# Zagotavljanje kakovosti pri integriranju z zunanjimi viri podatkov

Tadej Volgemut

Databox, Ptuj, Slovenija
tadej.volgemut@databox.com

Zagotavljanje kakovosti pri povezovanju z zunanjimi viri podatkov predstavlja enega najpomembnejših izzivov v podatkovno usmerjenih sistemih, kot je Databox. Raznolikost programskih vmesnikov (ang. application programming interface, API), pomanjkljiva dokumentacija in nenapovedane spremembe zahtevajo natančno načrtovan in prilagojen pristop zagotavljanja kakovosti (QA), ki vključuje tehnične, procesne in vsebinske plasti preverjanja. V prispevku predstavimo, kako pristopamo k zagotavljanju kakovosti integracij: od začetne analize in testiranja povezav, do validacije podatkovne točnosti in konsistentnosti prikaza metrik. Opišemo uvedbo notranjih ogrodij, kot je Integration Testing Framework (ITF), ki omogoča razvijalcem samostojno testiranje že med implementacijo, in uporabo avtomatizacije ter umetne inteligence (AI) za generiranje testnih primerov in zaznavanje anomalij. Poudarek je tudi na vplivu kakovosti na uporabniško izkušnjo in na vlogi QA pri preprečevanju težav v produkciji. Prispevek temelji na praktičnih izkušnjah, ki so rezultat sodelovanja med razvojem in QA ekipo, ter ponuja pogled v smeri večje standardizacije in napredne uporabe AI v prihodnosti.

**Ključne besede:**
zagotavljanje kakovosti,
integracije,
API,
avtomatizirano testiranje,
umetna inteligenca.

Univerzitetna založba Univerze v Mariboru

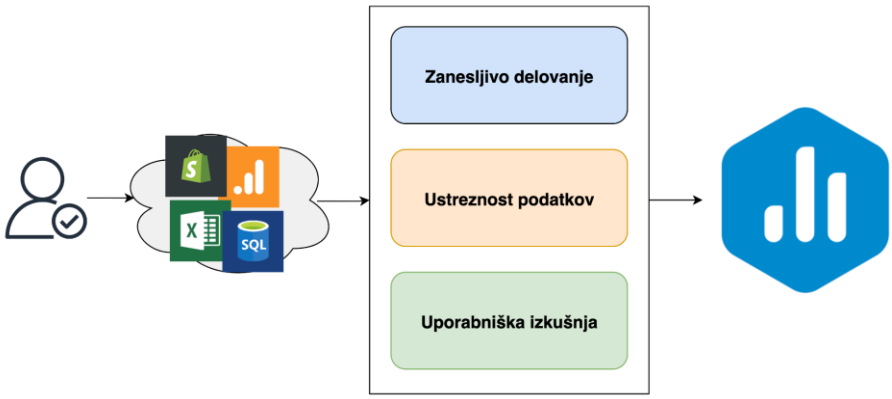
ISBN 978-961-299-037-4
DOI https://doi.org/10.18690/um.feri.7.2025.19

# 1 Uvod

Integracije z zunanjimi viri podatkov predstavljajo hrbtenico podjetja Databox. Uporabnikom ponujamo povezovanje z različnimi viri in s tem dostop do podatkov več kot 100 ponudnikov (npr. Google Analytics 4, Shopify, Excel, baze podatkov) kar skupaj tvori podatkovni ekosistem, na katerem temelji več naših produktov. Glede na raznolikost integracij ter kompleksnost podatkovnih tokov in struktur je zagotavljanje kakovosti ključnega pomena – tako na ravni povezovanja s ponudnikom kot tudi pri nadaljnji obdelavi in prikazu podatkov. V prispevku se osredotočamo na tri ključna področja zagotavljanja kakovosti:

- Zanesljivost delovanja – od konsistentne tehnične implementacije in varne vzpostavitve povezave z zunanjimi ponudniki do stabilnega in zanesljivega prenosa podatkov.
- Točnost in ustreznost podatkov – podatki morajo biti pravilno interpretirani, tehnično in semantično ujemajoči ter umeščeni v ustrezen kontekst uporabe.
- Uporabniška izkušnja – preglednost, intuitivnost in enostavna uporaba so pomembni vidiki kakovosti, ki dopolnjujejo tehnično plat rešitve.

