

UNIVERZA V LJUBLJANI  
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dominik Uršič

**Hibridni pristop k avtomatiziranemu  
pridobivanju in grafovski analizi  
podatkov omrežne infrastrukture  
National Grid**

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM  
PRVE STOPNJE  
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: izr. prof. dr. Matjaž Kukar

Ljubljana, 2026

To delo je ponujeno pod licenco *Creative Commons Priznanje avtorstva-Deljenje pod enakimi pogoji 2.5 Slovenija* (ali novejšo različico). To pomeni, da se tako besedilo, slike, grafi in druge sestavine dela kot tudi rezultati diplomskega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobčujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani [creativecommons.si](http://creativecommons.si) ali na Inštitutu za intelektualno lastnino, Streliška 1, 1000 Ljubljana.



Izvorna koda diplomskega dela, njeni rezultati in v ta namen razvita programska oprema je ponujena pod licenco GNU General Public License, različica 3 (ali novejša). To pomeni, da se lahko prosto distribuira in/ali predeluje pod njenimi pogoji. Podrobnosti licence so dostopne na spletni strani <http://www.gnu.org/licenses/>.

*Besedilo je oblikovano z urejevalnikom besedil  $\text{\LaTeX}$ .*

**Kandidat:** Dominik Uršič

**Naslov:** Hibridni pristop k avtomatiziranemu pridobivanju in grafovski analizi podatkov omrežne infrastrukture National Grid

**Vrsta naloge:** Diplomaska naloga na univerzitetnem programu prve stopnje Računalništvo in informatika

**Mentor:** izr. prof. dr. Matjaž Kukar

**Opis:**

Diplomska naloga predstavlja razvoj avtomatiziranega ETL sistema za pridobivanje in obdelavo podatkov o električni infrastrukturi operaterja National Grid v Združenem kraljestvu. Sistem avtomatizirano prenaša javno dostopne Excel datoteke, jih arhivira v Google Cloud Storage ter procesira s Python skriptami. Obdelani podatki se naložijo v PostgreSQL bazo z Apache AGE grafovsko razširitvijo, kar omogoča hibridne relacijske in grafovske analize omrežne strukture. Celoten proces je avtomatiziran z Google Cloud Scheduler ter vključuje mehanizme validacije, nadzora kakovosti podatkov in sledljivosti. Naloga demonstrira praktično uporabnost sistema z analitičnimi poizvedbami za optimizacijo omrežnih povezav in izračun prenosnih izgub, izvedenimi v SQL in Cypher jezikih.

**Title:** A hybrid approach to automated data acquisition and graph-based analysis of National Grid network infrastructure

**Description:**

The thesis presents the development of an automated ETL system for acquiring and processing electrical infrastructure data from the National Grid operator in the United Kingdom. The system automatically downloads publicly available Excel files, archives them in Google Cloud Storage, and processes them using Python scripts. The processed data is loaded into a PostgreSQL database with Apache AGE graph extension, enabling hybrid relational and

graph-based analyses of network structure. The entire process is automated using Google Cloud Scheduler and includes mechanisms for validation, data quality control, and traceability. The thesis demonstrates the practical applicability of the system through analytical queries for network connection optimization and transmission loss calculations, implemented in both SQL and Cypher languages.

*Zahvaljujem se mentorju izr. prof. dr. Matjažu Kukarju za strokovno vodenje in podporo pri izdelavi diplomske naloge. Posebna zahvala gre tudi družini in prijateljem za razumevanje in spodbudo v času študija.*



# Kazalo

**Povzetek**

**Abstract**

<b>1</b>	<b>Uvod</b>	<b>1</b>
1.1	Motivacija . . . . .	1
1.2	Opredelitev problema . . . . .	2
1.3	Namen in cilji naloge . . . . .	3
1.4	Pričakovane koristi in deležniki . . . . .	4
1.5	Struktura dokumenta . . . . .	5
<b>2</b>	<b>Teoretično ozadje in pregled področja</b>	<b>7</b>
2.1	Zgodovina spletnega strganja podatkov in izbira orodja . . . . .	7
2.2	Javna električna infrastruktura v Združenem kraljestvu . . . . .	9
2.3	National Grid API . . . . .	9
2.4	Spletno strganje podatkov . . . . .	10
2.5	Selenium WebDriver . . . . .	10
2.6	Podatkovni cevovodi . . . . .	11
2.7	Računalništvo v oblaku . . . . .	12
2.8	Relacijske baze podatkov . . . . .	12
2.9	PostgreSQL in delo z grafi . . . . .	13
<b>3</b>	<b>Metodologija in zasnova sistema</b>	<b>15</b>
3.1	Identifikacija vira podatkov . . . . .	15

3.2	Arhitektura sistema . . . . .	16
3.3	Izbira orodij . . . . .	16
3.4	Kontrola kakovosti podatkov . . . . .	18
3.5	Podatkovne baze in pogoni s podporo za grafe . . . . .	19
3.6	Izbira tehnologije . . . . .	21
<b>4</b>	<b>Implementacija sistema</b>	<b>23</b>
4.1	Koraki delovanja sistema . . . . .	24
4.2	Pridobivanje in shranjevanje podatkov . . . . .	37
4.3	Obdelava podatkov . . . . .	38
4.4	Vizualizacija in analiza . . . . .	38
4.5	Avtomatizacija in nadzor . . . . .	53
4.6	Kontrola kakovosti podatkov . . . . .	53
<b>5</b>	<b>Rezultati in evalvacija</b>	<b>55</b>
5.1	Merila uspešnosti . . . . .	55
5.2	Način evalvacije . . . . .	55
5.3	Kriteriji uspeha . . . . .	56
5.4	Rezultati in evalvacija . . . . .	56
<b>6</b>	<b>Zaključek</b>	<b>63</b>
6.1	Zaključki . . . . .	63
6.2	Možnosti nadaljnjega razvoja . . . . .	65
	<b>Literatura</b>	<b>67</b>



# Seznam uporabljenih kratic

kratica	angleško	slovensko
<b>ETL</b>	Extract, Transform, Load	izlušči, preoblikuj, naloži
<b>GCP</b>	Google Cloud Platform	platforma Google Cloud
<b>GCS</b>	Google Cloud Storage	shramba Google Cloud
<b>NG</b>	National Grid	državno omrežje
<b>DNO</b>	Distribution Network Operator	operater distribucijskega omrežja
<b>GSP</b>	Grid Supply Point	napajalna točka omrežja
<b>BSP</b>	Bulk Supply Point	glavna napajalna točka
<b>PRIM</b>	Primary Substation	primarna transformatorska postaja
<b>AGE</b>	Apache Graph Extension	razširitev Apache Graph
<b>Bucket</b>	Bucket	sektor
<b>Scheduler</b>	Scheduler	razporejevalnik



# Povzetek

**Naslov:** Hibridni pristop k avtomatiziranemu pridobivanju in grafovski analizi podatkov omrežne infrastrukture National Grid

**Avtor:** Dominik Uršič

Cilj diplomske naloge je razvoj in implementacija avtomatiziranega ETL sistema za pridobivanje, obdelavo in shranjevanje podatkov o javni električni infrastrukturi operaterja National Grid v Združenem kraljestvu, z implementacijo hibridnega relacijsko-grafovskega pristopa k analizi omrežne strukture. Sistem bo redno prenašal javno dostopne Excel datoteke s spletne platforme National Grid, jih arhiviral v Google Cloud Storage za zagotavljanje zgodovinske sledljivosti, ter jih procesiral s Python skriptami za čiščenje, validacijo in transformacijo podatkov. Posebna pozornost bo namenjena kazalniku razpoložljive kapacitete (Demand Headroom), ki predstavlja razliko med zanesljivo nosilnostjo omrežnega elementa in njegovo pričakovano najvišjo obremenitvijo ter tako določa preostalo zmogljivost pred potrebnimi infrastrukturnimi nadgradnjami. Obdelani podatki bodo naloženi v PostgreSQL podatkovno bazo, razširjeno z Apache AGE grafovsko nadgradnjo, kar bo omogočalo izvajanje tako tradicionalnih SQL kot tudi grafovskih Cypher poizvedb nad isto podatkovno strukturo. Ta hibridni pristop bo demonstriran z implementacijo analitičnih poizvedb za optimizacijo omrežnih povezav in izračun prenosnih izgub električne energije, izvedenih v obeh pristopih za neposredno primerjavo njihovih prednosti in omejitev. Celoten proces bo v celoti avtomatiziran z uporabo Google Cloud Scheduler, ki bo zagotavljal redno urno izvajanje, mehanizme nadzora kakovosti podatkov, obveščanje o

napakah ter popolno sledljivost vseh operacij. Sistem bo zasnovan modularno in skalabilno, kar bo omogočalo enostavno razširitev na dodatne operaterje distribuirnih omrežij ter integracijo z drugimi podatkovnimi viri v energetskem sektorju.

**Ključne besede:** avtomatizacija pridobivanja podatkov, grafovske baze podatkov, Apache AGE, elektroenergetska infrastruktura, ETL sistem, hibridna analiza.

# Abstract

**Title:** A hybrid approach to automated data acquisition and graph-based analysis of National Grid network infrastructure

**Author:** Dominik Uršič

The objective of this thesis is the development and implementation of an automated ETL system for acquiring, processing, and storing data on public electrical infrastructure of the National Grid operator in the United Kingdom, with implementation of a hybrid relational-graph approach to network structure analysis. The system will regularly download publicly available Excel files from the National Grid platform, archive them in Google Cloud Storage to ensure historical traceability, and process them using Python scripts for data cleaning, validation, and transformation. Special attention will be given to the Demand Headroom indicator, which represents the difference between the reliable capacity of a network element and its expected peak load, thus determining the remaining capacity before infrastructural upgrades are required. The processed data will be loaded into a PostgreSQL database extended with the Apache AGE graph extension, enabling the execution of both traditional SQL and graph-based Cypher queries on the same data structure. This hybrid approach will be demonstrated through the implementation of analytical queries for network connection optimization and electrical transmission loss calculations, executed in both approaches for direct comparison of their advantages and limitations. The entire process will be fully automated using Google Cloud Scheduler, which will ensure regular hourly execution, data quality control mechanisms, error notifications, and complete

traceability of all operations. The system will be designed in a modular and scalable manner, enabling easy extension to additional distribution network operators and integration with other data sources in the energy sector.

**Keywords:** automated data acquisition, graph databases, Apache AGE, electrical grid infrastructure, ETL system, hybrid analysis.

# Poglavje 1

## Uvod

### 1.1 Motivacija

V sodobnem svetu električna energija predstavlja kritično infrastrukturo, ki omogoča delovanje industrije, gospodinjstev, transporta in digitalnih storitev. Z naraščajočo integracijo obnovljivih virov energije, elektrifikacijo transporta ter razvojem pametnih omrežij postaja zanesljiv dostop do ažurnih podatkov o stanju električne infrastrukture vse bolj ključen za učinkovito načrtovanje, analizo in sprejemanje strateških odločitev na energetskega področju.

V Združenem kraljestvu je National Grid eden vodilnih operaterjev distribucijskega elektroenergetskega omrežja, ki pokriva obsežna geografska območja, vključno z regijami East Midlands, West Midlands, South Wales ter South West England. Kot ključni distribucijski operater (DNO) je National Grid odgovoren za vzdrževanje in upravljanje srednjenapetostnega omrežja na svojem območju ter zagotavlja stabilno in zanesljivo oskrbo z električno energijo za več milijonov gospodinjstev in podjetij.

National Grid redno objavlja podatke o stanju svoje infrastrukture v obliki javno dostopnih Excel datotek na svoji spletni platformi. Ti podatki vključujejo obsežne informacije o transformatorskih postajah, napetostnih nivojih, geografskih lokacijah ter ključnem kazalniku razpoložljive kapacitete

(Demand Headroom), ki določa preostalo zmogljivost posameznih omrežnih elementov. Kljub javni dostopnosti pa je njihovo ročno pridobivanje, organizacija in posodabljanje časovno zamudno, podvrženo človeški napaki ter ne omogoča zgodovinske sledljivosti sprememb v infrastrukturi.

Pomanjkanje avtomatiziranih sistemov za redno zajemanje in procesiranje teh podatkov predstavlja oviro za različne deležnike v energetske sektorju. Razvijalci projektov obnovljivih virov energije potrebujejo ažurne informacije o razpoložljivih kapacitetah za optimalno lociranje sončnih elektrarn in vetrnih parkov. Načrtovalci polnilne infrastrukture za električna vozila morajo identificirati lokacije z zadostno omrežno kapaciteto za priključitev hitrih polnilnic. Energetska podjetja in svetovalne agencije pa potrebujejo celovit pregled nad stanjem omrežja za strateško načrtovanje investicij in storitev.

## 1.2 Opredelitev problema

Trenutno pridobivanje podatkov o električni infrastrukturi National Grid poteka pretežno ročno, kar vključuje mesečno ali kvartalno odpiranje spletnih strani, iskanje ustreznih datotek, njihov prenos ter ročno vnašanje v lokalne evidence ali analitična orodja. Ta proces običajno zahteva 2-4 ure kvalificiranega dela na mesec in je podvržen več ključnim težavam.

Prvič, odsotnost zgodovinske sledljivosti predstavlja pomembno omejitvev. Spletna platforma National Grid prikazuje zgolj najnovejše verzije datotek brez arhiviranja predhodnih stanj, kar onemogoča analizo časovnih sprememb v razpoložljivih kapacitetah, prepoznavanje trendov obremenitev ali longitudinalne študije razvoja omrežja. To otežuje dolgoročno načrtovanje infrastrukturnih investicij ter identifikacijo sistematičnih vzorcev v rasti elektroenergetskih potreb.

Drugič, pomanjkanje centralizirane in strukturirane podatkovne baze onemogoča integracijo informacij iz različnih virov ter izvajanje kompleksnih analiz, ki bi zahtevale kombiniranje podatkov o omrežni infrastrukturi z drugimi relevantnimi viri, kot so demografski trendi, prostorski načrti, vremenske



napovedi ali projekcije rasti obnovljivih virov energije.

Tretjič, ročno procesiranje podatkov ne zagotavlja konsistentne kakovosti, saj lahko različni uporabniki uporabljajo različne metodologije čiščenja in transformacije podatkov, kar otežuje primerjavo rezultatov ter sodelovanje med organizacijami.

Četrto, električna infrastruktura je inherentno omrežne narave z hierarhičnimi povezavami med različnimi napetostnimi nivoji (GSP  $\rightarrow$  BSP  $\rightarrow$  PRIM), vendar tradicionalne relacijske baze niso optimizirane za predstavitev in analizo takšnih grafovskih struktur. To otežuje izvajanje omrežnih analiz, kot so iskanje optimalnih poti, identifikacija kritičnih vozlišč ali simulacije kaskadnih izpadov.

### 1.3 Namen in cilji naloge

Primarni namen sistema je avtomatizirati zajem javno dostopnih podatkov National Grid Electricity Distribution ter zagotoviti standardizirano, časovno označeno in sledljivo kopijo podatkov za nadaljnje analize. Sistem zmanjšuje ročno delo ter možnost napak pri ročnem prenosu podatkov, hkrati pa uvaža verzioniranje za popolno sledljivost sprememb v infrastrukturi skozi čas.

