V primerih, ko parametrizirane poizvedbe niso izvedljive, se lahko uporabi ustrezno ubežno kodiranje (escaping) uporabniškega vnosa. Ta pristop poskrbi, da posebni znaki izgubijo svoj sintaktični pomen v SQL jeziku. Vendar OWASP poudarja, da je ta metoda manj zanesljiva in naj se uporablja le kot zadnja možnost, saj je zelo odvisna od pravilne implementacije in specifičnega podatkovnega sistema.

Pomemben del zaščite je tudi omejevanje pravic uporabniških računov v podatkovni bazi. Aplikacija naj uporablja uporabniški račun z najmanjšim možnim naborom pravic, ki so potrebne za delovanje. Tudi v primeru uspešnega SQL injection napada to zmanjša obseg škode, saj napadalec ne more izvajati nevarnih operacij, kot sta brisanje tabel ali spreminjanje strukture baze.

Dodatno varnost zagotavlja tudi pravilno obravnavanje napak. Aplikacija ne sme razkrivati podrobnih informacij o SQL napakah ali strukturi podatkovne baze, saj so takšni izpisi dragocen vir informacij za napadalce. Namesto tega je priporočljivo uporabljati generična sporočila o napakah in podrobnosti beležiti le v strežniške dnevnike.

Kot dodatna obrambna plast se lahko uporabi tudi Web Application Firewall (WAF). WAF analizira prihajajoči promet do aplikacije in zaznava sumljive vzorce zahtevkov, vključno s tipičnimi SQL injection sintaksami. Takšen pristop omogoča blokiranje zlonamernih zahtev, še preden dosežejo aplikacijsko kodo ali podatkovno bazo. WAF ne nadomešča varnega razvoja aplikacije, vendar predstavlja pomembno dodatno zaščito zlasti pri starejših ali težje vzdrževanih sistemih.

Nenazadnje igra pomembno vlogo pri omejevanju posledic napada tudi šifriranje občutljivih podatkov. Če napadalec kljub zaščitnim mehanizmom uspe pridobiti dostop do podatkov, šifriranje zagotavlja, da so ti podatki brez ustreznih kriptografskih ključev neuporabni.

Po daljšem in sistematičnem pregledu strokovne literature ter sodobnih spletnih virov, vključno z dokumentacijo OWASP, Microsoft, EnterpriseDB, eSecurityPlanet, Acunetix, strokovnimi članki na platformi Medium ter angleško literaturo s področja SQL injection strategij, vidimo, da se pri obravnavi obrambnih mehanizmov proti SQL injection napadom njena vsebina praktično ne razlikuje. Ne glede na izvor vira ali način obravnave vsi poudarjajo enake pristope k zaščiti, kot so uporaba parametriziranih poizvedb, ločevanje SQL kode od uporabniškega vnosa, validacija vhodnih podatkov, omejevanje privilegijev podatkovnih uporabnikov, pravilna obravnava napak ter uporaba dodatnih zaščitnih slojev, kot so ORM ogrodja in Web Application Firewalli.

Pri analizi teh virov ni zaznati pojava novih, vsebinsko drugačnih obrambnih mehanizmov, ki bi predstavljali prelomnico ali nov pristop k preprečevanju SQL injection napadov. Obramba pred tovrstnimi napadi se v zadnjih letih ne razvija v smeri novih konceptov, ampak predvsem v smeri izboljšane implementacije, avtomatizacije in uporabe že dolgo znanih varnostnih praks. Večina sodobnih priporočil stoji na istih načelih, ki so bila uveljavljena že pred več kot

desetletjem, pri čemer se danes predvsem prilagajajo sodobnim razvojnim okoljem, arhitekturam aplikacij in podatkovnim bazam.

Na podlagi tega lahko sklepamo, da se v zadnjih petih letih na področju preprečevanja SQL injection napadov niso pojavili novi pristopi, ampak nadgradnja in ponovna uporaba obstoječih metod v novejših aplikacijah in podatkovnih sistemih. S tem je hipoteza potrjena. Razvoj na tem področju se kaže predvsem v boljši ozaveščenosti, standardizaciji in integraciji varnostnih mehanizmov v razvojne procese, ne pa v uvajanju novih obrambnih sistemov.