Ker kakovost obravnavamo celostno, jo v prispevku predstavljamo skozi procesne pristope, tehnične rešitve in – posebej – testiranje. Poudarek namenjamo vlogi ekipe za zagotavljanje kakovosti (QA) od začetne analize, skozi implementacijo, do izdaje in vzdrževanja integracije. Izzive in pristope prikazujemo na konkretnih primerih iz prakse.
Raznolikost integracij ter pogosto pomanjkljiva dokumentacija ali nepričakovane spremembe na strani ponudnikov [1] dodatno krepijo pomen sistematičnega pristopa k zagotavljanju kakovosti.
Slika 1 ponazarja, kako trije ključni vidiki kakovosti (zanesljivost, točnost, uporabniška izkušnja) neposredno vplivajo na uspešnost in sprejetost integracij v Databox okolju. Da jih preverjen uporabnik lahko uporablja, in da ga obdržimo, mora njihovo delovanje biti zanesljivo, podatki ustrezni, adopcija pa enostavna in intuitivna.




**Slika 1:** Vloga kakovosti integracije v Databox.

# 2 Zunanji viri podatkov kot temelj ekosistema

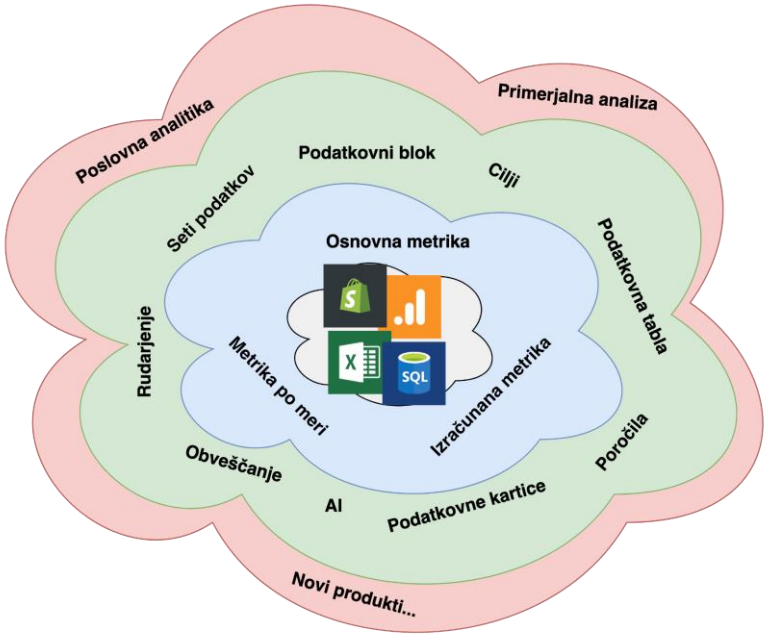
## 2.1. Sistemske razlike med integracijami

Integracije, ki jih ponujamo uporabnikom, so raznolike iz tehničnega in vsebinskega vidika. Razvoj in QA morata imeti v mislih predvsem te lastnosti:

- Preverjanje identitete (avtentikacija): osnovna (npr. *Basic*), ki zahteva uporabniško ime in geslo, ali napredna (npr. *OAuth 2.0*), ki vključuje pridobivanje in uporabo žetona (ang. *token*), pogosta je tudi avtentikacija z API ključem (*API key*), ki zahteva le vnaprej pripravljen ključ.

- Dodeljevanje pravic (avtorizacija): definira kaj lahko uporabnik počne. Za nas so pomembe predvsem bralne (ang. *read-only*) pravice (npr. *read:analytics*).
- Strukturo in obseg vhodnih in izhodnih podatkov: ali vstopamo prek parametrov spletnega naslova (ang. *URL parameters*), ali s podatki v telesu (ang. *body*) sporočila. Izstopni format je lahko različnih tipov (*JSON*, *XML*) in razdeljen na več strani (*paginacija*). Preferiramo paginiran *JSON*.
- Omejitve poizvedb (rate limiting): optimalno bi bilo, da nismo omejeni. Ponavadi pa je to določeno s številom zahtev, ki jih lahko naredimo v danem času.
- Svojevrstno tolmačenje podatkovne semantike: kar je pogosto še bolj kritično. Tako lahko dve integraciji, ki na prvi pogled podajata enako metriko (npr. "št. uporabnikov"), v praksi merita popolnoma različne stvari.

Skupni imenovalec vseh ponudnikov so njihovi podatki, ki jih ovrednotimo, oplemenitimo in ponudimo uporabniku. Slika 2 prikazuje konceptualno pot podatka: od njegove pridobitve iz zunanjega vira, prek pretvorbe v metriko, do prikaza v posameznem produktu znotraj Databox okolja.