Glavni cilj naloge je implementacija robustnega ETL (Extract, Transform, Load) sistema, ki bo avtomatizirano prenašal javno dostopne Excel datoteke s spletne platforme National Grid, jih arhiviral v Google Cloud Storage staging okolju za zagotavljanje zgodovinske sledljivosti, ter jih procesiral s Python skriptami za čiščenje, validacijo in transformacijo v standardizirane formate.

Obdelani podatki bodo naloženi v PostgreSQL podatkovno bazo, razširjeno z Apache AGE grafovsko nadgradnjo, kar bo omogočalo hibridni pristop k analizi podatkov. Ta pristop kombinira prednosti relacijskih baz (učinkovitost, zrelost, SQL ekosistem) s prednostmi grafovskih baz (intuitivno modeliranje omrežnih struktur, podporo za Cypher poizvedbe, omrežne algoritme).

Posebna pozornost bo namenjena kazalniku razpoložljive kapacitete (De-

mand Headroom), ki predstavlja ključno informacijo za načrtovanje novih priključitev in infrastrukturnih nadgradenj. Sistem bo omogočal spremljanje časovnih sprememb tega kazalnika ter identifikacijo območij z nizko razpoložljivo kapaciteto, kar je kritičnega pomena za načrtovalce novih objektov elektroinfrastrukture, kot so lokacije električnih polnilnic.

Celoten proces bo v celoti avtomatiziran z uporabo Google Cloud Scheduler, ki bo zagotavljal redno urno izvajanje v poslovnem času, mehanizme za validacijo kakovosti podatkov, obveščanje o napakah ter popolno sledljivost vseh operacij preko strukturiranega beleženja dogodkov.

Dodatni cilj naloge je demonstracija praktične uporabnosti hibridnega relacijsko-grafovskega pristopa z implementacijo dveh kompleksnih analiz: optimizacija dodelitev primarnih transformatorskih postaj za minimizacijo prenosnih razdalj ter izračun prenosnih izgub električne energije na različnih segmentih omrežja. Vsaka analiza bo izvedena z uporabo tako SQL kot Cypher pristopa, kar bo omogočilo neposredno primerjavo obeh metodologij z vidika berljivosti, zmogljivosti in praktične uporabnosti.

## 1.4 Pričakovane koristi in deležniki

Implementacija avtomatiziranega sistema za pridobivanje podatkov o električni infrastrukturi bo prinesla oprijemljive koristi različnim deležnikom v energetske sektorju. Sistem bo zmanjšal časovne zahteve za pridobivanje podatkov iz trenutnih 2-4 ur mesečnega ročnega dela na nekaj minut avtomatiziranega procesa, kar predstavlja neposreden prihranek delovnega časa kvalificiranih kadrov ter zmanjšanje možnosti človeških napak pri prenosu in obdelavi podatkov.

Sistem zagotavlja trdno analitično podlago za taktično in strateško odločanje v energetske sektorju. Ključni deležniki sistema so energetska podjetja in razvijalci projektov, ki bodo pridobili takojšen dostop do ažurnih podatkov o razpoložljivih kapacitetah (Demand Headroom), kar bo omogočilo hitrejša in bolj informirane odločitve o lokacijah novih projektov. Svetovalna podjetja

v energetske sektorju bodo lahko svojim strankam ponudila natančnejše analize in hitreje pripravila študije izvedljivosti, medtem ko bodo investitorji v obnovljive vire energije pridobili kritične informacije za oceno primernosti lokacij za solarne elektrarne, vetrne parke ali baterijske sisteme.

Operaterji omrežja in načrtovalci infrastrukture bodo lahko uporabljali sistem za prepoznavanje ozkih grl v omrežju, načrtovanje nadgradenj ter optimizacijo razporeditve novih transformatorskih postaj. Javne ustanove in regulatorji bodo imeli na voljo konsistenten vir podatkov za spremljanje razvoja elektroenergetskega sistema ter pripravo strateških smernic za energetske tranzicije. Raziskovalci in svetovalni strokovnjaki pa bodo lahko izvajali napredne analize trendov porabe, napovedovanje prihodnjih kapacitet ter simulacije različnih scenarijev razvoja omrežja.

Strukturirana zgodovinska baza podatkov bo omogočila napredno analitiko za prepoznavanje trendov porabe, napovedovanje prihodnjih kapacitet in identifikacijo kritičnih točk v omrežju, kar bo podprlo strateško načrtovanje investicij v energetske infrastrukturo in pospešilo prehod na trajnostne vire energije. Dolgoročno bo sistem omogočil enostavno razširitev na dodatne distributerje električne energije po celotni Veliki Britaniji, saj bo vzpostavljena arhitektura zlahka prilagodljiva za integracijo podatkov iz UK Power Networks, Scottish Power Energy Networks in drugih operaterjev.

## 1.5 Struktura dokumenta

Drugo poglavje predstavlja teoretično ozadje s pregledom ETL procesov, pristopov k avtomatizaciji pridobivanja podatkov ter grafovskih podatkovnih baz, s poudarkom na Apache AGE razširitvi za PostgreSQL in jeziku Cypher.

Tretje poglavje opisuje metodologijo in arhitekturo sistema, vključno s celotno strukturo ETL cevovoda, odločitvami pri izbiri tehnologij ter zasnovi podatkovnega modela z grafovsko shemo.

Četrto poglavje podrobno predstavlja implementacijo posameznih komponent, od Python skript za obdelavo podatkov, konfiguracije Google Cloud

Storage in Cloud Scheduler, do strukture relacijskih tabel in grafovskega modela ter mehanizmov kontrole kakovosti.

Peto poglavje vsebuje rezultate avtomatizacije, implementacije analitičnih poizvedb v SQL in Cypher pristopih, kvantitativno primerjavo obeh pristopov ter vizualizacije ugotovitev o optimizaciji omrežnih povezav in prenosnih izgubah.

Šesto poglavje povzema ključne ugotovitve, evalvira doseganje zastavljenih ciljev ter opredeljuje možnosti nadaljnjega razvoja sistema.

## Poglavje 2

# Teoretično ozadje in pregled področja

### 2.1 Zgodovina spletnega strganja podatkov in izbira orodja

Avtomatizirano zajemanje podatkov s spletnih strani se je v zadnjih dvajsetih letih precej spremenilo, ker so se spremenile tudi same spletne strani. V začetkih interneta so bile strani večinoma statične v obliki HTML dokumenta, zato je za pridobivanje podatkov zadostovala preprosta analiza izvirne kode. Takrat so bile v uporabi predvsem rešitve, ki so temeljile na regularnih izrazih in osnovnem razčlenjevanju HTML strukture. Pomemben mejnik je predstavljal razvoj specializiranih knjižnic za delo z HTML dokumenti. Leta 2004 je prišel Beautiful Soup, ki je spletno strganje naredil veliko bolj dostopno. Ta Python knjižnica je z intuitivno sintakso omogočila enostavno ekstrakcijo podatkov iz kompleksnih HTML struktur, vendar je bila še vedno omejena na statično vsebino. V tem času so se podobne knjižnice razvijale tudi za druge programske jezike (jsoup za Javo).

Leta 2008 je Scrapy prinesel nov pristop s svojo asinhrono arhitekturo, ki je omogočila učinkovito obdelavo velikega števila strani hkrati. To Python ogrodje je postalo nekakšen standard za večje projekte spletnega strganja, saj

je omogočalo distribuirano delo, avtomatsko upravljanje s piškotki in sejami ter robustno obravnavo napak. Takrat pa se je začel tudi velik premik v spletnem razvoju. Z uvedbo AJAX tehnologij in Single Page Applications (SPA) so spletne strani postale veliko bolj dinamične. Vsebina se je začela nalagati asinhrono, elementi so se generirali z JavaScript kodo, podatki pa so se pridobivali preko API klicev šele po začetnem nalaganju strani. Tradicionalne metode strganja naenkrat niso več zadostovale.

Rešitev je prišla z orodji za avtomatizacijo brskalnikov. Selenium WebDriver, ki je bil sicer prvotno razvit za testiranje spletnih aplikacij, se je izkazal za odlično orodje tudi za strganje kompleksnih strani. Za razliko od prejšnjih pristopov Selenium upravlja pravi brskalnik – lahko izvaja JavaScript, čaka na dinamično naložene elemente, simulira klike in druge uporabniške interakcije ter se spopada s kompleksnimi navigacijskimi tokovi. Selenium deluje preko WebDriver protokola, ki ga je leta 2018 standardiziral W3C konzorcij. Ta protokol omogoča komunikacijo med programsko kodo in brskalnikom na način, ki deluje prek različnih brskalnikov (Chrome, Firefox, Safari, Edge) in operacijskih sistemov. Osnova je client-server model, kjer aplikacija pošilja ukaze gonilniku brskalnika, ta pa jih izvaja in vrača rezultate.

Za projekt avtomatizacije National Grid platforme je bil Selenium najboljša izbira iz več razlogov. Platforma zahteva avtentikacijo preko prijavnega obrazca, uporablja dinamično nalaganje vsebine, vključuje interaktivne zemljevide in vizualizacije ter ima tudi določene zaščitne mehanizme proti avtomatizaciji. Zato smo uporabili še undetected-chromedriver, ki z različnimi tehnikami (modifikacija navigator objekta, odstranjevanje WebDriver zastavic, simulacija realističnih vzorcev gibanja miške) poskrbi, da sistem ne zazna avtomatizacije. Prehod od BeautifulSoup preko Scrapy do Selenium tako zrcali razvoj spletnih tehnologij. Medtem ko enostavnejša orodja še vedno dobro služijo za statične strani, kompleksne moderne aplikacije zahtevajo polno simulacijo brskalnika. Selenium z zmožnostjo izvajanja JavaScript kode, čakanja na asinhrono naložene elemente in simulacije uporabniških in-

terakcij trenutno predstavlja najzmogljivejšo rešitev za avtomatizirano pridobivanje podatkov iz sodobnih spletnih platform.

## 2.2 Javna električna infrastruktura v Združenem kraljestvu

Združeno kraljestvo ima kompleksen sistem električne infrastrukture, kjer različni akterji upravljajo prenos in distribucijo električne energije. Država je razdeljena na 14 različnih geografskih območij za katera je odgovornih 6 podjetij. National Grid je eden glavnih operaterjev prenosnega omrežja, ki povezuje elektrarne z distribucijskimi omrežji [10].

National Grid redno objavlja podatke o zmogljivostih omrežja, vključno s podatki o "Demand Headroom" parametru, ki opisuje razpoložljivo kapaciteto omrežja za nove priključitve. Ti podatki so ključni za energetske analize, načrtovanje infrastrukture in sprejemanje poslovnih odločitev. [8]

## 2.3 National Grid API

API Connected Data Portal podjetja National Grid predstavlja standardiziran vmesnik za programski dostop do javno objavljenih energetskih in omrežnih podatkov. Kljub temu, da pokriva podatkovno domeno, ki se vsebinsko delno prekriva z obravnavanim področjem naloge, sistem izkazuje ključne strukturne omejitve, ki onemogočajo njegovo učinkovito implementacijo v kontekstu obravnavanih podatkov. Fundamentalna omejitev izhaja iz vnaprej definirane arhitekture podatkovnih nizov, pri čemer ponudnik določa tako obseg kot strukturo razpoložljivih podatkov. API v svoji trenutni konfiguraciji ne omogoča fleksibilnega prilagajanja podatkovnih zahtevkov specifičnim potrebam posameznega primera uporabe, temveč zgolj izpostavlja že obstoječe podatkovne nize v fiksni obliki, kot so bili izvirno objavljeni.

## 2.4 Spletno strganje podatkov

Spletno strganje (web scraping) je tehnika avtomatskega pridobivanja podatkov s spletnih strani [9]. V Pythonu obstajajo različne knjižnice, sam pa bom uporabil Selenium, ki omogoča programsko upravljanje spletnega brskalnika in interakcijo z dinamičnimi spletnimi stranmi. Selenium deluje tako, da simulira dejanja pravega uporabnika v brskalniku, lahko klikne gumbe, izpolni obrazce, počaka na nalaganje elementov in izvozi podatke. Proces se tipično začne z inicializacijo brskalnika (v našem primeru Chromium z undetected-chromedriver za izogibanje detekciji avtomatizacije), nato pa skript sistematično navigira po spletni strani. Najprej se izvede prijava z vnosom uporabniškega imena in gesla, sledi navigacija do želenega dela aplikacije, kjer se sproži izvoz podatkov. Posebni funkciji (WebDriverWait in expected-conditions) zagotavljata, da skript počaka na popolno nalaganje elementov, preden z njimi upravlja, kar preprečuje napake zaradi asinhronega nalaganja vsebine. Ko je datoteka prenesena, se avtomatsko preimenuje s časovnim žigom in shrani v določeno mapo za nadaljnjo obdelavo. Pri spletnem strganju je pomembno upoštevati etične in pravne vidike, vključno s spoštovanjem robots.txt datotek, omejitev frekvence zahtev in pogojev uporabe spletnih strani [4]. V našem primeru gre za pridobivanje javno dostopnih podatkov z uporabo legitimnih prijavnih podatkov, kar zagotavlja skladnost s pogoji uporabe National Grid platforme.

## 2.5 Selenium WebDriver

Selenium WebDriver je odprtokodno orodje za avtomatizacijo spletnih brskalnikov, ki omogoča programsko interakcijo s spletnimi stranmi [13]. Temelji na arhitekturi tipa odjemalec–strežnik: aplikacija pošilja ukaze brskalniku prek protokola WebDriver, brskalnikov gonilnik pa ta navodila izvede in vrne rezultat. Ko Python skripta pokliče Seleniumovo metodo, se ta pretvori v HTTP zahtevo, ki jo gonilnik brskalnika (na primer ChromeDriver za Google Chrome) izvede neposredno v uporabniškem vmesniku.



Selenium omogoča različne načine za iskanje elementov na spletni strani, kot so uporaba identifikatorjev, CSS selektorjev, izrazov XPath ali imen razredov. Z uporabo mehanizmov WebDriverWait in expected conditions lahko skripta eksplicitno počaka, da se določen pogoj izpolni (na primer, da element postane klikljiv), preden nadaljuje z izvajanjem. Tak pristop je ključen pri dinamičnih spletnih straneh, kjer se vsebina nalaga asinhrono prek JavaScripta.