5.3 Praktični primer varne implementacije

V nadaljevanju poglavja je predstavljen praktični primer prijavnega obrazca, s katerim pokažemo razliko med nepravilno in pravilno implementacijo zaščite pred SQL injection napadi. Namen primera je prikazati, kako lahko že preprosta prijavna funkcionalnost postane ranljiva, če aplikacija uporabniški vnos vključuje neposredno v SQL poizvedbo, in kako se takšna ranljivost učinkovito odpravi z uporabo ustreznih obrambnih mehanizmov.

Najprej je prikazan primer nepravilne implementacije, pri kateri se uporabniško ime in geslo neposredno vstavljata v SQL stavek, kar omogoča manipulacijo poizvedbe in obhod avtentikacije. Nato sledi primer pravilne implementacije, ki uporablja parametrizirane poizvedbe, ko sta SQL logika in uporabniški vnos strogo ločena. S tem praktičnim primerom je prikazano, da je učinkovita zaščita pred SQL injection napadi dosegljiva že z uporabo osnovnih in uveljavljenih varnostnih praks.

```
username = input("Username: ")
password = input("Password: ")

query = (
    "SELECT * FROM users "
    "WHERE username = '" + username + "' "
    "AND password = '" + password + "'"
)

cursor.execute(query)
result = cursor.fetchone()

if result:
    print("Prijava uspešna")
else:
    print("Napačni podatki")
```

Slika 41: Primer nepravilne implementacije

Vir: lastni vir

Prikazani primer predstavlja nepravilno implementacijo prijavne logike, ki je ranljiva za SQL injection napade. Aplikacija od uporabnika pridobi uporabniško ime in geslo ter nato SQL poizvedbo sestavi z neposrednim združevanjem nizov. Uporabniški vnos se v tem primeru brez kakršnekoli obdelave ali zaščite vstavi neposredno v SQL stavek.

SQL poizvedba se dinamično sestavi tako, da se vrednosti spremenljivk "username" in "password" dodajo v pogoj WHERE. Takšen pristop pomeni, da aplikacija ne loči med SQL logiko in uporabniškimi podatki, ampak obravnava celoten vnos kot del izvršljive SQL kode. Posledično lahko zlonamerni uporabnik v polja za vnos vnese SQL izraz, ki spremeni logiko poizvedbe.

Zaradi neposrednega vključevanja uporabniškega vnosa je omogočen obhod avtentikacije. Če napadalec v polje za uporabniško ime vnese posebej oblikovan niz ' OR '1'='1' --, lahko spremeni pogoj WHERE v vedno resničen izraz, zaradi česar podatkovna baza vrne zapis ne glede na dejansko veljavnost prijavnih podatkov. Aplikacija tak rezultat obravnava kot uspešno prijavo in uporabniku omogoči dostop.

```
username = input("Username: ")
password = input("Password: ")

query = "SELECT * FROM users WHERE username = %s AND password = %s"
cursor.execute(query, (username, password))

result = cursor.fetchone()

if result:
    print("Prijava uspešna")
else:
    print("Napačni podatki")
```

Slika 42: Primer pravilne implementacije

Vir: lastni vir

Prikazani primer predstavlja pravilno implementacijo prijavne logike, ki je napisana tako, da učinkovito preprečuje SQL injection napade. Aplikacija od uporabnika pridobi uporabniško ime in geslo, vendar teh vrednosti ne vključuje neposredno v SQL poizvedbo. Namesto tega uporablja parametrizirano poizvedbo, pri kateri sta SQL koda in uporabniški vnos strogo ločena.

SQL poizvedba je definirana vnaprej, pri čemer so v njej uporabljeni posebni označevalci (%s) za parametre. Uporabniško ime in geslo se podatkovni bazi posredujeta ločeno kot vrednosti parametrov ob klicu metode za izvajanje poizvedbe. Podatkovna baza tako obravnava vnesene podatke izključno kot vrednosti in ne kot del izvršljive SQL logike.

Takšen pristop onemogoča manipulacijo poizvedbe, saj tudi v primeru, ko uporabnik vnese SQL sintakso, ti ne morejo vplivati na strukturo stavka WHERE. Vneseni podatki se ne interpretirajo kot SQL koda, pač pa kot običajni nizi, kar učinkovito preprečuje obhod avtentikacije in druge oblike SQL injection napadov.