**Slika 2:** Podatek gre v metriko, ki je osnova za funkcionalnost na nekem produktu.

## 2.2. Skupni QA pristopi

Določeni pristopi, ki jih uporabljamo pri strukturiranju kode, obravnavi napak in validaciji podatkov, so podrobneje opisani tudi v tehničnem blogu [1]. Ključni poudarki so naslednji:

- Koda mora biti ustrezno strukturirana in implementirana po smiselnih vzorcih, le tako je lahko dodajanje novih in vzdrževanje obstoječih integracij hitro in učinkovito.
- Delegiranje napak mora biti sistemsko - uporabnik se ne sme znajti v situaciji, ko ne ve kaj je šlo narobe (npr. napaka na avtentikaciji, nezadostne pravice za ogled določenega podatka, ali presežena meja poizvedb v nekem času).
- Preverjanje pravilnosti podatkov pomeni, če npr. metrika Total users prikazuje podatke za vse uporabnike. Točnost na drugi strani pove če se vrednosti, ki jih pokažemo uporabniku ujemajo s tistimi na izvorni platformi (npr. Google Analytics dashboard).

- Notranja orodja in ogrodja (ang. *frameworks*) pomagajo pri hitrosti in zanesljivosti prejšnjih treh točk. V naslednjih poglavjih bomo podrobneje predstavili ogrodje, ki omogoča posnetek metrike in primerjavo z živimi podatki.

Naše izkušnje kažejo, da največ težav ne povzroča tehnična izvedba sama po sebi, temveč funkcionalnosti, ki pogosto niso dokumentirane ali so implementirane v nasprotju z ponudnikovo dokumentacijo. Velik izziv predstavljajo tudi večje spremembe, ki jih ponudniki včasih objavijo zelo pozno, ali jih sploh ne. Zato QA pristop pri integracijah ne more biti generičen – vsak primer zahteva prilagojeno testiranje, pogosto tudi ročno analizo in kontekstualno razlago.
Kljub tehnični raznolikosti integracij uvajamo enoten nabor preverjanj, ki tvorijo osnovo našega pristopa k zagotavljanju kakovosti. Ti koraki vključujejo:

- Preverjanje povezave – osnovna validacija avtentikacije in avtorizacije ter uspešne vzpostavitve povezave z virom podatkov.
- Validacija podatkovnega toka – preverjanje, ali integracija pravilno pridobi ključne metrike v ustrezni strukturi in obsegu.
- Testiranje zgodovinskih podatkov – preverjanje, ali API vrne celovit odziv za daljše časovne obsege ter ustrezno podpira paginacijo.
- Semantična skladnost metrik – usklajenost med pridobljenimi podatki in pričakovano vsebino, kot jo prikazuje izvorna platforma.
- Regresijsko preverjanje – primerjava trenutnih rezultatov z vnaprej zajetimi posnetki s čimer odkrivamo morebitne napake pri izboljšavah.

Ti koraki se izvajajo s pomočjo notranjih orodij in testnih ogrodij, kot je Integration Testing Framework (ITF), in predstavljajo sistematiziran pristop, ki ga uspešno uporabljamo tudi pri kompleksnih integracijah – na primer pri integraciji z Google Analytics 4, kjer se pokaže večina omenjenih izzivov.

### 2.3. Poseben primer: GA4 testni pristop

Kot konkreten primer predstavljamo povezovanje z zunanjim virom Google Analytics 4 (GA4), kjer se združujejo številni izzivi, značilni za kompleksne API-je: avtentikacija prek protokola OAuth 2.0, poizvedovanje prek strukturiranih zahtevkov ter obravnava kvot in omejitev. V nadaljevanju opisujemo, kako v praksi izvajamo preverjanje kakovosti pri tej integraciji.

**Avtentikacija in avtorizacija**
GA4 uporablja protokol OAuth 2.0, pri čemer mora uporabnik najprej izvesti prijavo v svoj Google račun. Sistem nato prejme:

- žeton za dostop (ang. *access token*), ki omogoča začasni dostop do API-ja,
- žeton za osvežitev (ang. *refresh token*), ki omogoča avtomatsko pridobivanje novega dostopnega žetona, brez ponovne prijave uporabnika.

Dovoljenja se določijo ob avtentikaciji z zahtevanim bralnim (ang. *readonly*) dostopom. Ta obseg dovoljuje zgolj branje analitičnih podatkov in predstavlja osnovni primer avtorizacije.