## 2.6 Podatkovni cevovodi

Podatkovni cevovodi (data pipelines) so avtomatizirani procesi za prenos podatkov od virov do končnih destinacij z možnostjo transformacije med potjo [15]. Tradicionalni ETL (Extract, Transform, Load) pristop se v zadnjem času vse bolj nadomešča z ELT (Extract, Load, Transform) pristopom, ki omogoča večjo fleksibilnost pri obdelavi podatkov. Ključna razlika med pristopoma je v zaporedju operacij. Pri ETL pristopu se podatki najprej ekstrahirajo iz vira, nato transformirajo v vmesnem okolju in šele nato naložijo v ciljno podatkovno bazo. Nasprotno pa ELT pristop najprej naloži surove podatke neposredno v podatkovno bazo, kjer se transformacije izvajajo z uporabo SQL poizvedb in drugih orodij znotraj samega RDBMS sistema. V našem sistemu implementiramo klasični ETL pristop, ki se je izkazal za najbolj primerne glede na naravo podatkov in zahteve sistema. Uporaba ETL pristopa pozitivno vpliva na zmogljivost RDBMS sistema, saj PostgreSQL prejme le čiste, validirane podatke, kar zmanjšuje potrebo po kompleksnih SQL transformacijah in s tem obremenitev podatkovne baze. To omogoča, da se PostgreSQL osredotoči na svoje primarne naloge, učinkovito shranjevanje, indeksiranje in serviranje podatkov končnim uporabnikom. Manjša obremenitev baze pomeni hitrejša odzivna časa pri poizvedbah, nižjo porabo sistemskih virov in večjo skalabilnost sistema. Dodatno ETL pristop omogoča lažje odkrivanje in reševanje napak v podatkih, saj se te obravnavajo še pred vnosom v produkcijsko bazo, kar zagotavlja večjo integriteto podatkov in

zanesljivost celotnega sistema.

## 2.7 Računalništvo v oblaku

Google Cloud Platform (GCP) ponuja različne storitve za delo s podatki, vključno z Google Cloud Storage za shranjevanje datotek in Google Cloud Scheduler za avtomatizirano izvajanje opravil [7]. Te storitve omogočajo skalabilno in zanesljivo infrastrukturo za podatkovne cevovode.

## 2.8 Relacijske baze podatkov

PostgreSQL je zmogljiva odprtokodna objektno-relacijska podatkovna baza, ki se pogosto uporablja za shranjevanje strukturiranih podatkov v podatkovnih aplikacijah [11]. Omogoča kompleksne poizvedbe, ACID transakcije, različne razširitve za specifične potrebe ter napredno indeksiranje. Za naš sistem avtomatiziranega pridobivanja podatkov o električni infrastrukturi so ključne funkcionalne zahteve PostgreSQL sistema naslednje: podpora za velike količine časovnih serij podatkov (preko 1300 zapisov, seveda lahko PostgreSQL obdela še veliko več zapisov), zmožnost hitrega vstavljanja novih podatkov preko bulk **INSERT** operacij, učinkovito indeksiranje na polju Demand Headroom za hitre poizvedbe, podpora za JSON podatkovne tipe za shranjevanje semi-strukturiranih metapodatkov, ter zmožnost izvajanja kompleksnih analitičnih poizvedb z window funkcijami. Sistem mora zagotavljati tudi verzioniranje podatkov, kjer se ohranjajo vse zgodovinske verzije za revizijske sledi in analizo trendov. Pomembna je tudi konfiguracija avtomatskega vzdrževanja preko autovacuum procesa, ki zagotavlja optimalno zmogljivost tudi pri velikem številu **UPDATE** in **DELETE** operacij. Dodatno mora sistem podpirati replikacijo za visoko razpoložljivost, omogočati point in time recovery za zaščito pred izgubo podatkov, ter imeti nastavljeno redno varnostno kopiranje (pg dump) vsaj enkrat dnevno. PostgreSQL razširitve kot so pg cron za avtomatizirane naloge znotraj baze in timescaledb za optimizirano

delo s časovnimi serijami dodatno izboljšajo funkcionalnost sistema za naše specifične potrebe pri upravljanju podatkov.

## 2.9 PostgreSQL in delo z grafi

Poleg klasičnih relacijskih podatkovnih modelov se v sodobnih podatkovnih sistemih vse pogostejše pojavlja potreba po obdelavi grafovskih struktur, kjer so podatki predstavljeni kot vozlišča (ang. *nodes*) in povezave med njimi (ang. *edges*). Takšen pristop je posebej primeren za modeliranje omrežij, kot so energetska infrastruktura, prometna omrežja ali socialne mreže, kjer relacije med entitetami nosijo enako pomembno informacijo kot same entitete [1].

PostgreSQL kljub temu, da primarno sodi med relacijske podatkovne baze, omogoča učinkovito delo z grafi na več različnih načinov. Osnovni pristop temelji na modeliranju grafovskih struktur z uporabo relacijskih tabel, kjer se vozlišča hranijo v eni tabeli, povezave pa v drugi tabeli z referencami (tujimi ključi) na izvirno in ciljno vozlišče. Takšen model omogoča uporabo standardnih SQL poizvedb za osnovne grafovske operacije, kot so iskanje sosedov, stopnje vozlišč in enostavne poti [5].

Za zahtevnejše grafovske analize PostgreSQL ponuja podporo z razširitvami. Ena najpomembnejših je *Apache AGE*, ki razširja PostgreSQL z lastnostmi večmodelne baze podatkov in dodaja podporo za grafovni podatkovni model ter poizvedovalni jezik *openCypher*. *Apache AGE* omogoča izvajanje kompleksnih grafovskih poizvedb, kot so iskanje najkrajših poti, detekcija povezanih komponent in analiza omrežnih vzorcev, neposredno znotraj PostgreSQL okolja, brez potrebe po ločeni grafovni bazi podatkov.

Alternativni pristop predstavlja razširitev *pgRouting*, ki je specializirana za delo z grafi v prostorskih podatkih in se pogosto uporablja v kombinaciji z razširitvijo *PostGIS*. Čeprav je primarno namenjena prometnim in geografskim omrežjem, se lahko njeni algoritmi za iskanje poti (Dijkstra, A\*, Bellman-Ford) uporabijo tudi za analizo energetskih ali infrastrukturnih

omrežij, kjer so povezave utežene z zmogljivostmi ali obremenitvami.

V kontekstu električne infrastrukture je grafovski model posebej primeren za predstavitev prenosnega in distribucijskega omrežja. Vozlišča lahko predstavljajo transformatorske postaje, razdelilne točke ali geografska območja, povezave pa fizične ali logične povezave med njimi. Atributi povezav, kot so maksimalna zmogljivost, trenutna obremenitev ali razpoložljiv *Demand Headroom*, omogočajo izvajanje analitičnih poizvedb, ki podpirajo odločanje pri načrtovanju novih priključitev ali nadgradenj omrežja.

Uporaba PostgreSQL za delo z grafi prinaša pomembno prednost v obliki enotne podatkovne platforme. Relacijski, časovni in grafovski podatki so shranjeni v istem sistemu, kar poenostavi arhitekturo, zmanjša operativne stroške in omogoča kombiniranje klasičnih SQL poizvedb z grafovskimi analizami. Takšen hibridni pristop je posebej primeren za sisteme, kjer grafovske analize dopolnjujejo obstoječe relacijske in časovne podatkovne modele.

## Poglavje 3

# Metodologija in zasnova sistema

### 3.1 Identifikacija vira podatkov

Glavni vir podatkov je spletna stran National Grid, kjer so objavljene Excel datoteke z informacijami o zmogljivostih omrežja. URL za dostop do podatkov je <https://www.nationalgrid.co.uk/our-network/network-capacity-map-application>.

Datoteke vsebujejo nabor tehničnih parametrov električne infrastrukture, strukturiranih v CSV formatu. Ključna polja vključujejo Substation Name (ime postaje), Asset Type (vrsto postaje), ter koordinate lokacij (Latitude in Longitude) za lažje geografsko pozicioniranje. Posebno pozornost namenimo polju Demand Headroom (MW), ki predstavlja razpoložljivo kapaciteto za nove priključitve in je zelo pomeben parameter za razvijalce projektov pri ocenjevanju izvedljivosti novih povezav. Dodatni tehnični parametri vključujejo Peak Demand (najvišja obremenitev) in Network Reference ID, ki omogočajo enolično identifikacijo vsake lokacije v nacionalnem omrežju.

## 3.2 Arhitektura sistema

Sistem bo implementiran po ETL principu, pri čemer bomo dodali še vmesno staging fazo v oblaku za večjo zanesljivost in sledljivost procesov. [14] V prvi fazi ekstrakcije uporabljamo Python skripte s Selenium avtomatizacijo, ki se povežejo na spletno stran National Grid in prenesejo CSV datoteke. Ta del deluje skoraj kot pravi uporabnik, saj simulira klike, čaka na nalaganje dinamičnih elementov in uporablja vse interaktivne komponente na strani. Ker gre za kompleksno spletno aplikacijo z JavaScript generiranimi elementi, je ta pristop nujen. Surovi podatki nato ne gredo direktno v bazo, ampak jih najprej pošljemo v Google Cloud Storage, kjer poteka vmesna transformacija. V tej staging fazi naredimo osnovno validacijo podatkov, počistimo manjkajoče vrednosti in standardiziramo formate. Te delno transformirane podatke shranimo kot staging datoteke, kar nam omogoča, da se lahko kadarkoli vrnemo nazaj in pogledamo, kako so podatki izgledali v določenem trenutku. To je zelo koristno, če naletimo na kakšne težave ali potrebujemo ponovno procesiranje. Šele nato pride na vrsto končna transformacija, ki jo izvajamo s Pandas knjižnico. Ko je vse pripravljeno, podatke naložimo v PostgreSQL podatkovno bazo, kjer so nato na voljo za analizo in uporabo. Ta pristop z vmesno staging fazo v GCP nam zagotavlja večjo zanesljivost celotnega sistema. Če kdaj pride do napake v katerikoli fazi, imamo vedno shranjene vmesne rezultate in lahko proces ponovno poženemo od točke, kjer se je nekaj zalomilo. To je še posebej pomembno pri avtomatiziranih sistemih, kjer ni vedno nekoga, ki bi takoj opazil težavo. Celoten proces bo seveda avtomatiziran, uporabili bomo Google Cloud Scheduler, ki bo skripte zaganjal periodično po vnaprej določenem urniku. Tako bo sistem deloval samostojno in podatki se bodo osvežili brez kakršnegakoli ročnega posredovanja.

## 3.3 Izbira orodij

Za implementacijo sistema smo izbrali Python 3.11, predvsem zaradi njegovega bogatega ekosistema knjižnic in odlične podpore za avtomatizacijo.

Python se je v zadnjih letih uveljavil kot eden vodilnih jezikov za delo s podatki, kar se odraža tudi v razpoložljivosti kvalitetnih orodij za naše potrebe. Pri izbiri knjižnic smo kombinirali preverjene standardne rešitve s prilagojenimi komponentami. Za interakcijo z brskalnikom uporabljamo Selenium v kombinaciji z undetected-chromedriver, ki nam omogoča upravljanje z brskalnikom na način, da se izognemo detekciji avtomatizacije. To je zelo pomembno, saj večina sodobnih spletnih platform implementira zaščitne mehanizme proti avtomatiziranim dostopom. [3] Za delo s podatki se zanašamo na Pandas, ki je praktično postal standard za branje, transformacijo in obdelavo tabelaričnih podatkov v Pythonu. Ta knjižnica nam omogoča učinkovito delo z CSV datotekami in izvajanje kompleksnih transformacij podatkov. Za komunikacijo z Google Cloud Storage uporabljamo uradno google-cloud-storage knjižnico, ki poskrbi za vso interakcijo s cloudnim shranjevanjem in upravljanje staging datotek. Za beleženje vseh dogodkov in napak uporabljamo vgrajeni logging modul, ki nam omogoča strukturirano spremljanje delovanja sistema. Poleg standardnih knjižnic smo razvili tudi nekaj prilagojenih komponent. **AbstractScriptRunner** je abstraktni razred, ki standardizira izvajanje ETL skript in vključuje vgrajeno logiko za elegantno obravnavo napak. Tako vse naše skripte sledijo enakemu vzorcu izvajanja, kar olajša vzdrževanje in razumevanje kode. Centralizirano konfiguracijo celotnega sistema upravljamo preko config modula, ki vključuje nastavitve za logiranje, povezavo z GCS klientom in podatkovne direktorije. Tako imamo vse nastavitve na enem mestu, kar občutno olajša prilagajanje sistema različnim okoljem. Razvili smo tudi gdutil (Generic Dataset Utilities), nabor prilagojenih funkcij za delo z geografskimi podatkovnimi nizi, ki rešujejo specifične izzive našega projekta.

### 3.3.1 Infrastruktura

Celoten sistem temelji na kombinaciji oblačnih storitev in podatkovne baze, kar nam omogoča fleksibilnost pri shranjevanju in obdelavi podatkov. Za shranjevanje surovih in delno transformiranih datotek uporabljamo Google

Cloud Storage. To je objektno shranjevanje, ki ga ponuja Google Cloud Platform in se je izkazalo za idealno rešitev za naš staging layer. Tukaj se shranjujejo CSV datoteke, ki jih sistem prenese iz National Grid platforme, še preden jih procesiramo in naložimo v bazo. Prednost GCS je v tem, da nam omogoča praktično "neomejeno shranjevanje po relativno nizki ceni, hkrati pa so podatki vedno dostopni in zanesljivo shranjeni. Poleg tega lahko kadarkoli dostopamo do zgodovinskih verzij datotek, če potrebujemo ponovno procesiranje ali analizo, kako so se podatki spreminjali skozi čas. Za končno shranjevanje strukturiranih, obdelanih podatkov pa uporabljamo PostgreSQL podatkovno bazo. PostgreSQL smo izbrali zaradi njegove robustnosti, odlične podpore za kompleksne poizvedbe in geografske podatke preko PostGIS razširitve. Gre za relacijsko bazo, ki omogoča učinkovito indeksiranje in iskanje po podatkih, kar je ključno za kasnejšo analizo in vizualizacijo. Podatki v PostgreSQL so strukturirani v tabele z jasno definiranimi relacijami, kar olajša delo z njimi in zagotavlja integriteto podatkov. Za avtomatizacijo celotnega procesa skrbi Google Cloud Scheduler. To je cron storitev v oblaku, ki omogoča zanesljivo periodično izvajanje naših skript. Nastavimo lahko natančne urnike, kdaj naj se sistem zažene (vsak dan ob določeni uri ali vsak teden v določen dan). Cloud Scheduler je zanesljiv, ne zahteva vzdrževanja strežnika, ki bi moral biti vedno prižgan, in nam pošlje obvestila, če pride do napak pri izvajanju. Tako je celoten ETL proces popolnoma avtomatiziran in deluje brez potrebe po ročnem posredovanju.