Po izvedbi poizvedbe aplikacija preveri, ali je bil vrnjen kakšen zapis. Če rezultat obstaja, se prijava šteje za uspešno, v nasprotnem primeru aplikacija uporabniku vrne splošno sporočilo o napačnih podatkih. Pomembno je poudariti, da aplikacija ne razkriva, ali je bilo napačno uporabniško ime ali geslo, kar dodatno zmanjšuje možnost zlorab.

Razložimo še en primer nepravilne implementacije, pri kateri je na ravni uporabnikovega vnosa že implementiran sistem parametrizirane poizvedbe.

```
company = input("Ime podjetja: ")

query = "INSERT INTO profiles (company) VALUES (%s)"
cursor.execute(query, (company,))
```

Slika 43: Primer parametrizirane poizvedbe

Vir: lastni vir

Če sedaj napadalec vnese niz test'; DROP TABLE orders - - pod uporabniško ime, se zlonamerni niz shrani v podatkovno bazo kot navaden tekst in ne povzroči nobene napake ali takojšnjega napada. To je tipičen primer zakasnjenega SQL injection napada.

Kasneje razvijalec napačno predpostavi, da so podatki iz baze varni, in sestavi poizvedbo za prikaz vseh podatkov iz tabele naročil na naslednji način:

```
company = row["company"]

query = (
    "SELECT * FROM orders "
    "WHERE company = '" + company + "'"
)

cursor.execute(query)
```

Slika 44: Primer nepravilne poizvedbe

Vir: lastni vir

Koda, ki se sestavi, izgleda takole:

SELECT * **FROM** orders

WHERE company = 'test'; **DROP TABLE** orders --'.

V tistem trenutku se zlonamerni SQL ukaz dejansko izvede, kar povzroči brisanje celotne tabele oziroma podatkov v tabeli. Glavna napaka razvijalca je napačna predpostavka, da so podatki, pridobljeni iz podatkovne baze, že varni. Čeprav so bili podatki ob vnosu pravilno obravnavani, jih aplikacija kasneje ponovno vstavi v SQL poizvedbo z uporabo dinamičnega sestavljanja

nizov. To potrjuje pomembno varnostno načelo, da so vsi podatki potencialno nevarni, ne glede na njihov izvor.

```
company = row["company"]

query = "SELECT * FROM orders WHERE company = %s"
cursor.execute(query, (company,))
```

Slika 45: Pravilna implementacija poizvedbe razvijalca

Vir: lastni vir

V tem poglavju je bila obravnavana varnost pri uporabi podatkovnih cevovodov s poudarkom na SQL injection ranljivostih. Predstavljena sta bila razvoj SQL injection napadov in glavne oblike napadov, vključno z napadi na ravni vnosnih obrazcev, obhodom avtentikacije, slepimi SQL injection napadi, napadi z uporabo UNION poizvedb ter zakasnjenimi SQL injection napadi, ki so v podatkovnih cevovodih zaradi večkratne obdelave podatkov še posebej nevarni.

Analiza sodobnih virov je pokazala, da se obrambni mehanizmi proti SQL injection napadom v zadnjih letih niso spremenili. Zaščita še vedno bazira na uveljavljenih pristopih, kot so parametrizirane poizvedbe, validacija uporabniškega vnosa, omejevanje privilegijev in pravilna obravnava napak. Poglavje se zaključi s praktičnim primerom, ki jasno ponazori razliko med nepravilno in pravilno implementacijo dostopa do podatkovne baze ter poudari pomen varnega ravnanja z uporabniškim vnosom.

6. SKLEP

Diplomsko delo je bilo usmerjeno v ugotavljanje smiselnosti in učinkovitosti vpeljave podatkovnega cevovoda v manjše podjetje, ki je do sedaj stalo predvsem na ročni obdelavi podatkov v programu Microsoft Excel. Osrednji cilj naloge je bil preveriti, ali lahko sodobni pristopi podatkovnega inženiringa tudi v manjšem poslovnem okolju prinesejo konkretne izboljšave pri obdelavi podatkov, poročanju, analitiki in varnosti.