**Struktura vhodnih in izhodnih podatkov**

Pri integraciji z Google Analytics 4 uporabljamo Google Data API, kjer je osnovna metoda za pridobivanje metrik *runReport*. Ta metoda omogoča prilagojeno poizvedbo metrik in dimenzij glede na časovni razpon, filtre in parametre sortiranja. Uporablja se za večino standardnih poročil in je zato osrednji del QA režima znotraj naše integracije. Podatke pošiljamo v strukturirani obliki (*JSON*), ki vključuje:

- identifikator vira podatkov (npr. `property: "properties/123456"`) - nastavimo v parametru URL,
- seznam dimenzij in metrik (npr. `"dimensions": [{ "name": "date" }], "metrics": [{ "name": "activeUsers" }]`),
- filtriranje in razvrščanje.

Primer 1: Vhodni podatki za poizvedbo GA4 RunReport

```
{
  "dimensions": [{ "name": "date" }],
  "metrics": [{ "name": "activeUsers" }],
  "dateRanges": [{ "startDate": "2024-01-01", "endDate": "2024-01-31" }]
}
```

Primer 2: Izhodni podatki po uspešni poizvedbi

```
{
  "dimensionHeaders": [{ "name": "date" }],
  "metricHeaders": [{ "name": "activeUsers" }],
  "rows": [
    {
      "dimensionValues": [{ "value": "2024-01-01" }],
      "metricValues": [{ "value": "1243" }]
    }
  ]
}
```

Pri preverjanju kakovosti je ključna skladnost med tem, kar pričakujemo (npr. po vizualizaciji na strani Google Analytics), in tem, kar vrne API. Preverja se:

- celovitost podatkov (ali so zajeti vsi dnevi),
- ujemanje metrik (ali vrednosti ustrezajo vmesniku),
- doslednost strukture odziva tudi pri praznih ali delno napolnjenih rezultatih.

**Omejitve poizvedb**

GA4 API omejuje število zahtevkov na uporabnika, projekt in kombinacijo dimenzij/metrik:

- 10 zahtevkov na sekundo na uporabnika,
- 50.000 vrstic podatkov na dan,
- Quota Error (HTTP 429) je vrnjen, ko presežemo omejitve.

Zato je nujno, da testno okolje zajema primere:

- kako aplikacija ravna ob napaki *HTTP 429 Too Many Requests*,
- ali so implementirani algoritmi za ponovni poizkus z zakasnitvijo (ang. *backoff*),
- in ali lahko ob preobremenjenosti sistem degradira svojo funkcionalnost (npr. prednaloženi podatki - ang. *cache*).

Pri testiranju take integracije vključujemo naslednje vidike:

- preverjanje veljavnosti žetonov ter ukrepanje v primeru njihove neveljavnosti (npr. zaradi poteka veljavnosti, preklica ali napake),
- preverjanje pravilnosti vhodnih podatkov (zahtevki z napačno strukturo, manjkajoče metrike),
- preverjanje odziva API-ja ob napačni avtorizaciji (npr. odstranjen obseg),
- simulacijo povečanega prometa, ki povzroči dosego rate limita.

# 3 Procesi zagotavljanja kakovosti integracij z zunanjimi viri

## 3.1. Priprava in analiza API-ja

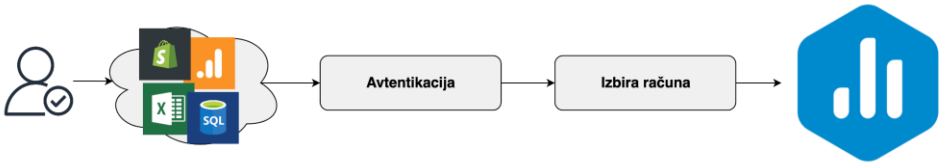
Pred pričetkom implementacije integracije se API željenega ponudnika skrbno preveri ali ustreza kriterijem vključitve. Pregleda se dostopna dokumentacija in vzpostavi nekaj inicialnih poizvedb, ponavadi kar prek orodja za delo z API-ji, Postman [2]. Z analizo rezultatov integraciji določimo lastnosti, ki smo jih opisali na začetku prejšnjega poglavja.

## 3.2. Usmeritve QA v začetni fazi

Za QA je pomembno, da se vključimo že v tej fazi, saj tako dobimo prvi vtis o kvaliteti in delovanju API-ja, količini in naravi metrik, ki jih bomo preverjali, predvsem pa imamo možnost opozoriti na potencialne napake, ki bi se lahko zaznale šele med funkcionalnim testom. Vsebinska in tehnična pričakovanja uskladimo neposredno z razvojnim oddelkom, po potrebi pa tudi z vodjo projekta. Vse to je ključno pri podajanju ocene testiranja in pripravi testnega plana.