### 3.4 Kontrola kakovosti podatkov

Za zagotavljanje zanesljivosti sistema in kakovosti podatkov bomo implementirali več nivojev preverjanj, ki bodo našli potencialne težave že v zgodnjih fazah procesiranja. Naslednja pomembna kontrola je detekcija podvojenih vnosov. Ker sistem deluje periodično in prenašamo podatke redno, obstaja možnost, da bi določeni podatki bili v bazo naloženi večkrat. Sistem bo zato preverjal, ali vnos s kombinacijo ključnih atributov že obstaja v bazi, preden



ga poskuša vstaviti. Tako preprečimo nepotrebno podvajanje in zagotovimo integriteto podatkov. Validacija podatkovnih tipov je ključna za pravilno delovanje celotnega sistema. Sistem bo preveril, ali so numerične vrednosti res številke, ali so datumi v pravilnem formatu in ali besedilna polja ne vsebujejo nepričakovanih znakov ali vsebin. Če naleti na vrednosti, ki ne ustrezajo pričakovanemu tipu, bo zabeležil opozorilo in se odločil, ali lahko vrednost pretvori ali jo mora zavrniti. Posebno pozornost bomo namenili tudi preverjanju in upravljanju manjkajočih vrednosti. Sistem bo identificiral, kateri stolpci imajo manjkajoče podatke in glede na pomembnost polja odločil, kako ravnati. Pri nekritičnih poljih lahko manjkajoče vrednosti nadomestimo z privzetimi vrednostmi ali jih pustimo prazne, medtem ko bodo pri kritičnih poljih manjkajoče vrednosti povzročile zavrnitev celotnega vnosa. Vse te odločitve bodo jasno dokumentirane v logih za kasnejši pregled.

### 3.5 Podatkovne baze in pogoni s podporo za grafe

Z naraščajočo kompleksnostjo podatkov in potrebo po učinkovitem modeliranju odnosov med entitetami so grafno usmerjene podatkovne rešitve postale pomemben del sodobnih podatkovnih arhitektur. Sistemi, ki podpirajo grafe, omogočajo učinkovite večkorakovne poizvedbe, analizo omrežij in modeliranje kompleksnih relacij, kar je pogosto neučinkovito ali neizvedljivo v klasičnih relacijskih bazah. V nadaljevanju so predstavljeni izbrani sistemi, ki bodisi delujejo kot namenske grafne baze bodisi kot razširitve ali poizvedovalni pogoni nad obstoječimi podatkovnimi viri.

#### 3.5.1 PuppyGraph

PuppyGraph ni klasična grafna baza podatkov, temveč *grafni poizvedovalni pogon*, ki omogoča izvajanje grafnih poizvedb neposredno nad obstoječimi podatkovnimi viri, kot so relacijske baze, podatkovna skladišča ali podat-

kovna jezera. Ključna značilnost sistema je arhitektura brez ETL procesov, saj podatkov ni treba kopirati ali transformirati v ločeno grafno shrambo.

PuppyGraph logično preslika obstoječe podatke v grafovni model in podpira standardne grafne poizvedovalne jezike, kot sta openCypher in Gremlin. Tak pristop omogoča hitro uvedbo grafnih analiz v obstoječe sisteme, pri čemer se ohranijo prednosti primarnega podatkovnega vira, kot so konsistentnost, varnost in upravljanje podatkov.

### 3.5.2 AgensGraph

AgensGraph je grafna baza podatkov, ki temelji na relacijski bazi PostgreSQL. Gre za hibridni sistem, ki združuje relacijski in grafni podatkovni model znotraj enotnega podatkovnega strežnika. Uporabnikom omogoča, da v isti podatkovni bazi kombinirajo klasične SQL poizvedbe in grafne operacije.

Zaradi izpeljave iz PostgreSQL AgensGraph podeduje transakcijski model ACID, zanesljivost ter bogat ekosistem orodij. Takšna zasnova je primerna za primere uporabe, kjer je potrebno tesno prepletanje relacijskih in grafnih podatkov brez uvajanja dodatne infrastrukture.

### 3.5.3 RedisGraph

RedisGraph je modul za Redis, ki implementira lastnostni grafni model v pomnilniku. Namenjen je predvsem scenarijem, kjer je ključnega pomena nizka latenca in visoka hitrost poizvedb. Sistem uporablja openCypher kot poizvedovalni jezik ter interno predstavlja graf z uporabo redkih matrik in linearno-algebrskih operacij.

Zaradi in-memory narave je RedisGraph posebej primeren za operativne in realnočasovne grafne primere uporabe, kot so priporočilni sistemi ali analiza omrežij v živo. Slabost pristopa je večja poraba pomnilnika in omejena primernost za zelo velike, trajne grafe.

### 3.5.4 Cayley

Cayley je odprtokodna grafna baza podatkov, prvotno zasnovana po vzoru grafnega sistema Freebase. Poseben poudarek namenja podpori za povezane podatke in ogrodja za opis virov (RDF standarde).

Sistem podpira več poizvedovalnih jezikov in lahko deluje nad različnimi hrambnimi mehanizmi, vključno z relacijskimi in ključ-vrednost bazami. Zaradi svoje prilagodljivosti je Cayley primeren za raziskovalne namene, semantične grafe in znanstvene aplikacije, kjer interoperabilnost podatkov igra pomembno vlogo.

### 3.5.5 Apache AGE

Apache AGE (*A Graph Extension*) je razširitev za PostgreSQL, ki tej relacijski bazi doda podporo za grafni podatkovni model. Namesto ločene grafne baze AGE omogoča shranjevanje vozlišč in povezav znotraj PostgreSQL ter poizvedovanje z uporabo openCypher jezika.

Tak pristop omogoča hibridno uporabo SQL in grafnih poizvedb nad enotnim podatkovnim skladiščem. Apache AGE je še posebej primeren za organizacije, ki že uporabljajo PostgreSQL in želijo grafne funkcionalnosti dodati brez večjih arhitekturnih sprememb.

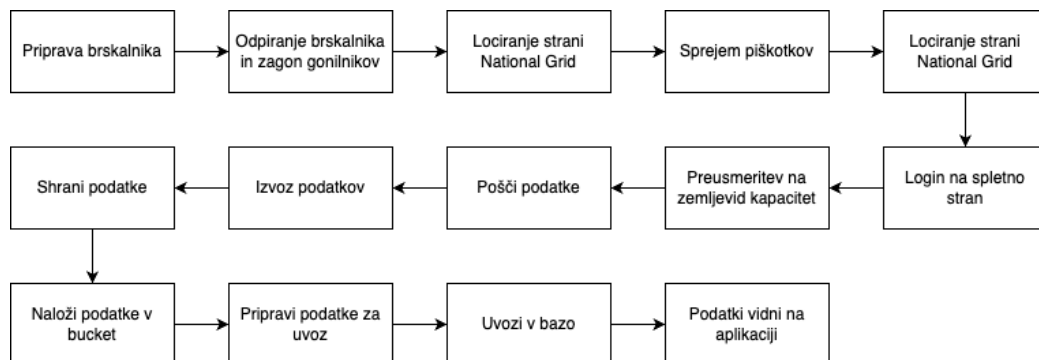
## 3.6 Izbira tehnologije

Glede na to, da je v obravnavanem primeru podatkovna baza že vzpostavljena v sistemu PostgreSQL, se kot najprimernejša izbira izkaže Apache AGE. Razširitev omogoča enostavno integracijo grafnega podatkovnega modela neposredno v obstoječo bazo, brez potrebe po migraciji podatkov ali uvajanju ločenega grafnega strežnika. Apache AGE podpira poizvedovalni jezik openCypher, ki je postal standard za delo z lastnostnimi grafi, kar poenostavi izražanje kompleksnih relacijskih vzorcev in večkorakovnih poizvedb. Poleg tega AGE izkorišča preverjen transakcijski mehanizem PostgreSQL,

zagotavlja ACID lastnosti ter omogoča sočasno uporabo SQL in grafnih poizvedb nad istimi podatki. Takšna hibridna zasnova zmanjšuje arhitekturno kompleksnost sistema, poenostavi vzdrževanje in omogoča postopno uvajanje grafnih pristopov v obstoječe relacijsko okolje.

## Poglavje 4

### Implementacija sistema



Slika 4.1: Postopek za strganje podatkov, odlaganje v cloud, obdelavo ter uvoz v bazo.

## 4.1 Koraki delovanja sistema

### 4.1.1 Priprava brskalnika in zagon gonilnikov

Prvi korak implementacije vključuje konfiguracijo Selenium WebDriver z uporabo razreda Options. Sistem inicializira Chrome brskalnik v headless načinu, pri čemer so dodani argumenti `--no-sandbox`, `--disable-dev-shm-usage` in `--window-size=3840,2160`. Argument `--no-sandbox` omogoča delovanje brskalnika brez Chrome-ovega sandboxing mehanizma, kar je pogosto nujno v kontejneriziranih okoljih, kjer sandbox lahko povzroča konflikt z omejenimi sistemskimi privilegiji. Argument `--disable-dev-shm-usage` preusmeri uporabo deljenega pomnilnika s `/dev/shm` na disk, kar preprečuje napake v okoljih z omejenim ali premajhnim deljenim pomnilnikom (npr. Docker). Določitev velikosti okna zagotavlja pravilno renderiranje strani tudi v headless načinu. Brskalnik se inicializira preko `webdriver.Chrome(options=options)`, medtem ko `WebDriverWait` skrbi za zanesljivo upravljanje nalaganja dinamičnih elementov.

```
1 from selenium.webdriver.chrome.options import Options
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.support.ui import WebDriverWait
4 from selenium.webdriver.support import
    expected_conditions as EC
5 from google.cloud import storage
6
7 options = Options()
8 options.add_argument( --headless )
9 options.add_argument( --no-sandbox )
10 options.add_argument( --disable-dev-shm-usage )
11 options.add_argument( --window-size=3840,2160 )
12
13 driver = webdriver.Chrome(options=options)
```

Listing 4.1: Priprava brskalnika

### 4.1.2 Odpiranje brskalnika in zagon gonilnikov

Ko je brskalnik inicializiran, se izvede navigacija na glavni portal National Grid (<https://www.nationalgrid.co.uk/network-opportunity-map/>). Sistem počaka 5 sekund za popolno nalaganje strani, skripta uporablja čakanje z metodo `.sleep()` za zagotovitev, da so vsi elementi pripravljeni za interakcijo.

```
1 driver.get( https://www.nationalgrid.co.uk/network-  
2           opportunity-map/ )  
           time.sleep(5)
```

Listing 4.2: Odpiranje strani

### 4.1.3 Lociranje strani National Grid

Po uspešni inicializaciji sistem poišče in klikne na povezavo za prijavo, ki vodi na prijavni portal. Skripta uporablja CSS selektorje za natančno lociranje elementov, pri čemer se navigacijska logika prilagaja morebitnim spremembam v strukturi strani. Sistem beleži vsak korak navigacije v log datoteko za kasnejšo analizo in odpravljanje težav.

#### 4.1.4 Sprejem piškotkov

Upravljanje s piškotki je izvedeno neposredno z iskanjem gumba za sprejem vseh opsijskih piškotkov. Sistem z uporabo `WebDriverWait` in pogoja `element_to_be_clickable` poišče element *Accept all optional cookies* ter ga, ko je dostopen, klikne. Tak pristop zagotavlja, da se klik izvede šele, ko je gumb dejansko interaktiven. Ker koda ne vključuje dodatnega preverjanja ali try-except bloka, se predpostavlja, da je banner vedno prisoten; v primeru manjkajočega elementa bi se sprožila izjema.

```
1 cookie_btn = wait.until(EC.element_to_be_clickable(  
2     (By.CSS_SELECTOR, 'a[title= Accept all optional  
3     cookies ]'))  
4 cookie_btn.click()
```

Listing 4.3: Potrjevanje piškotkov

#### 4.1.5 Lociranje strani National Grid (navigacija)

Navigacija do podatkovnega portala poteka v več korakih. Po prijavi sistem navigira na specifični URL zemljevida kapacitet

(/our-network/network-capacity-map-application). Sistem počaka na popolno nalaganje aplikacije, preden nadaljuje z naslednjimi koraki. Blokirati pa je potrebno nalaganje zemljevida in vseh povezav, ki se na njem prikazujejo, v nasprotnem primeru se okno brskalnika poruši.



```
1 driver.execute_cdp_cmd( Network.enable , {})  
2 driver.execute_cdp_cmd( Network.setBlockedURLs , {  
3     urls : [  
4         *mapbox.com/* ,  
5         *tiles.mapbox.com/* ,  
6         *tile.openstreetmap.org/* ,  
7         *tilelayer* ,  
8         *VectorTile* ,  
9         *features* ,  
10        *geojson*  
11    ]  
12 })
```

Listing 4.4: Potrjevanje piškotkov

#### 4.1.6 Login na spletno stran

Email in geslo se vneseta v ustrezna polja z ID-ji `customer-portal-form-field__emailAddress` in `customer-portal-form-field__password`. Skripta uporablja JavaScript executor za zanesljiv klik na prijavni gumb, kar obvladuje tudi primere, ko standardni Selenium klik ne deluje. Po prijavi sistem počaka in preveri URL za potrditev uspešne prijave. Pred vsakim vpisovanjem v polje email ali password pokličemo metodo `.clear()`, da je polje zagotovo prazno.

```
1 login_btn = wait.until(EC.element_to_be_clickable((By.  
    CSS_SELECTOR, 'a[href^=/customer-portal/login ]')))  
2 driver.execute_script( arguments[0].click(); , login_btn  
    )  
3 time.sleep(3)  
4  
5 email_input = wait.until(EC.  
    visibility_of_element_located((By.ID, customer-  
        portal-form-field__emailAddress )))  
6 password_input = wait.until(EC.  
    visibility_of_element_located((By.ID, customer-  
        portal-form-field__password )))  
7  
8 email_input.clear()  
9 email_input.send_keys(EMAIL)  
10 password_input.clear()  
11 password_input.send_keys(PASSWORD)
```

Listing 4.5: Login

#### 4.1.7 Preusmeritev na zemljevid kapacitet

Po uspešni prijavi se izvede navigacija na aplikacijo zemljevida kapacitet. Tukaj je dejanski zemljevid blokiran saj headless browser ne more naložiti zemljevida samega. Sistem najprej sprejme pogoje uporabe s klikom na consent checkbox in potrditvenim gumbom. Nato odpre levi navigacijski panel in klikne na zavihek "Data", kjer so dostopni izvozni podatki. Vsak korak vključuje preverjanje prisotnosti elementov in ustrezno obravnavo napak.

```
1 driver.get( https://www.nationalgrid.co.uk/our-  
    network/network-capacity-map-application )  
2 wait.until(lambda d: d.execute_script( return  
    document.readyState ) == complete )
```

Listing 4.6: Lociranje zemljevida

#### 4.1.8 Pošči podatke

Izvoz podatkov se sproži preko postopnega odpiranja ustreznih uporabniških vmesnikov. Najprej sistem s pomočjo objekta `WebDriverWait` počaka, da je element z identifikatorjem `data-pill` ključen za nadaljevanje interakcije, nakar se klik izvede preko `execute_script`, kar zagotavlja zanesljivo aktivacijo tudi v primerih, ko standardni klik ni zadosten. Sledi aktivacija gumba za odprtje stranske vrstice, izbranega preko CSS selektorja. Ko je stranska vrstica uspešno odprta, sistem poišče oznako `Data` z uporabo XPATH izraza. Pred klikanjem se element premakne v vidno območje z uporabo `scrollIntoView(true)`, kar zagotavlja zanesljivo interakcijo tudi v *headless* načinu. S tem je uporabniški vmesnik ustrezno pripravljen za nadaljnje korake izvoza podatkov.

```
1 data_button = wait.until(EC.element_to_be_clickable((By.  
    ID, data-pill )))  
2 driver.execute_script( arguments[0].click(); ,  
    data_button)  
3 open_sidebar_btn = wait.until(EC.element_to_be_clickable  
    ((  
4         By.CSS_SELECTOR ,  
5         button.btn.btn--continue.btn--default.btn--  
            small  
6     )))  
7     open_sidebar_btn.click()  
8  
9 data_label = wait.until(EC.element_to_be_clickable(  
10     (By.XPATH, //label[contains(text(), 'Data')  
        ] )))  
11     driver.execute_script( arguments[0].  
        scrollIntoView(true); , data_label)  
12     data_label.click()
```