Analiza obstoječega stanja v podjetju KG-EKO, d. o. o. je pokazala, da je trenutni način dela izrazito časovno potraten, podvržen človeškim napakam in slabo prilagojen avtomatizirani obdelavi. Neurejeni zapisi, ročni vnosi, neustrezna struktura podatkov in potreba po interpretaciji podatkov s strani direktorja so povzročali, da je bila izdelava poročil dolgotrajna in nepregledna. Takšen način dela sicer deluje pri manjšem obsegu podatkov, vendar hitro postane neučinkovit že pri zmerni rasti vrstic v podatkovni bazi.

V okviru diplomskega dela je bil zato zasnovan in implementiran podatkovni cevovod po vzorcu ETL (Extract, Transform, Load), ki omogoča nadzorovan vnos podatkov, njihovo validacijo in standardizacijo že ob samem izvoru. Razvita rešitev je pokazala, da je mogoče večino napak, ki so bile prisotne v obstoječem sistemu, odpraviti že v fazi vnosa, kar izboljša kakovost podatkov v celotnem nadaljnjem procesu. Avtomatizirano generiranje poročil na podlagi takšnih podatkov je prineslo izjemen časovni prihranek, kar jasno potrjuje doseganje zastavljenih ciljev diplomskega dela.

Rezultati meritev so pokazali, da se je čas izdelave poročil z avtomatiziranim pristopom skrajšal za več kot 99 %, s čimer je bila potrjena hipoteza H1. Ta ugotovitev jasno dokazuje, da je vpeljava podatkovnega cevovoda smiselna tudi v manjšem podjetju, v katerem se pogosto predpostavlja, da takšni sistemi niso potrebni.

V nadaljevanju je bila izvedena primerjava obdelave podatkov s pomočjo Pandas DataFrame in relacijske podatkovne baze PostgreSQL. Čeprav se Pandas izkaže kot zelo fleksibilno in razvojno prijazno orodje za manjše projekte, so meritve pokazale, da je PostgreSQL pri večjem obsegu podatkov hitrejši in učinkovitejši. Na tej podlagi je bila hipoteza H2 ovržena. Kljub temu pa primerjava jasno pokaže, da ima vsako orodje svoje mesto, odvisno od faze projekta in zahtev podjetja.

Pomemben del diplomskega dela je bil namenjen tudi vizualizaciji podatkov. Razviti nadzorni plošči v orodjih Microsoft Power BI in Tableau sta omogočili neposredno primerjavo

uporabniške izkušnje. Na podlagi praktične uporabe in izvedenih intervjujev je bilo ugotovljeno, da je Power BI predvsem zaradi enostavnejše uporabe, hitrejšega razvoja in bolj intuitivne interakcije za končnega uporabnika, za manjše podjetje primernejše orodje. S tem je bila potrjena hipoteza H3.

Zadnji del naloge je obravnaval varnost podatkovnih cevovodov s poudarkom na SQL injection ranljivostih. Analiza zgodovinskega razvoja in sodobne literature je pokazala, da kljub razvoju tehnologij v zadnjih letih ni prišlo do novih pristopov za preprečevanje SQL injection napadov, s čimer sem tudi potrdil hipotezo H4. Obramba še vedno stoji na uporabi preverjenih mehanizmov, kot so parametrizirane poizvedbe, validacija vhodnih podatkov, omejevanje privilegijev in pravilna obravnava napak, kar je tudi potrjevalo četrto hipotezo. Poseben poudarek je bil namenjen zakasnjenim (second order) SQL injection napadom.

Na podlagi vseh ugotovitev lahko zaključimo, da diplomsko delo uspešno povezuje teorijo in prakso ter dokazuje, da je mogoče z relativno preprostimi, a premišljenimi pristopi izboljšati obdelavo podatkov, poročanje in varnost tudi v manjšem podjetju. Vpeljava podatkovnega cevovoda se je izkazala kot učinkovita rešitev, ki ne prinaša samo časovnega prihranka, ampak tudi večjo zanesljivost, preglednost in kakovost poslovnih informacij.

V prihodnje bi bilo smiselno razmisliti o selitvi podatkov iz Excel okolja v relacijsko podatkovno bazo, ki bi bila primarni vir podatkov, kar bi dodatno izboljšalo skalabilnost in varnost sistema. Možna nadgradnja vključuje tudi uvedbo uporabniških vlog, revizijskih sledi in avtomatiziranega nadzora nad spremembami podatkov.