## 3.3. Skriptiranje in avtomatizacija povezave

Kasneje si QA pripravi podrobnejši seznam metrik, ki jih bo preverjal ter naredi podporne skripte in avtomatske teste. S skriptami si pomagamo pri analizi podatkovnih struktur, ki jih API vrača. Na primer, že zgodaj lahko preverimo, če poizvedba vrne vse podatke v enem odgovoru, ali večih (paginacija), če so vključeni vsi podatki izbranega časovnega okvirja in ali je struktura podatkov pravilna. Z avtomatskimi testi pa želimo avtomatizirati ponavljajoče delo in pokriti najbolj kritična področja, na primer povezovanje z računom integracije (najpogosteje pri integracijah z *API key* avtentikacijo), s čem potrdimo da povezovanje integracije s ponudnikom deluje, kar je pogoj, da lahko začnemo pridobivati podatke. Slika 3 prikazuje osnovni potek povezovanja integracije z zunanjim virom, kjer QA s pomočjo avtomatskih testov preveri, ali je povezava uspešna in ali sistem lahko prične z zajemom podatkov.




**Slika 3:** Uporabnik poveže svojo integracijo z Databox.

## 3.4. Preverjanje točnosti in ujemanja metrik

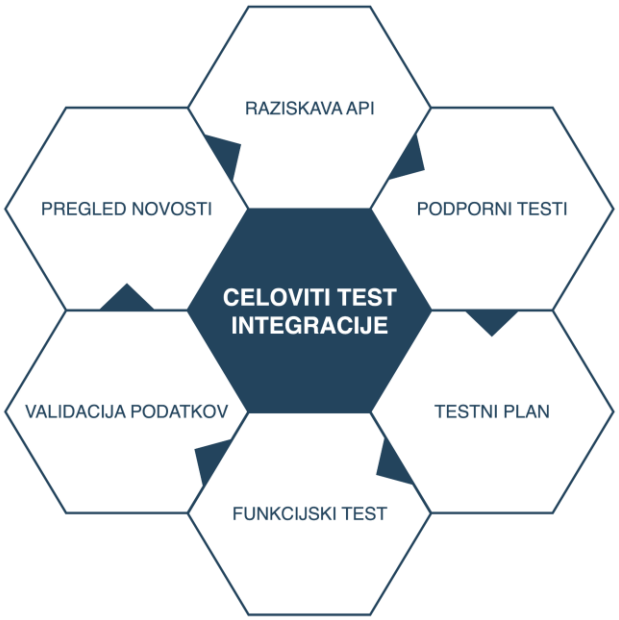
Velik delež testov zaradi raznolikosti še vedno ostaja ročen, in to je preverjanje točnosti podatkov. Zraven ustreznosti (format, časovna cona, uporaba enot, opis metrike), kar lažje avtomatiziramo, je pomembno, da se

pridobljeni podatki ujemajo z izvornimi. Pri veliki količini metrik je ta korak obsežen, in zahteva vzporedno primerjavo podatkov na naših vizualizacijah s prikazom na ponudnikovem uporabniškem vmesniku.
Pri podatkih izstopata 2 izziva:

- Vsi podatki niso na voljo na obeh straneh: zgodi se, da API vrača več, kot prikazuje ponudnik na svojem uporabniškem vmesniku. Točnosti takšnih podatkov ne moremo preveriti zato se velikokrat zanašamo na uporabniški odziv.
- Podatki niso enotni že pri ponudniku: tukaj lahko gre za (a) različne interpretacije metrike v API-ju in uporabniškem vmesniku ponudnika ali (b) napako. V tem primeru je ključno poznavanje dokumentacije in komunikacija z ponudnikom, kar z našo pomočjo opravi vodja projekta.

### 3.5. Komunikacija in prioritizacija

Tukaj je ključna komunikacija tako z vodjem projekta kot z razvojnim oddelkom. Uporabniški vmesniki ponudnikov so velikokrat kompleksni in iskanje metrik je le redko intuitivno. Ker je razvojni oddelek skozi začetno raziskavo že deloma spoznal vmesnik, nam s svojim znanjem lahko prihrani ogromno časa. Ker smo ponavadi omejeni z viri za testiranje, se je večkrat potrebno uskladiti glede pomembnosti metrik, in skladno s tem sestaviti testni plan. Če zunanji vir prek API-ja ponuja večje število metrik (npr. 125), ekipa za zagotavljanje kakovosti v sodelovanju z razvojem določi ključne metrike, ki jih preverimo podrobneje. Preostale vključimo v širši nabor osnovnih preverjanj (dimni test), pri katerih ne izvajamo ročne validacije vrednosti, temveč zgolj preverimo njihovo prisotnost in tehnično skladnost. Integracije z večjim povpraševanjem imajo ponavadi tesnejše roke dostave, kjer prav tako moramo biti pragmatični.
Celoviti test integracije kot ga prikazuje slika 4, se teoretično ne zaključi in je živ, dokler jo ponujamo uporabnikom. Zelo pomemben del je spremljanje novosti, kar pomeni preverjanje objav ponudnikov za večjimi spremembami, ki bi lahko vplivale na našo integracijo, ali nenazadnje o novih metrikah, ki bi jih lahko ponudili. Za vsako integracijo imamo vzpostavljen kanal, kamor prihajajo takšne novosti (ponavadi je to RSS naročnina, ali pa vsaj prijava na elektronske novice). QA te kanale redno spremlja in potencialne spremembe skomunicira z vodjem integracijskega oddelka.