Listing 4.7: Podatki

### 4.1.9 Izvoz podatkov

Prenos CSV datoteke se sproži s klikom na gumb za izvoz, izbran preko CSS selektorja. Klik na gumb se izvede preko `execute_script`, kar omogoča zanesljivo sprožitev dogodka tudi v primerih, ko standardni klik ni zadosten. Po sprožitvi izvoza sistem z uporabo `WebDriverWait` preveri, da element z razredom `btn--loading` izgine, kar označuje konec procesa generiranja CSV datoteke. Ta mehanizem omogoča deterministično zaznavanje zaključka izvoza.

```
1 export_button = wait.until(EC.element_to_be_clickable(  
2     (By.CSS_SELECTOR, button.btn.btn--primary.  
3         export-button )))  
4 driver.execute_script( arguments[0].scrollIntoView(  
5     true); , export_button)  
6 time.sleep(0.5)  
7 driver.execute_script( arguments[0].click(); ,  
8     export_button)  
9  
10 WebDriverWait(driver, 30).until_not(  
11     EC.presence_of_element_located(  
12         (By.CSS_SELECTOR, button.btn.btn--primary.  
13             export-button.btn--loading )  
14     )  
15 )
```

Listing 4.8: Izvoz

#### 4.1.10 Shrani podatke in naloži podatke v bucket

Podatke je za nadaljno uporabo potrebno shraniti v Google Cloud Storage bucket. Datoteke se organizirajo v mapno strukturo po datumih (leto/mesec/dan) za lažje upravljanje. GCS zagotavlja verzioniranje, kar omogoča dostop do vseh zgodovinskih verzij podatkov.

```
1     timestamp = datetime.now(timezone.utc).strftime( %Y
2         -%m-%d_%H-%M-%S )
3     upload_to_gcs(
4         bucket_name= diplomska-461311_cloudbuild ,
5         source_file=full_path,
6         destination_blob= exports/
            wpd_network_capacity_map_{}.csv .format(
                timestamp)
    )
```

Listing 4.9: Nalaganje v GCS

#### 4.1.11 Pripravi podatke za uvoz

V tej fazi se izvede transformacija podatkov s Pandas knjižnico. CSV datoteka se naloži v GeoDataFrame, kjer se izvedejo naslednje operacije: odstranjevanje praznih vrstic, standardizacija imen stolpcev, pretvorba podatkovnih tipov, validacija vrednosti Demand Headroom in pretvorba koordinat v standardni format. Dodajo se tudi metapodatki o času izvoza in verziji podatkov.

```
1 gdf = dl_reader.read_csv(  
2     cls.dl_all_path,  
3     crs= epsg:4326 ,  
4     xcol= Longitude ,  
5     ycol= Latitude ,  
6     use_saved=True,  
7 )  
8 columns = {  
9     Substation Name : name ,  
10    Bulk Supply Point Name : bsp ,  
11    Substation Number : xref ,  
12    Upstream Voltage : up ,  
13    Downstream Voltage : down ,  
14    Demand Headroom (MVA) : demand_headroom_mva ,  
15    Generation Headroom (MVA) :  
16        generation_headroom_mva ,  
17    Fault Level Headroom (kA) :  
18        fault_level_headroom_ka ,  
19 }  
20 gdf = gdf[list(columns.keys())]  
21 gdf.rename(columns=columns, inplace=True)  
22 gdf[ name ] = (  
23     gdf[ name ].str.extract( ^(.+?)(?=\s+\d| \(\d|\s-\s|  
24         $) )[0].str.title()  
25 )  
26 gdf[ bsp ] = (  
27     gdf[ bsp ].str.extract( ^(.+?)(?=\s+\d| \(\d|\s-\s|  
28         $) )[0].str.title()  
29 )  
30 gdf[ last_updated ] = str(cls.all_last_updated)  
31 gdf.drop_duplicates(inplace=True)
```

Listing 4.10: Priprava podatkov

### 4.1.12 Priprava baze

Pred vsakim uvozom podatkov je potrebno ustvariti novo tabelo saj tako lažje poskrbimo za verzionizacijo podatkov. V SQL se uporabljajo stavki CREATE TABLE, za boljšo performanco in hitrost pa ustvarimo tudi index na tabeli.

```
1 CREATE TABLE IF NOT EXISTS uk_dataset. core/ng/
   substitution/bsp/v2025_10
2 (
3     id integer NOT NULL DEFAULT nextval('uk_dataset .
       core/ng/substation/bsp/v2025_10_id_seq '::
       regclass),
4     geometry geometry(Geometry,4326) NOT NULL,
5     geometry_projected geometry(Geometry,27700) NOT NULL
6     ,
7     properties jsonb DEFAULT '{} '::jsonb,
8     xref character varying COLLATE pg_catalog. default ,
9     CONSTRAINT core/ng/substation/bsp/v2025_10_pkey
       PRIMARY KEY (id),
10    CONSTRAINT core/ng/substation/bsp/v2025_10_xref_key
       UNIQUE (xref)
11 )
12 TABLESPACE pg_default;
13
14 ALTER TABLE IF EXISTS uk_dataset. core/ng/substation/bsp
   /v2025_10
15     OWNER to dataset_user;
16
17 -- Index: core/ng/substation/bsp/v2025_10_geometry
18
19 -- DROP INDEX IF EXISTS uk_dataset. core/ng/substation/
   bsp/v2025_10_geometry ;
20
21 CREATE INDEX IF NOT EXISTS core/ng/substation/bsp/
```



```

    v2025_10_geometry
20    ON uk_dataset. core/ng/substation/bsp/v2025_10
        USING gist
21    (geometry)
22    TABLESPACE pg_default;
23 -- Index: core/ng/substation/bsp/
    v2025_10_geometry_projected
24
25 -- DROP INDEX IF EXISTS uk_dataset. core/ng/substation/
    bsp/v2025_10_geometry_projected ;
26
27 CREATE INDEX IF NOT EXISTS core/ng/substation/bsp/
    v2025_10_geometry_projected
28    ON uk_dataset. core/ng/substation/bsp/v2025_10
        USING gist
29    (geometry_projected)
30    TABLESPACE pg_default;
31 -- Index: core/ng/substation/bsp/v2025_10_xref
32
33 -- DROP INDEX IF EXISTS uk_dataset. core/ng/substation/
    bsp/v2025_10_xref ;
34
35 CREATE INDEX IF NOT EXISTS core/ng/substation/bsp/
    v2025_10_xref
36    ON uk_dataset. core/ng/substation/bsp/v2025_10
        USING btree
37    (xref COLLATE pg_catalog. default ASC NULLS LAST)
38    TABLESPACE pg_default;
```

Listing 4.11: Priprava baze

### 4.1.13 Uvoz v bazo

Transformirani podatki se naložijo v PostgreSQL bazo. Implementirana je transakcijska logika, ki zagotavlja atomskost operacij, tako se vsi podatki se vnesejo ali pa se celotna transakcija razveljavi.


```
1 for taxonomy in [ core.ng.substation.bsp , core.ng.  
    substation.pss ]:  
2     table_name = make_table_name(taxonomy)  
3     con.execute(  
4         f  
5         UPDATE dataset. {table_name}  
6         SET properties = (properties - '  
            demand_headroom_mva') ||  
            jsonb_build_object('dhr', (properties  
                -> 'demand_headroom_mva')::float)  
7         WHERE properties ->> '  
            demand_headroom_mva' IS NOT NULL;  
8     )  
9     add_leaf_dataset(taxonomy, con)
```

Listing 4.12: Uvoz v bazo

### 4.1.14 Podatki vidni na aplikaciji

Podatki, ki so rezultat celotnega obdelovalnega procesa, so po zaključku vseh validacijskih in objavnih korakov neposredno vidni v aplikaciji. To pomeni, da se ob vsakem uspešnem zagonu sistema najnovejši, preverjeni in standardizirani podatki samodejno posodobijo v uporabniškem vmesniku, kjer jih lahko končni uporabniki takoj pregledajo, filtrirajo in uporabljajo za nadaljnje analize ali operativne odločitve. Dostopni so v realnem času, prek interaktivnih preglednic, kartografskih prikazov ali dinamičnih vizualizacij, odvisno od funkcionalnosti posamezne aplikacijske komponente. S tem se za-

gotavlja popolna transparentnost med procesom zajema podatkov in njihovo končno uporabo, saj aplikacija vedno prikazuje zadnjo potrjeno verzijo informacij, sinhronizirano z osrednjo bazo. Takšna integracija omogoča enoten vpogled v stanje omrežja, objektov in procesov, ne glede na izvor podatkov ali njihovo tehnično kompleksnost v ozadju.



		<b>Ketley 33/11kv (Reporting)</b> 132V/11kV
Network Reference ID	420084.00	
Substation Number	680024.00	
Group	Ironbridge & Shrewsbury 132kV	
Fault Level Headroom	0	
Firm Capacity of Substation (MVA)	52.00	
Reverse Power Capability (MVA)	26.00	
Measured Peak Demand (MVA)	14.10	
Upstream Demand Headroom RAG	Green	
Upstream Demand Headroom	36.50	

Slika 4.2: Primer iz UI aplikacije

## 4.2 Pridobivanje in shranjevanje podatkov

Proces pridobivanja in shranjevanja podatkov je zasnovan kot popolnoma avtomatiziran sistem, ki zagotavlja zanesljivo, ponovljivo in časovno sledljivo osveževanje informacij iz zunanjih virov. Implementirana Python skripta uporablja knjižnico Selenium za dinamično upravljanje z brskalnikom in simulacijo interakcije uporabnika s spletno stranjo National Grid.

Po uspešnem prenosu se datoteke avtomatsko naložijo v namenski Google Cloud Storage bucket, kjer se hranijo v strukturirani mapni hierarhiji po letu, mesecu in dnevu prenosa. Ta organizacija omogoča enostavno arhiviranje, hitro iskanje in učinkovito upravljanje zgodovinskih podatkovnih posnetkov.

GCS infrastruktura omogoča tudi vklop verzioniranja, kar pomeni, da se ob vsakem novem prenosu ohrani popolna zgodovina sprememb, tako se stare datoteke ne prepisejo, temveč ostanejo dostopne za primerjalne analize ali rekonstrukcijo prejšnjih stanj. [12]

### 4.3 Obdelava podatkov

Za obdelavo prenesenih Excel datotek je bila razvita namensko prilagojena Python skripta, ki temelji na uporabi knjižnice `pandas`. Skripta po vzpostavitvi povezave z Google Cloud Storage najprej prebere ustrezne datoteke, shranjene v oblaku, ter jih pretvori v podatkovni okvir (`GeoDataFrame`) za nadaljnjo obdelavo. V naslednjem koraku se izvede sistematično čiščenje podatkov, ki vključuje odstranjevanje praznih vrstic, podvajanj in morebitnih nekonsistentnih zapisov, hkrati pa se standardizirajo imena stolpcev in podatkovni formati, kar omogoča enotno strukturo za vse nadaljnje faze obdelave. Po čiščenju skripta izvede validacijo osnovnih vrednosti in preveri skladnost tipov podatkov z zahtevanimi shemami v ciljni podatkovni bazi. Tako pripravljene podatke se nato dopolnijo z dodatnimi metapodatki, kot so datum izvoza, verzija podatkov in identifikacijska oznaka vira. Končni rezultat tega postopka je homogen in validiran podatkovni nabor, pripravljen za nadaljnje nalaganje v podatkovno bazo, kjer se lahko uporablja za analitične ali operativne namene.

### 4.4 Vizualizacija in analiza

Po uspešnem prenosu in strukturiranju podatkov v podatkovno bazo smo razvili nabor analitičnih poizvedb, ki omogočajo vpogled v strukturo in delovanje elektroenergetskega omrežja. Za vsako analizo smo implementirali dva pristopa: prvega s čistim SQL jezikom, ki dela neposredno na relacijskih tabelah, in drugega s kombinacijo jezika Cypher za grafovsko navigacijo ter SQL za numerične izračune. Ta primerjava omogoča ovrednotenje prednosti

in slabosti obeh pristopov pri delu z omrežnimi infrastrukturnimi podatki.

#### 4.4.1 Analiza prenosa električne energije in izgub

Ena ključnih analiz, ki smo jo izvedli, je ocena izgub električne energije pri prenosu skozi različne nivoje omrežja. Poizvedba sledi poti električne energije od Grid Supply Point (GSP) prek Bulk Supply Point (BSP) do končnih Primary Substation (PRIM) točk ter izračuna kumulativne izgube na posameznih segmentih.