Na področju poročanja bi bilo smiselno razširiti obstoječe nadzorne plošče z dodatnimi kazalniki uspešnosti in omogočiti večjo samostojnost končnega uporabnika pri izbiri filtrov in časovnih obdobij. Z vidika varnosti bi bila priporočljiva uvedba rednih varnostnih pregledov kode in dodatnih zaščitnih slojev, kot sta centralizirano upravljanje dostopov in nadzor dostopa do podatkov.

Takšne nadgradnje bi sistem še dodatno približale produkcijski rabi in dolgoročno povečale njegovo uporabno in poslovno vrednost.

7. VIRI IN LITERATURA

Acunetix. (6. december 2025). *SQL Injection (SQLi).* Pridobljeno 17. december 2025 iz Acunetix: https://www.acunetix.com/websitesecurity/sql-injection/

Bartley, K. (28. maj 2025). *ETL vs ELT: What's The Difference?* Pridobljeno iz Rivery: https://rivery.io/blog/etl-vs-elt/

Duarte, F. (24. april 2025). *Amount of Data Created Daily (2025).* Pridobljeno iz Exploding topics: https://explodingtopics.com/blog/data-generated-per-day

E. Caselli, E. G. (2020). *SQL Injection Strategies: Practical techniques to secure old vulnerabilities against modern attacks.* Birmingham: Packt Publishing.

Follin, L. (16. november 2025). *Second Order SQL Injection (Part 3 of the SQL Series).* Pridobljeno iz PentestPeople: https://www.pentestpeople.com/blog-posts/second-order-sql-injection-part-3-of-the-sql-series

Holden, N. (29. april 2024). *Protecting Against SQL Injection.* Pridobljeno iz EDB Postgres: https://www.enterprisedb.com/blog/protecting-against-sql-injection

Invicti. (21. avgust 2013). *14 years of SQL injection history and still the most dangerous vulnerability.* Pridobljeno iz Invicti: https://www.invicti.com/blog/web-security/sql-injection-vulnerability-history

Jackson, M. (4. december 2025). *The State of SQL Injection.* Pridobljeno iz Aikido: https://www.aikido.dev/blog/the-state-of-sql-injections

Johnivan, J. (19. marec 2025). *Power BI vs Tableau: Data Analytics Duel Continues.* Pridobljeno iz TechnologyAdvice: https://technologyadvice.com/blog/information-technology/power-bi-vs-tableau/

MacDonald, L. (25. november 2024). *9 Essential Data Pipeline Design Patterns You Should Know.* Pridobljeno iz Monte Carlo data: https://www.montecarlodata.com/blog-data-pipeline-design-patterns/

Matillion. (28. julij 2025). *How to Build a Data Pipeline from Scratch in 2025.* Pridobljeno iz Matillion: https://www.matillion.com/learn/blog/how-to-build-a-data-pipeline

Matt Gonzales, K. M. (9. julij 2025). *SQL Injection Prevention: 6 Ways to Protect Your Stack.* Pridobljeno iz eSecurity Planet: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks/

Microsoft. (18. november 2025). *SQL injection.* Pridobljeno iz Learn Microsoft: https://learn.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver17

Monga, A. (10. marec 2024). *Why SQL Injection Remains a Top Vulnerability from two decades? SQLi.* Pridobljeno iz Medium: https://medium.com/@ajay.monga73/decades-later-why-sql-injection-remains-a-top-security-concern-9f2092f62844

OWASP. (27. december 2025). *SQL Injection Prevention Cheat Sheet.* Pridobljeno 17. december 2025 iz OWASP Cheat sheet series: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Parameswaran, A. (28. junij 2022). *Pandas vs. SQL — Part 1: The Food Court and the Michelin-Style Restaurant.* Pridobljeno iz Medium: https://medium.com/data-science/pandas-vs-sql-part-1-the-food-court-and-the-michelin-style-restaurant-6921a0786cab

Smith, P. (3. junij 2020). *Data Entry: What Data Entry Error Rate is Acceptable?* Pridobljeno iz Conexiom: https://conexiom.com/blog/understanding-impact-100-percent-order-accuracy/