**Slika 4:** Celoviti test, ki se začne z raziskavo, in živi dokler je integracija omogočena v Databox.

# 4 Tehnični pristopi in avtomatizacija

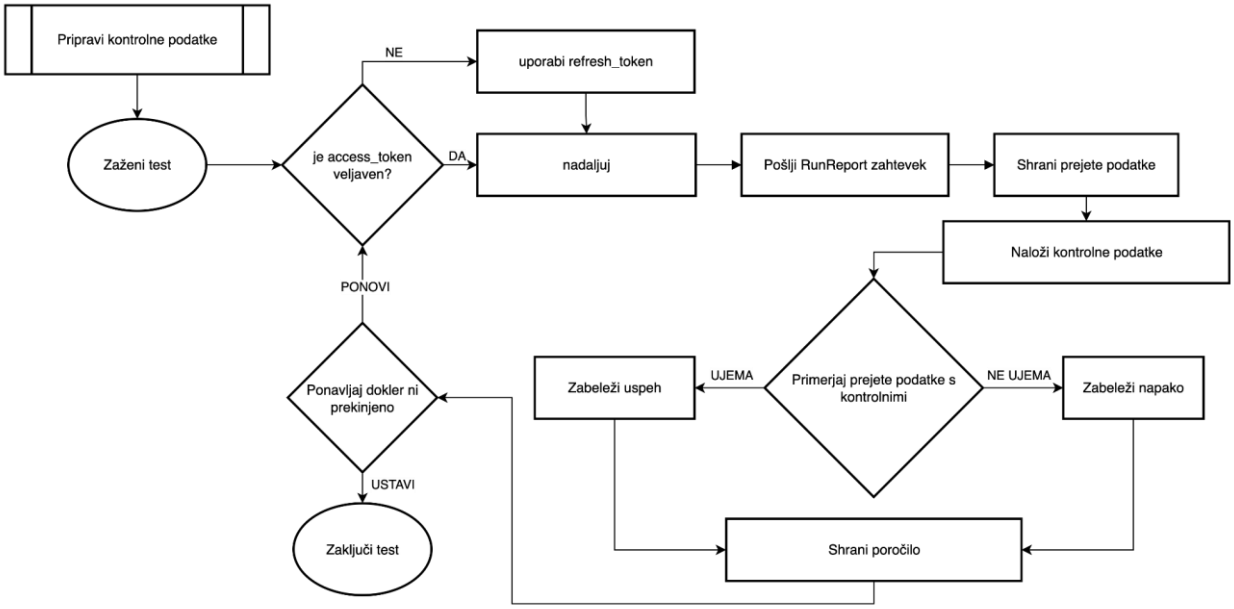
Za učinkovito zagotavljanje kakovosti je dobro, da je QA dovolj tehnično podkovan, saj lahko le tako celostno pripomore k razvoju. Razvili smo interno ogrodje za preverjanje podatkov, ki je namenjeno razvijalcem integracij, ob enem širimo pokritost testnih primerov z avtomatskimi testi, in sledimo AI trendom.

## 4.1. Interno ogrodje za preverjanje podatkov ITF

QA je zasnoval ogrodje za preverjanje točnosti in pravilnosti podatkov (Integration Testing Framework, ITF). Gre za dodano funkcionalnost, ki je pogojno vključena v poslovno logiko pridobivanja in priprave podatkov. Deluje tako, da se najprej posname poizvedba na API, nato pa se sestavi test enot (*unit test*), ki žive podatke primerja s posnetkom. Test se požene po vsaki večji spremembi, praviloma na strani razvijalca, in je vključen tudi v proces neprekinjene integracije in dostave (*CI/CD*). Tu se testi sprožijo samodejno in omogočajo hitro zaznavo nepričakovanih sprememb v metrikah. Ogrodje zagotavlja, da poseg v poslovno logiko ni vplival na točnost in pravilnost podatkov. Njegov cilj je razvijalcem omogočiti preverjanje ustreznosti kode že med implementacijo, zmanjšati odvisnost od ekipe za zagotavljanje kakovosti ter povečati zaupanje v stabilnost obsežnejših sprememb. Ob enem pa je bil namen tudi zmanjšati obseg končnega dela za QA, ter pohitriti izdajo. Uporaba se je izkazala za učinkovito, predvsem pri ponudnikih z velikim številom metrik, kot je Google Analytics. Prototip in znanje je QA predal integracijskemu oddelku, ki je ogrodje uspešno vkomponiral v svoj razvojni cikel.