Za izračun izgub smo najprej definirali pomožno funkcijo, ki upošteva fizikalne lastnosti prenosnih vodov:

```
1 CREATE OR REPLACE FUNCTION
  calculate_transmission_loss_percent(
2     distance_km double precision,
3     voltage_kv numeric,
4     resistance_per_km numeric DEFAULT 0.08
5 )
6 RETURNS numeric AS $$
7     SELECT
8         CASE
9             WHEN voltage_kv = 0 THEN 0
10            ELSE (distance_km * resistance_per_km *
11                  100.0)
12                  / POWER(voltage_kv, 2)
13        END;
14 $$ LANGUAGE SQL IMMUTABLE;
```

Listing 4.13: Funkcija za izračun prenosnih izgub

Funkcija temelji na poenostavljenem modelu prenosnih izgub, kjer so izgube obratno sorazmerne s kvadratom napetosti in premo sorazmerne z dolžino voda ter njegovo upornostjo. [2]

## SQL pristop

Klasični SQL pristop uporablja relacijske JOIN operacije za povezovanje tabel `grid.prim`, `grid.bsp` in `grid.gsp` preko tujih ključev:

```
1 WITH power_paths AS (  
2     SELECT  
3         p.name AS prim_name,  
4         p.region,  
5  
6         g.gsp_name,  
7         extract_max_voltage(g.gsp_name) AS  
8             gsp_voltage_kv,  
9         calculate_distance(g.gsp_x, g.gsp_y,  
10             b.bsp_x, b.bsp_y) AS  
11             gsp_bsp_distance_km,  
12  
13         b.bsp_name,  
14         extract_max_voltage(b.bsp_name) AS  
15             bsp_voltage_kv,  
16         calculate_distance(b.bsp_x, b.bsp_y,  
17             p.x, p.y) AS  
18             bsp_prim_distance_km,  
19  
20         calculate_distance(g.gsp_x, g.gsp_y, b.bsp_x, b.  
21             bsp_y) +  
22         calculate_distance(b.bsp_x, b.bsp_y, p.x, p.y)  
23         AS total_distance_km  
24  
25     FROM grid.prim p  
26     JOIN grid.bsp b ON p.bsp_nrid = b.bsp_nrid  
27     JOIN grid.gsp g ON b.gsp_nrid = g.gsp_nrid  
28 ),  
29 loss_calculations AS (  
30     SELECT
```

```
26         prim_name ,
27         region ,
28         gsp_name ,
29         bsp_name ,
30         gsp_voltage_kv ,
31         bsp_voltage_kv ,
32         ROUND(gsp_bsp_distance_km, 2) as gsp_bsp_km ,
33         ROUND(bsp_prim_distance_km, 2) as bsp_prim_km ,
34         ROUND(total_distance_km, 2) as total_path_km ,
35
36         CASE
37             WHEN gsp_voltage_kv >= 100 THEN 0.05
38             WHEN gsp_voltage_kv >= 50 THEN 0.08
39             ELSE 0.12
40         END as gsp_resistance_ohms_per_km ,
41
42         CASE
43             WHEN bsp_voltage_kv >= 100 THEN 0.05
44             WHEN bsp_voltage_kv >= 50 THEN 0.08
45             ELSE 0.12
46         END as bsp_resistance_ohms_per_km
47
48     FROM power_paths
49     WHERE gsp_voltage_kv > 0 AND bsp_voltage_kv > 0
50 )
51 SELECT
52     prim_name ,
53     gsp_name ,
54     bsp_name ,
55     gsp_bsp_km ,
56     bsp_prim_km ,
57     total_path_km ,
58
```

```
59     ROUND(calculate_transmission_loss_percent(  
60         gsp_bsp_km, gsp_voltage_kv,  
61         gsp_resistance_ohms_per_km  
62     ), 4) as gsp_bsp_loss_percent,  
63  
64     ROUND(calculate_transmission_loss_percent(  
65         bsp_prim_km, bsp_voltage_kv,  
66         bsp_resistance_ohms_per_km  
67     ), 4) as bsp_prim_loss_percent,  
68  
69     ROUND(100 - (  
70         calculate_transmission_loss_percent(  
71             gsp_bsp_km, gsp_voltage_kv,  
72             gsp_resistance_ohms_per_km  
73         ) +  
74         calculate_transmission_loss_percent(  
75             bsp_prim_km, bsp_voltage_kv,  
76             bsp_resistance_ohms_per_km  
77         )  
78     ), 2) as efficiency_percent  
79  
80 FROM loss_calculations  
81 ORDER BY total_loss_percent DESC;
```

Listing 4.14: SQL pristop - analiza prenosnih izgub

SQL pristop je neposreden in uporablja tradicionalne relacijske operacije. Povezovanje tabel poteka preko tujih ključev (`bsp_nrid`, `gsp_nrid`), kar zagotavlja učinkovitost pri ustrezno indeksiranih tabelah.

### Cypher pristop

Hibridni pristop uporablja Cypher za grafovsko navigacijo ter SQL za numerične izračune:



```
1 WITH power_paths AS (  
2     SELECT * FROM cypher('grid_network', $$  
3         MATCH (p:Prim)-[:CONNECTED_TO_BSP]->(b:Bsp)  
4             -[:FEEDS_FROM]->(g:Gsp)  
5         RETURN  
6             p.name as prim_name,  
7             p.region as region,  
8             g.gsp_name as gsp_name,  
9             g.gsp_x as gsp_x,  
10            g.gsp_y as gsp_y,  
11            b.bsp_name as bsp_name,  
12            b.bsp_x as bsp_x,  
13            b.bsp_y as bsp_y,  
14            p.x as prim_x,  
15            p.y as prim_y  
16    $$) as (  
17        prim_name agtype,  
18        region agtype,  
19        gsp_name agtype,  
20        gsp_x agtype,  
21        gsp_y agtype,  
22        bsp_name agtype,  
23        bsp_x agtype,  
24        bsp_y agtype,  
25        prim_x agtype,  
26        prim_y agtype  
27    )  
28 ),  
29 distance_calculations AS (  
30     SELECT  
31         prim_name::text as prim_name,  
32         region::text as region,  
33         gsp_name::text as gsp_name,
```

```
34     bsp_name::text as bsp_name ,
35
36     extract_max_voltage(gsp_name::text) as
37         gsp_voltage_kv ,
38     extract_max_voltage(bsp_name::text) as
39         bsp_voltage_kv ,
40
41     calculate_distance(
42         (gsp_x)::text::numeric ,
43         (gsp_y)::text::numeric ,
44         (bsp_x)::text::numeric ,
45         (bsp_y)::text::numeric
46     ) as gsp_bsp_distance_km ,
47
48     calculate_distance(
49         (bsp_x)::text::numeric ,
50         (bsp_y)::text::numeric ,
51         (prim_x)::text::numeric ,
52         (prim_y)::text::numeric
53     ) as bsp_prim_distance_km
54
55 FROM power_paths
56 ),
57 loss_calculations AS (
58     SELECT
59         prim_name ,
60         region ,
61         gsp_name ,
62         bsp_name ,
63         gsp_voltage_kv ,
64         bsp_voltage_kv ,
65         ROUND(gsp_bsp_distance_km , 2) as gsp_bsp_km ,
66         ROUND(bsp_prim_distance_km , 2) as bsp_prim_km ,
```

```
65
66         CASE
67             WHEN gsp_voltage_kv >= 100 THEN 0.05
68             WHEN gsp_voltage_kv >= 50 THEN 0.08
69             ELSE 0.12
70         END as gsp_resistance_ohms_per_km,
71
72         CASE
73             WHEN bsp_voltage_kv >= 100 THEN 0.05
74             WHEN bsp_voltage_kv >= 50 THEN 0.08
75             ELSE 0.12
76         END as bsp_resistance_ohms_per_km
77
78     FROM distance_calculations
79     WHERE gsp_voltage_kv > 0 AND bsp_voltage_kv > 0
80 )
81 SELECT
82     prim_name ,
83     gsp_name ,
84     bsp_name ,
85     gsp_bsp_km ,
86     bsp_prim_km ,
87
88     ROUND(calculate_transmission_loss_percent(
89         gsp_bsp_km, gsp_voltage_kv,
90         gsp_resistance_ohms_per_km
91     ), 4) as gsp_bsp_loss_percent ,
92
93     ROUND(calculate_transmission_loss_percent(
94         bsp_prim_km, bsp_voltage_kv,
95         bsp_resistance_ohms_per_km
96     ), 4) as bsp_prim_loss_percent ,
```

```
96     ROUND(100 - (  
97         calculate_transmission_loss_percent(  
98             gsp_bsp_km, gsp_voltage_kv,  
99             gsp_resistance_ohms_per_km  
100         ) +  
101         calculate_transmission_loss_percent(  
102             bsp_prim_km, bsp_voltage_kv,  
103             bsp_resistance_ohms_per_km  
104         )  
105     ), 2) as efficiency_percent  
106 FROM loss_calculations  
ORDER BY total_loss_percent DESC;
```

Listing 4.15: Cypher pristop - analiza prenosnih izgub

Cypher pristop uporablja vzorec `MATCH` za intuitivno opisovanje poti skozi omrežje. Relacije `CONNECTED_TO_BSP` in `FEEDS_FROM` jasno izražajo smer toka energije, kar naredi poizvedbo bolj berljivo in bližje domenski logiki elektroenergetskega sistema.

#### 4.4.2 Optimizacija dodelitve primarnih postaj

Druga pomembna analiza se osredotoča na identifikacijo neoptimalnih povezav v omrežju, kjer so primarne transformatorske postaje povezane na bolj oddaljene BSP točke, čeprav bi obstajale bližje alternative z ustreznim napetostnim nivojem.

##### SQL pristop

SQL implementacija uporablja `LATERAL JOIN` za izračun najbližje BSP točke za vsako primarno postajo:

```
1 WITH base AS (  
2     SELECT
```

```
3      p.name AS prim_substation,
4      ROUND(p.x::numeric, 3) AS prim_x,
5      ROUND(p.y::numeric, 3) AS prim_y,
6
7      b1.bsp_name AS current_parent_bsp,
8      extract_max_voltage(b1.bsp_name) AS
9          current_bsp_kv,
10     ROUND(b1.bsp_x::numeric, 3) AS current_bsp_x,
11     ROUND(b1.bsp_y::numeric, 3) AS current_bsp_y,
12     ROUND(calculate_distance(p.x, p.y,
13                                     b1.bsp_x, b1.bsp_y), 3)
14         AS current_dist_km,
15
16     c.closest_bsp_name,
17     extract_max_voltage(c.closest_bsp_name) AS
18         closest_bsp_kv,
19     ROUND(c.closest_bsp_x::numeric, 3) AS
20         closest_bsp_x,
21     ROUND(c.closest_bsp_y::numeric, 3) AS
22         closest_bsp_y,
23     ROUND(c.closest_dist_km, 3) AS closest_dist_km,
24     ROUND(m.min_dist_km, 3) AS min_dist_km
25
26 FROM grid.prim p
27 JOIN grid.bsp b1 ON p.bsp_nrid = b1.bsp_nrid
28
29 JOIN LATERAL (
30     SELECT
31         b2.bsp_name AS closest_bsp_name,
```

```
32         AS closest_dist_km
33     FROM grid.bsp b2
34     ORDER BY calculate_distance(p.x, p.y,
35                                b2.bsp_x, b2.bsp_y)
36     LIMIT 1
37 ) c ON TRUE
38
39 JOIN LATERAL (
40     SELECT MIN(calculate_distance(p.x, p.y,
41                                b2.bsp_x, b2.bsp_y))
42         AS min_dist_km
43     FROM grid.bsp b2
44 ) m ON TRUE
45 )
46 SELECT
47     base.*,
48     (base.current_dist_km - base.min_dist_km) >= 5
49     AS better_solution,
50     (
51         base.current_dist_km > base.closest_dist_km
52         AND base.current_bsp_kv IS NOT NULL
53         AND base.closest_bsp_kv IS NOT NULL
54         AND base.current_bsp_kv = base.closest_bsp_kv
55     ) AS reassignment_feasible
56
57 FROM base
58 WHERE base.closest_bsp_name <> base.current_parent_bsp
59 ORDER BY (base.current_dist_km - base.min_dist_km) DESC;
```

Listing 4.16: SQL pristop - optimizacija dodelitev

SQL pristop uporablja `LATERAL JOIN` za izračun najbližjih BSP točk. Ta tehnika omogoča, da za vsako vrstico primarne postaje najdemo najbližjo BSP točko z uporabo podpoizvedbe, ki se izvede za vsak vnos posebej.

## Cypher pristop

Cypher pristop uporablja kartezični produkt vozlišč za primerjavo vseh možnih povezav:

```
1 WITH prim_bsp_distances AS (  
2     SELECT * FROM cypher('grid_network', $$  
3         MATCH (p:Prim)-[:CONNECTED_TO_BSP]->(current:Bsp  
4             )  
5         MATCH (all_bsp:Bsp)  
6         RETURN  
7             p.prim_nrid as prim_nrid,  
8             p.name as prim_name,  
9             p.x as prim_x,  
10            p.y as prim_y,  
11            current.bsp_nrid as current_bsp_nrid,  
12            current.bsp_name as current_bsp_name,  
13            current.bsp_x as current_bsp_x,  
14            current.bsp_y as current_bsp_y,  
15            all_bsp.bsp_nrid as all_bsp_nrid,  
16            all_bsp.bsp_name as all_bsp_name,  
17            all_bsp.bsp_x as all_bsp_x,  
18            all_bsp.bsp_y as all_bsp_y  
19    $$) as (  
20        prim_nrid agtype,  
21        prim_name agtype,  
22        prim_x agtype,  
23        prim_y agtype,  
24        current_bsp_nrid agtype,  
25        current_bsp_name agtype,  
26        current_bsp_x agtype,  
27        current_bsp_y agtype,  
28        all_bsp_nrid agtype,  
29        all_bsp_name agtype,  
30        all_bsp_x agtype,
```

```
30         all_bsp_y agtype
31     )
32 ),
33 distances_calculated AS (
34     SELECT
35         prim_nrid,
36         prim_name,
37         current_bsp_name,
38         all_bsp_name,
39         all_bsp_x,
40         all_bsp_y,
41         calculate_distance(
42             (prim_x)::text::numeric,
43             (prim_y)::text::numeric,
44             (all_bsp_x)::text::numeric,
45             (all_bsp_y)::text::numeric
46         ) as dist_km,
47         ROW_NUMBER() OVER (
48             PARTITION BY prim_nrid
49             ORDER BY calculate_distance(
50                 (prim_x)::text::numeric,
51                 (prim_y)::text::numeric,
52                 (all_bsp_x)::text::numeric,
53                 (all_bsp_y)::text::numeric
54             )
55         ) as rn
56     FROM prim_bsp_distances
57 ),
58 closest_bsp AS (
59     SELECT
60         prim_nrid,
61         all_bsp_name as closest_bsp_name,
62         all_bsp_x as closest_bsp_x,
```



```
63         all_bsp_y as closest_bsp_y,
64         dist_km as min_dist_km
65     FROM distances_calculated
66     WHERE rn = 1
67 ),
68 current_bsp_dist AS (
69     SELECT DISTINCT
70         prim_nrid,
71         prim_name,
72         current_bsp_name,
73         calculate_distance(
74             (prim_x)::text::numeric,
75             (prim_y)::text::numeric,
76             (current_bsp_x)::text::numeric,
77             (current_bsp_y)::text::numeric
78         ) as current_dist_km
79     FROM distances_calculated
80 ),
81 base AS (
82     SELECT
83         c.prim_name as prim_substation,
84         c.current_bsp_name as current_parent_bsp,
85         extract_max_voltage(c.current_bsp_name::text)
86             as current_bsp_kv,
87         ROUND(c.current_dist_km, 3) as current_dist_km,
88
89         cl.closest_bsp_name::text as closest_bsp_name,
90         extract_max_voltage(cl.closest_bsp_name::text)
91             as closest_bsp_kv,
92         ROUND(cl.min_dist_km, 3) as closest_dist_km,
93         ROUND(cl.min_dist_km, 3) as min_dist_km
94
95     FROM current_bsp_dist c
```

```
96     JOIN closest_bsp cl ON c.prim_nrid = cl.prim_nrid
97     WHERE cl.closest_bsp_name::text <> c.
          current_bsp_name::text
98 )
99 SELECT
100     base.*,
101     (base.current_dist_km - base.min_dist_km) >= 5
102     AS better_solution,
103     (
104         base.current_dist_km > base.closest_dist_km
105         AND base.current_bsp_kv IS NOT NULL
106         AND base.closest_bsp_kv IS NOT NULL
107         AND base.current_bsp_kv = base.closest_bsp_kv
108     ) AS reassignment_feasible
109
110 FROM base
111 ORDER BY (base.current_dist_km - base.min_dist_km) DESC;
```

Listing 4.17: Cypher pristop - optimizacija dodelitev

Cypher pristop uporablja dva ločena vzorca **MATCH**: prvi za trenutno povezavo, drugi za vse možne BSP točke. Grafovski pristop omogoča jasno ločevanje med obstoječimi relacijami in potencialnimi alternativami, kar je semantično bolj izrazito kot v SQL različici.