## 4.2. Avtomatski testi

Prve testne primere smo avtomatizirali z orodjem Katalon Studio [3], ki je omogočalo relativno dobro izvedbo. Kljub temu smo se odločili za prehod na ogrodje Playwright [4], saj bolje ustreza našim potrebam po *zmogljivosti*, *hitrosti izvajanja*, *prijaznosti razvijalcem* in podpori več programskih jezikov. Playwright omogoča učinkovite teste tipa konec-konec (ang. *end to end, E2E*) v modernih brskalnikih, ima veliko razvojno skupnost in je zaenkrat brezplačen. Implikacijo v še eno ogrodje, namenjeno tudi razvijalcem, predstavlja naš prispevek iz leta 2023 [5].
Vedno več virov usmerjamo v avtomatizacijo testnih primerov v *Playwright*, saj lahko le tako ohranjamo ali celo zvišujemo trend dostave v hitrosti in kvaliteti. Slika 5 prikazuje avtomatiziran tok testiranja integracije GA4 z uporabo orodja Playwright. Test vključuje preverjanje avtorizacije, zajem podatkov in validacijo rezultatov v primerjavi s pričakovanimi vrednostmi. Kontrolni podatki se pripravijo ročno. Vstop v test je lahko enkratni ročni zagon, ali pa akcija na *CI/CD*. Najprej se preveri veljavnost žetona, nato se pridobi podatke in se jih preveri.




**Slika 5:** diagram poteka avtomatiziranega testa za GA4, ki pridobi in preveri podatke.

Zraven Playwright oddelki ali posamezni inženirji uporabljajo še druge tehnologije. Iz vidika proaktivnosti, kreativnosti in samostojnosti je to zelo dobrodošlo, se pa QA vseeno zavzema za standardizacijo postopkov in orodij, vzpodbuja k deljenju znanja in nestandardnih primerov uporabe. Tak primer je uporaba ogrodja za obremenitveno testiranje *K6* [6], ki počasi postaja formalizirana, saj bo koristna predvsem za potrebe zagotavljanja konsistence pri obravnavi velike količine podatkov.

### 4.3. Umetna inteligenca

Umetna inteligenca (ang. *Artificial Intelligence, AI*) je zagotovo nepogrešljiva in z velikim potencialom, tudi na področju QA. Vzpodbujamo celosten AI pristop, kar pomeni da ga želimo vključevati na vseh nivojih dela. Od priprave testnih planov, generiranja testnih primerov, do preoblikovanja in izboljšav testnih ogrodij in izdelave virtualnih pomočnikov (ang. *bot*). Predstavili bomo dva.

**Izdelava testnih primerov in avtomatskih testov s pomočjo AI**

Vsebine pomoči na Databox so obsežne, predvsem za integracije, ki jih ponujamo uporabnikom. To predstavlja ogromno količino znanja, ki ga lahko posredujemo AI in relativno hitro dosežemo visoko stopnjo pokritosti. Z ustreznim zahtevkom (ang. *prompt*) ki zajema referenco do naših vsebin, smo npr. v kratkem času dobili testne scenarije, ki smo jih lako dalje pretvorili v uvozno datoteko, katero smo naložili v sistem za upravljanje s testi (ang. *Test management tool*). Iz predlaganih testnih korakov nam je AI pomagal tudi pri generiranju avtomatskih testov na različnih stopnjah kvalitete integracije. Pri tem je seveda ključno zavedanje, da se tudi AI lahko zmoti, zato je bila procesna umestitev preverjanja ustreznosti generiranih rezultatov nujna. S tem ja QA postal veliko hitrejši in agilnejši.

**Nastavitev in treniranje virtualnega QA pomočnika za testiranje integracij**

Dodaten vir znanja, ki ga lahko predamo AI je javna dokumentacija ponudnikov, koda naših implementacij, in nenazadnje naše dosedanje izkušnje. S tem izhodiščem smo ustvarili virtualnega pomočnika (ang. *AI bot*), ki ga sproti učimo z vodenjem in dodajanjem ključnih informacij. Ta pomočnik je nepogrešljiv sogovornik pri zagotavljanju kvalitete novih integracij, ali testiranju sprememb na starejših. Pomagal je npr. identificirati neočitne pasti, opozoriti na prihajajoče večje spremembe API-ja, in preverjati ustreznost in točnost podatkov. Prednost

pristopa s pomočnikom je ta, da je na voljo ostalim članom, se sproti uči, in lahko dobi res širok spekter znanja. Med tem, ko smo spet pridobili na hitrosti in učinkovitosti, seveda ne smemo pozabiti na preverjanje generiranih rezultatov.

## 5 Vpliv kakovosti na uporabniško izkušnjo

Ko uporabnik pride k nam, ga želimo obdržati, zato je ključna dovolj dobra uporabniška izkušnja (ang. *UX*). Ogromni sistemi, predvsem v domeni poslovne inteligence (*ang. Business Intelligence - BI*), so kompleksni že sami po sebi. Slaba uporabniška izkušnja v obdobju, ko si posameznik ali podjetje ne more privoščiti dolgotrajnega uvajanja in čakati na odpravljene napake, negativno vpliva na uporabniški cikel in rast podjetja. Pri konkurenci opazimo veliko slabih praks. Tudi dobre so, katere želimo posnemati, ali biti še boljši.

Primeri slabih izkušenj pri nas niso pogosti, pa vseeno niso zanemarljivi. Na primer ponavljajoče večje odstopanje pri podatkih uporabnika, kritične napake v času predstavitve partnerjem, prekinjene povezave, izdaje z napakami, … zagotovo ne vplivajo dobro na zadovoljstvo naših uporabnikov.

Slabe izkušnje poskušamo omiliti z dobrimi praksami in aktivnim odnosom do uporabnika. Zato komuniciramo z uporabnikom, ki ima težave, poskušamo razumeti njegovo frustracijo, mu vsebinsko pomagati, in kar je najpomembneje, ga poskušamo obdržati, ko zaznamo da želi oditi. Na primer, pri integraciji z Facebook (Meta) API-jem smo uporabniku v manj kot dveh urah posredovali razlago napake, popravek pa je bil vključen v naslednjo izdajo.

QA ima tudi tukaj pomembno vlogo. Sodeluje npr. pri iskanju ponovitve redke napake pri veliki stranki, z hitrim odzivom doprinese k hitri in kvalitetni izdaji popravkov, celostno prispeva k razvoju procesov in ogrodij, ki dolgoročno izboljšujejo uporabniško izkušnjo.

## 6 Zaključek

V prispevku smo predstavili tehnične in procesne pristope zagotavljanja kakovosti pri povezovanju z zunanjimi viri podatkov v sistemu Databox. Poudarili smo pomen razumevanja raznolikosti API-jev, potrebe po semantični skladnosti podatkov in vlogo QA pri celotnem življenjskem ciklu integracije – od načrtovanja do vzdrževanja.

Naši rezultati kažejo, da sodelovanje med razvojem in QA bistveno izboljša učinkovitost, zanesljivost in hitrost integracij. Posebej se je izkazalo, da lahko z notranjimi ogrodji (kot je ITF) ter avtomatskimi testi razvijalci že v zgodnji fazi validirajo pravilnost svoje implementacije, kar razbremeni QA in skrajša čas do izdaje.

V prihodnosti želimo nadgraditi vpeljane prakse z večjo stopnjo standardizacije in poglobljeno uporabo umetne inteligence – tako pri generiranju testnih primerov kot pri odkrivanju sprememb. Želimo tudi nadaljevati z odpiranjem QA orodij širšemu razvojnemu ekosistemu in z boljšim povezovanjem uporabniške povratne informacije v proces kakovosti. Verjamemo, da lahko predstavljene izkušnje in pristopi služijo kot vzorčni model tudi za druge sisteme, ki se soočajo z izzivi integracije podatkov iz zunanjih virov.

## Literatura

[1] Databox, Our methods and challenges of API integration, https://databox.com/our-methods-and-challenges-of-api-integration, obiskano julij 2025.

[2] https://www.postman.com/, Postman, obiskano julij 2025

[3] https://katalon.com/, Katalon, obiskano julij 2025

[4] https://playwright.dev/, Microsoft, obiskano julij 2025

[5] OTS 2023 Sodobne informacijske tehnologije in storitve: Zbornik 26. konference, Testno ogrodje za razvijalce - ali kako doseči, da razvijalci celovito testirajo, Mitja Krajnc, Tadej Ciglič, Boris Ovčjak, Databox

[6] https://k6.io/, Grafana Labs, obiskano julij 2025