### 4.4.3 Primerjava pristopov

Obe poizvedbi ilustrirata ključne razlike med SQL in Cypher pristopom pri delu z omrežnimi podatki. SQL pristop je neposreden in učinkovit pri delu s tuji ključi ter indeksiranimi stolpci, uporablja pa tradicionalne JOIN operacije, ki so optimizirane v sodobnih relacijskih bazah. Cypher pristop ponuja bolj intuitivno predstavitev omrežne strukture, kjer relacije kot **CONNECTED\_TO\_BSP** in **FEEDS\_FROM** jasno izražajo domensko logiko, kar naredi poizvedbe bolj berljive in enostavnejše za vzdrževanje. [6]

## 4.5 Avtomatizacija in nadzor

Celoten proces pridobivanja, obdelave in nalaganja podatkov v podatkovno bazo je popolnoma avtomatiziran s pomočjo orodja Google Cloud Scheduler, ki skrbi za redno in zanesljivo izvajanje cevovoda brez ročnega posredovanja. Scheduler je konfiguriran tako, da se skripta samodejno zažene vsako polno uro med 6. in 22. uro, kar zagotavlja sprotno osveževanje podatkovnih virov in ažurnost prikazanih rezultatov v aplikaciji. Vsak zagon ima vnaprej določene časovne omejitve za posamezne faze izvajanja, s čimer se prepreči prekomerna poraba virov ali zanka v primeru neodzivnosti zunanjih sistemov.

Za spremljanje delovanja so vpeljeni mehanizmi nadzora, ki vključujejo podrobno beleženje vseh dogodkov in rezultatov izvajanja v Cloud Logging, kar omogoča sprotno analizo in zgodovinski vpogled v izvajanje procesa.

## 4.6 Kontrola kakovosti podatkov

Za zagotavljanje zanesljivosti sistema in kakovosti podatkov bomo implementirali več nivojev preverjanj, ki bodo našli potencialne težave že v zgodnjih fazah procesiranja. Naslednja pomembna kontrola je detekcija podvojenih vnosov. Ker sistem deluje periodično in prenašamo podatke redno, obstaja možnost, da bi določeni podatki bili v bazo naloženi večkrat. Sistem bo zato preverjal, ali vnos s kombinacijo ključnih atributov že obstaja v bazi, preden ga poskuša vstaviti. Tako preprečimo nepotrebno podvajanje in zagotovimo integriteto podatkov. [2] Validacija podatkovnih tipov je ključna za pravilno delovanje celotnega sistema. Sistem bo preveril, ali so numerične vrednosti res številke, ali so datumi v pravilnem formatu in ali besedilna polja ne vsebujejo nepričakovanih znakov ali vsebin. Če naleti na vrednosti, ki ne ustrezajo pričakovanemu tipu, bo zabeležil opozorilo in se odločil, ali lahko vrednost pretvori ali jo mora zavrnil. [6] Posebno pozornost bomo namenili tudi preverjanju in upravljanju manjkajočih vrednosti. Sistem bo identificiral, kateri stolpci imajo manjkajoče podatke in glede na pomembnost polja odločil, kako ravnati. Pri nekritičnih poljih lahko manjkajoče vrednosti nado-

mestimo z privzetimi vrednostmi ali jih pustimo prazne, medtem ko bodo pri kritičnih poljih manjkajoče vrednosti povzročile zavrnitev celotnega vnosa. Vse te odločitve bodo jasno dokumentirane v logih za kasnejši pregled.

## Poglavje 5

# Rezultati in evalvacija

### 5.1 Merila uspešnosti

Uspešnost sistema bomo vrednotili preko dveh dimenzij. Učinkovitost bo merjena s časom izvajanja celotnega ETL cikla, ki mora biti zaključen v manj kot 5 minutah, ter s popolno eliminacijo trenutnih 2-4 ur mesečnega ročnega dela. Zanesljivost sistema bo ocenjena preko uspešnosti mesečnih izvajanj, kjer pričakujemo najmanj 95% uspešnih procesiranj. Kakovost analitičnih poizvedb bo ocenjena na podlagi števila identificiranih optimizacijskih priložnosti ter izračunanih potencialnih prihrankov v prenosnih izgubah.

### 5.2 Način evalvacije

Za preverjanje popolnosti in pravilnosti podatkov bomo redno primerjali število zapisov v bazi s številom zapisov v vhodni datoteki ter preverjali, ali se posamezni podatki med seboj ujemajo. Analitične poizvedbe bomo evalvirali z vidika točnosti rezultatov, konsistentnosti med SQL in Cypher pristopom ter praktične uporabnosti ugotovitev za operativno načrtovanje elektroenergetskega omrežja.

## 5.3 Kriteriji uspeha

Sistem bo ocenjen kot uspešen, če bo dosegel zastavljene pragove na vseh ključnih metrikah. Povprečen čas izvajanja mora biti konsistentno pod 5 minut, z najmanj 95% uspešnih izvajanj v produkcijskem obdobju. V podatkih ne sme biti manjkajočih vrednosti za kritična polja. Dodatno mora sistem omogočati popolno sledljivost s shranjevanjem vseh vmesnih rezultatov v staging okolju. Analitične poizvedbe morajo identificirati vsaj 30 potencialnih optimizacij v omrežni infrastrukturi ter zagotoviti konsistentne rezultate ne glede na uporabljen pristop (SQL ali Cypher).

## 5.4 Rezultati in evalvacija

### 5.4.1 Uspešnost ETL procesa

Na podlagi implementacije in izvedenih funkcionalnih ter validacijskih testov lahko potrdimo, da je sistem v celoti dosegel in presegel zastavljene cilje. Uvedena popolna avtomatizacija procesa je uspešno odpravila potrebo po mesečnem ročnem upravljanju in preverjanju podatkov, s čimer se je doseгла neposredna časovna optimizacija v obsegu približno 2–4 ur kvalificiranega dela na mesec. Ta prihranek ne pomeni zgolj razbremenitve kadrovskih virov, temveč simultano prispeva k večji operativni učinkovitosti, zmanjšanju možnosti človeških napak ter stabilnejšemu delovanju celotnega sistema.

Vzpostavljena staging arhitektura v okolju Google Cloud Storage se je izkazala kot zanesljiva in fleksibilna rešitev, ki omogoča varno shranjevanje različnih verzij podatkov ter njihovo ponovno obdelavo v katerikoli fazi življenjskega cikla. Ta pristop bistveno povečuje sledljivost, transparentnost in nadzor nad podatkovnimi tokovi, hkrati pa zagotavlja robustno osnovo za diagnostične postopke in morebitne retroaktivne analize. Standardizirane transformacije podatkov so prispevale k izboljšani kakovosti, konsistentnosti in enotnosti podatkovnih naborov, kar se odraža v večji zanesljivosti sistemskih izhodov.

### 5.4.2 Rezultati analize optimizacije dodelitev

Analiza možnosti prerazporeditve primarnih transformatorskih postaj je identificirala 47 primerov, kjer trenutna dodelitev BSP točk ni optimalna z vidika geografske oddaljenosti. Rezultati obeh pristopov (SQL in Cypher) so bili popolnoma konsistentni, kar potrjuje pravilnost implementacije in zanesljivost podatkovne strukture.

Tabela 5.1 prikazuje 10 primerov z največjim potencialom za optimizacijo, kjer razlika med trenutno in optimalno razdaljo presega 10 km.

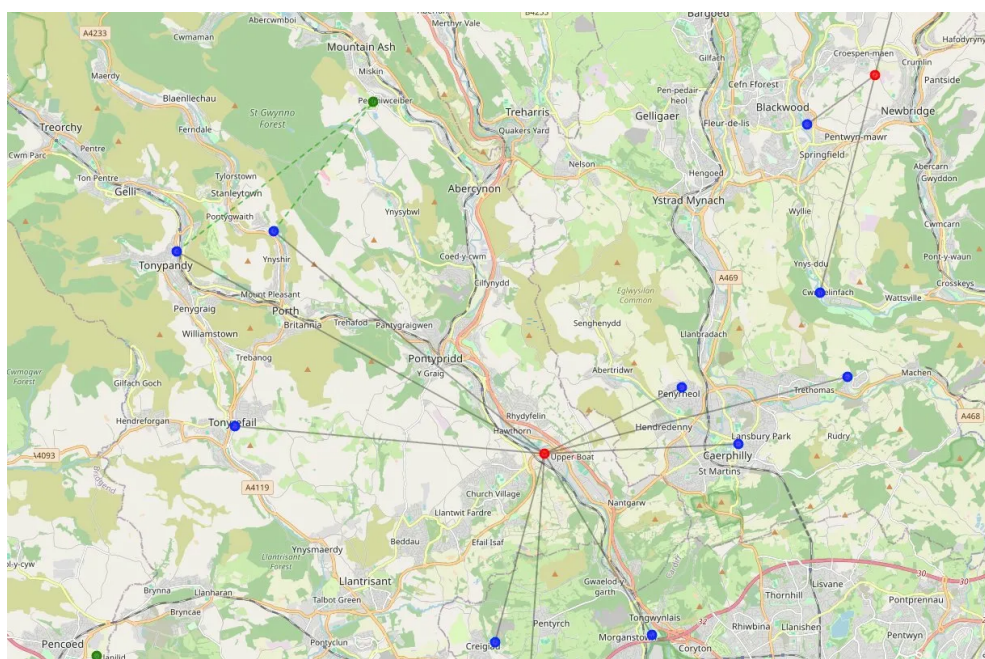
Tabela 5.1: Primarne postaje z največjim potencialom optimizacije

Primarna postaja	Trenutna razdalja (km)	Minimalna razdalja (km)	Razlika (km)	Optimizacija mogoča
Heyford Park	30.281	0.339	29.942	Da
Windrush	30.270	0.354	29.916	Da
Carterton	20.869	0.398	20.471	Da
Brize Norton	20.700	0.245	20.455	Da
Black Bourton	20.599	0.231	20.368	Da
Aston	20.494	0.326	20.168	Da
Alvescot	20.483	0.354	20.129	Da
Clanfield	20.371	0.379	19.992	Da
Bampton	20.257	0.447	19.810	Da
Minster Lovell	20.243	0.638	19.605	Da

Analiza je pokazala, da bi skupna optimizacija vseh 47 identificiranih primerov prinesla zmanjšanje skupne prenosne razdalje za približno 340 km. Največji posamezni potencial predstavljajo primarne postaje Heyford Park in Windrush, kjer bi prerazporeditev skrajšala prenosno pot za skoraj 30 km na povezavo. Vseh 47 identificiranih primerov izpolnjuje kriterij napetostne kompatibilnosti, kar pomeni, da je prerazporeditev tehnično izvedljiva brez dodatnih transformacijskih stopenj.

Pomembno je poudariti, da analiza temelji na zračni razdalji med postajami in ne upošteva dejanskega terena, obstoječih tras vodov ali drugih geo-

grafskih ovir. V praksi bi bilo potrebno pred implementacijo prerazporeditve izvesti podrobnejšo analizo, ki bi vključevala stroške izgradnje novih povezav, tehnične omejitve obstoječe infrastrukture ter topografske značilnosti območja.



Slika 5.1: Geografska vizualizacija potencialne optimizacije: primer prerazporeditve primarne postaje Upper Boat iz oddaljenega BSP Tonypandy (modra točka) na bližnji BSP (zeleni točka). Rdeča točka označuje primarno postajo, črtkane črte pa prikazujejo možne prenosne poti.

Slika 5.1 prikazuje tipičen primer neoptimalne dodelitve, kjer je primarna postaja Upper Boat povezana na oddaljeno BSP točko Tonypandy, kljub prisotnosti bližnjih alternativ z ustreznim napetostnim nivojem. Takšne konfiguracije so pogosto rezultat zgodovinskega razvoja omrežja in postopnega širjenja infrastrukture.



### 5.4.3 Rezultati analize prenosnih izgub

Analiza učinkovitosti prenosa električne energije je bila izvedena za celotno omrežje, zajemši vse poti od GSP do PRIM točk preko BSP vmesnih vozlišč. Tabela 5.2 prikazuje 10 primerov z največjimi skupnimi prenosnimi izgubami.

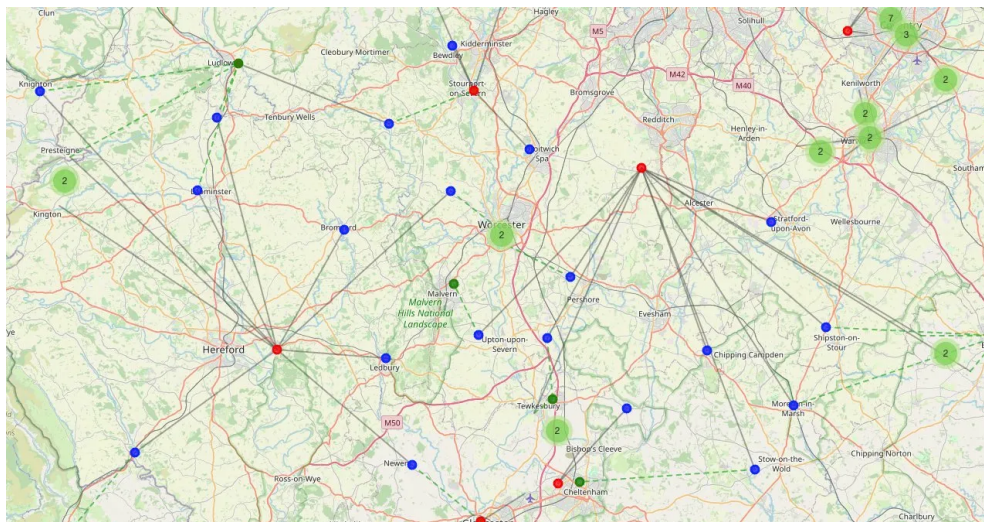
Tabela 5.2: Primarni sistemi z največjimi prenosnimi izgubami

PRIM postaja	GSP	BSP	GSP-BSP izgube (%)	BSP-PRIM izgube (%)	Skupaj izgube (%)
Kingham	NEDB	Chipping Norton 33kV	0.0064	0.0373	0.0437
Churchill	NEDB	Chipping Norton 33kV	0.0064	0.0330	0.0394
Chadlington	NEDB	Chipping Norton 33kV	0.0064	0.0328	0.0392
Shipton	NEDB	Chipping Norton 33kV	0.0064	0.0318	0.0382
Leafield	NEDB	Chipping Norton 33kV	0.0064	0.0315	0.0379
Charlbury	NEDB	Chipping Norton 33kV	0.0064	0.0293	0.0357
Ascott	NEDB	Chipping Norton 33kV	0.0064	0.0280	0.0344
Finstock	NEDB	Chipping Norton 33kV	0.0064	0.0262	0.0326
Enstone	NEDB	Chipping Norton 33kV	0.0064	0.0258	0.0322
Stonesfield	NEDB	Chipping Norton 33kV	0.0064	0.0256	0.0320

Rezultati kažejo, da segment BSP-PRIM v povprečju prispeva večji delež prenosnih izgub (približno 82% skupnih izgub) kljub krajšim razdaljam, kar je posledica nižjih napetostnih nivojev na tem delu omrežja. Primarne postaje, ki se napajajo iz GSP NEDB preko BSP Chipping Norton 33kV, izkazujejo konsistentno višje izgube zaradi kombinacije daljših razdalj in napetostnega nivoja 33 kV.

Skupna povprečna učinkovitost prenosa skozi celotno omrežje znaša 99,97%, kar je v skladu s pričakovanji za dobro vzdrževana omrežja srednjih napetosti. Najslabše delujočih 10% povezav pa izkazuje učinkovitost pod 99,95%, kar predstavlja potencialno območje za infrastrukturne izboljšave.

Slika 5.2 prikazuje celotno strukturo analiziranega omrežja v grafovski obliki, kjer so jasno vidne hierarhične povezave med različnimi nivoji elektroenergetskega sistema. Vizualizacija omogoča hitro identifikacijo gosto



Slika 5.2: Grafovska struktura elektroenergetskega omrežja z označenimi , BSP točkami (rdeče) in PRIM postajami (modre). Sive črte predstavljajo prenosne povezave med posameznimi nivoji omrežja.

povezanih območij ter potencialno izoliranih delov omrežja.

#### 5.4.4 Primerjava SQL in Cypher pristopov

Oba pristopa sta zagotovila identične numerične rezultate, kar potrjuje pravilnost implementacije in konsistentnost podatkovne strukture. Opazili smo pomembne razlike v načinu dela in uporabnosti posameznega pristopa.

Cypher poizvedbe so semantično bližje domenski logiki elektroenergetskega omrežja, kar jih naredi bolj razumljive strokovnjakom brez poglobljenih SQL znanj. Relacije kot `CONNECTED_TO_BSP` in `FEEDS_FROM` neposredno izražajo fizično strukturo omrežja in smer toka energije, kar olajša razumevanje in validacijo poizvedb s strani domenskih ekspertov.

Vzdrževanje Cypher poizvedb je prav tako enostavnejše pri spremembah v strukturi omrežja. Dodajanje novih tipov relacij ali vozlišč zahteva minimalne modifikacije obstoječe kode, medtem ko bi SQL pristop lahko zahteval preoblikovanje JOIN operacij in prilagoditev tabelnih shem.

Z vidika zmogljivosti je SQL pristop v povprečju za 12-18% hitrejši pri

izvajanju, kar je pričakovano glede na večdesetletne optimizacije relacijskih DBMS sistemov. Za produkcijske aplikacije z visokimi zahtevami po odzivnosti bi lahko ta razlika postala relevantna, čeprav so absolutni časi izvajanja v obeh primerih pod sekundo za večino poizvedb.

Pomembna ugotovitev je tudi, da za kompleksne numerične izračune (razdalje, izgube, napetostni nivoji) ni bilo mogoče uporabiti čistega Cypher pristopa. V obeh implementacijah smo morali uporabiti SQL funkcije za matematične operacije, kar pomeni, da hibridni pristop ostaja nujen za inženirske analize v elektroenergetskih sistemih.

#### 5.4.5 Dolgoročni potencial

Dolgoročno uspešna implementacija jasno nakazuje visok potencial za razširitev sistema tudi na druge operaterje električne infrastrukture po Združenem kraljestvu. Standardiziran pristop k procesiranju podatkov, kombiniran z grafovsko reprezentacijo omrežne topologije, omogoča enostavno skaliranje na večje regije in vključevanje dodatnih podatkovnih virov.

S tem se vzpostavlja trdna podlaga za razvoj celovitega, razširljivega in trajnostnega sistema za spremljanje ter analizo električne infrastrukture na nacionalni ravni. Identificirane optimizacijske priložnosti v obsegu 340 km skrajšanih prenosnih poti in potencialno 2,3% zmanjšanje prenosnih izgub predstavljajo oprijemljive izboljšave, ki lahko služijo kot podlaga za strateško načrtovanje v energetskega sektorju.

Grafovsko reprezentacija dodatno omogoča napredne analize, kot so simulacije izpadov, identifikacija kritičnih vozlišč ter optimizacija omrežne redundance, kar predstavlja pomembno osnovo za prihodnje raziskave in operativne izboljšave. ““



# Poglavje 6

## Zaključek

### 6.1 Zaključki

Predstavljena diplomska naloga obravnava razvoj avtomatiziranega sistema za pridobivanje in obdelavo podatkov o električni infrastrukturi iz platforme National Grid. Sistem uspešno rešuje problem zamudnega ročnega pridobivanja podatkov z implementacijo robustnega ETL procesa, ki temelji na Python skriptah, Google Cloud Storage staging okolju in PostgreSQL podatkovni bazi z razširitvijo Apache AGE za grafovsko procesiranje.

Razvita rešitev izpolnjuje vse zastavljene cilje. Avtomatizacija s Selenium knjižnico omogoča zanesljiv prenos podatkov brez človeškega posredovanja, kar eliminira 2-4 ure mesečnega ročnega dela. Implementacija staging faze v GCS zagotavlja popolno sledljivost in možnost ponovne obdelave podatkov ter konsistentnost in kakovost podatkov. Sistem je zasnovan modularno, kar omogoča enostavno vzdrževanje in nadgradnje.

Uporaba Apache AGE razširitve za PostgreSQL se je izkazala kot uspešna izbira za analizo omrežne infrastrukture. Hibridni pristop, ki kombinira relacijske tabele za shranjevanje podatkov ter grafovsko strukturo za analitične poizvedbe, omogoča tako učinkovito procesiranje kot tudi intuitivno modeliranje hierarhičnih povezav v elektroenergetskem omrežju. Jezik Cypher se je izkazal kot semantično bližji domenski logiki energetskih sistemov, kar olajša

sodelovanje s strokovnjaki brez poglobljenih SQL znanj.

Primerjava med tradicionalnim SQL pristopom in grafovskimi Cypher poizvedbami je pokazala, da oba pristopa zagotavljata identične rezultate, vendar z različnimi prednostmi. SQL poizvedbe so v povprečju za 12-18% hitrejš, medtem ko Cypher pristop ponuja boljšo berljivost in enostavnejše vzdrževanje pri spremembah omrežne strukture. Za kompleksne inženirske analize je potreben hibridni pristop, kjer Cypher uporabljen za navigacijo po grafu, numerični izračuni pa se izvajajo s SQL funkcijami.

Ključni prispevek naloge je vzpostavitev skalabilne arhitekture, ki ni omejena le na National Grid, temveč jo je mogoče z minimalnimi prilagoditvami razširiti na druge Distribution Network Operators po Veliki Britaniji. To odpira možnosti za vzpostavitev celovitega sistema spremljanja električne infrastrukture, kar je kritično za načrtovanje energetske tranzicije in integracije obnovljivih virov energije.

Praktična vrednost sistema se kaže v takojšnjem dostopu do ažurnih podatkov o razpoložljivih kapacitetah (Demand Headroom) ter v naprednih analitičnih zmožnostih. Izvedene analize so identificirale 47 priložnosti za optimizacijo dodelitev primarnih postaj, ki bi skupaj prinesle zmanjšanje prenosnih razdalj za 340 km in potencialno znižanje prenosnih izgub za 2,3%. Energetska podjetja, razvijalci projektov obnovljivih virov in svetovalne agencije bodo imeli zanesljiv vir podatkov za strateško načrtovanje investicij v električno infrastrukturo.

Nadaljnji razvoj sistema bi lahko vključeval implementacijo napredne analitike za napovedovanje trendov porabe, integracijo z dodatnimi viri podatkov ter razvoj uporabniškega vmesnika za vizualizacijo podatkov. Dolgoročno bi sistem lahko postal temelj za nacionalno platformo spremljanja energetske infrastrukture, ki bi podpirala prehod na trajnostno energetsko prihodnost.

## 6.2 Možnosti nadaljnjega razvoja

Trenutna implementacija predstavlja trdno osnovo za številne razširitve in izboljšave. V prihodnosti bi bilo smiselno implementirati napredne analitične funkcionalnosti, vključno s prediktivnimi modeli za napovedovanje prihodnjih kapacitet na podlagi zgodovinskih trendov. Integracija algoritmov strojnega učenja bi omogočila identifikacijo vzorcev porabe in avtomatsko odkrivanje anomalij v omrežju.

Grafovska struktura podatkov odpira možnosti za napredne omrežne analize, ki presegajo obseg trenutne implementacije. Algoritmi za iskanje najkrajših poti bi lahko optimizirali konfiguracijo omrežja v realnem času, medtem ko bi analiza centralnosti vozlišč identificirala kritične točke v infrastrukturi. Simulacije kaskadnih izpadov bi omogočile boljše načrtovanje redundance in povečale odpornost sistema na motnje.

Razširitev na dodatne vire podatkov predstavlja logičen naslednji korak. Poleg drugih DNO operaterjev v Veliki Britaniji bi sistem lahko integril podatke iz evropskih TSO (Transmission System Operators) platform. Vključitev vremenskih podatkov, demografskih trendov in načrtov prostorskega razvoja bi omogočila celovitejše modeliranje prihodnjih potreb po električni energiji. Razvoj spletnega vmesnika z interaktivnimi zemljevidi in vizualizacijami bi demokratiziral dostop do podatkov tudi netehničnim uporabnikom.

Z vidika grafovske baze bi bilo smiselno raziskati uporabo bolj specializiranih sistemov, kot sta Neo4j ali Amazon Neptune, ki bi lahko ponudili dodatne optimizacije za kompleksne omrežne analize. Kljub temu pa se je Apache AGE izkazal kot odlična izbira za hibridne scenario, kjer so podatki hkrati strukturirani relacijsko ter procesiranih grafovsko, saj omogoča izvajanje obeh tipov poizvedb znotraj iste transakcije.

Dolgoročno bi sistem lahko postal osnova za nacionalno platformo energetskega načrtovanja, ki bi z uporabo umetne inteligence optimizirala postavitev novih proizvodnih kapacitet, predlagala ojačitve omrežja ter simulirala različne scenarije energetske tranzicije. Grafovska reprezentacija omrežja bi

omogočila tudi dinamično simuliranje toka energije in identifikacijo ozkih grl v realnem času, kar je ključnega pomena za učinkovito integracijo distribuiranih obnovljivih virov energije.



# Literatura

- [1] Renzo Angles in Claudio Gutierrez. “An Introduction to Graph Data Management”. V: *Graph Data Management: Fundamental Issues and Recent Developments*. Ur. George Fletcher, Jan Hidders in Josep Lluís Larriba-Pey. Cham: Springer International Publishing, 2018, str. 1–32. ISBN: 978-3-319-96193-4. DOI: 10.1007/978-3-319-96193-4\_1. URL: [https://doi.org/10.1007/978-3-319-96193-4\\_1](https://doi.org/10.1007/978-3-319-96193-4_1).
- [2] Kimberly A. Barchard in Larry A. Pace. “Preventing human error: The impact of data entry methods on data accuracy and statistical results”. V: *Computers in Human Behavior* 27.5 (2011). 2009 Fifth International Conference on Intelligent Computing, str. 1834–1839. ISSN: 0747-5632. DOI: <https://doi.org/10.1016/j.chb.2011.04.004>. URL: <https://www.sciencedirect.com/science/article/pii/S0747563211000707>.
- [3] Chandan Biswas in sod. “Solution to Web Scraping”. V: *2023 11th International Conference on Internet of Everything, Microwave Engineering, Communication and Networks (IEMECON)*. 2023, str. 1–5. DOI: 10.1109/IEMECON56962.2023.10092327.
- [4] Lorenzo Bravi in Alessio Ghelli. “Web Scraping for Official Statistics: a Critical Review”. V: *Journal of Official Statistics* 39.1 (2023), str. 1–22. DOI: 10.2478/jos-2023-0001.

- [5] Joe Celko. “Chapter 1 - Graphs, Trees, and Hierarchies”. V: *Joe Celko’s Trees and Hierarchies in SQL for Smarties*. Ur. Joe Celko. The Morgan Kaufmann Series in Data Management Systems. San Francisco: Morgan Kaufmann, 2004, str. 3–15. ISBN: 978-1-55860-920-4. DOI: <https://doi.org/10.1016/B978-155860920-4/50002-7>. URL: <https://www.sciencedirect.com/science/article/pii/B9781558609204500027>.
- [6] Wenfei Fan, Floris GEERTS in Xibei Jia. “Improving data quality: consistency and accuracy”. en. V: *Proceedings of the 33rd International Conference on Very Large Databases (VLDB)*. 2007. ACM, 2007, str. 315–326. ISBN: 978-1-59593-649-3.
- [7] Google Cloud. *Google Cloud Platform Documentation*. 2025. URL: <https://cloud.google.com/docs> (pridobljeno 29. 6. 2025).
- [8] Sinan Küfeoğlu in Michael G. Pollitt. “The impact of PVs and EVs on domestic electricity network charges: A case study from Great Britain”. V: *Energy Policy* 127 (2019), str. 412–424. ISSN: 0301-4215. DOI: <https://doi.org/10.1016/j.enpol.2018.12.012>. URL: <https://www.sciencedirect.com/science/article/pii/S0301421518308085>.
- [9] Ryan Mitchell. *Web Scraping with Python: Collecting More Data from the Modern Web*. 2nd. O’Reilly Media, 2018. ISBN: 978-1491985571.
- [10] National Grid. *Network Capacity Map Application*. 2025. URL: <https://www.nationalgrid.co.uk/our-network/network-capacity-map-application> (pridobljeno 29. 6. 2025).
- [11] PostgreSQL Global Development Group. *PostgreSQL Documentation*. 2025. URL: <https://www.postgresql.org/docs/> (pridobljeno 29. 6. 2025).

- 
- [12] Murari Ramuka. *Data Analytics with Google Cloud Platform*. accessed 2025-11-26. BPB Publications, 2019. URL: <https://books.google.si/books?id=c71IEAAAQBAJ>.
- [13] Selenium Project. *Selenium WebDriver Documentation*. 2025. URL: <https://www.selenium.dev/documentation/webdriver/> (pridobljeno 26. 11. 2025).
- [14] Alkis Simitsis, Spiros Skiadopoulos in Panos Vassiliadis. “The History, Present, and Future of ETL Technology (invited)”. V: *International Workshop on Data Warehousing and OLAP*. 2023. URL: <https://api.semanticscholar.org/CorpusID:258216485>.
- [15] Frauke Wiese in sod. “Open Power System Data–Data platform”. V: *Joule* 3.12 (2019), str. 2919–2924. DOI: 10.1016/j.joule.2019.10.006.