

Ko umetna inteligenca prevzame razvoj internih orodij

Gregor Kovač, Tomaž Kozlar, Žan Žvokelj

Bilumina d.o.o., Maribor, Slovenija

gregor.kovac@bilumina.com, tomaz.kozlar@bilumina.com, zan.zvokelj@bilumina.com

V lanskem prispevku smo predstavili lastno low-code platformo za razvoj ERP sistema, zgrajeno na jeziku SQL in jedru v Javi, z orodjema FormTool in Linea. Letos preverjamo, ali lahko umetna inteligenca postane resnična pomoč pri razvoju in vzdrževanju naših internih orodij. Odgovor najprej iščemo na primeru QueryToola, ki ga s pomočjo orodja za agentno programiranje prepišemo iz obstoječega okolja na spletno platformo Laravel. Umetna inteligenca nam je pomagala tudi pri drugih internih orodjih, kot so spremljanje, postavitev strežnikov, pomoč uporabnikom in podobno. Izkušnje umestimo v širši, inženirski in etični okvir ter izpostavimo, kje je bila pomoč umetne intelligence produktivna in kje je vodilno vlogo ohranil razvijalec.

Ključne besede:

umetna inteligenca

low-code

Laravel

QueryTool

Linea

1 Uvod

V prispevku na lanski konferenci OTS 2025 smo predstavili lasten low-code pristop k razvoju kompleksnega in visoko konfigurabilnega ERP sistema [1]. Pristop temelji na programskem jeziku SQL in jedru, napisanem v Javi, podpirata pa ga dve interni orodji: FormTool za oblikovanje vnosnih mask in Linea za načrtovanje podatkovne baze. Na ta način smo doslej zgradili več kot tisoč vnosnih mask in več tisoč baznih objektov, ob tem pa skrajšali čas razvoja in zmanjšali možnost napak.

V preteklem letu se nam je vse pogostejše zastavljalo vprašanje, ki si ga danes zastavlja velik del razvojne stroke: ali lahko umetno inteligenco, ki je postala izrazito zmogljiva in dostopna, uporabimo kot resnično pomoč pri razvoju in vzdrževanju naših internih orodij – ne le kot pripomoček za dopolnjevanje kode, temveč kot sodelavca pri večjih, dobro opredeljenih nalogah?

2 Prepis QueryToola na Laravel s pomočjo umetne inteligence

2.1. Kaj je QueryTool in zakaj smo ga izbrali za pilotni projekt

QueryTool je naše interno orodje za razčlenitev stavkov SQL na sestavne komponente: stolpce v stavku SELECT s pripadajočimi podatkovnimi tipi, pogoje v stavku WHERE z operatorji, tabele in poglede v delih FROM in JOIN, parametre (npr. :ID, :COUN_ID), agregacijske in skupinske klavzule (GROUP BY, HAVING), urejanje (ORDER BY) ter ugnezdene poizvedbe. Razčlenjena struktura se uporablja na več mestih v rešitvi ERP – pri povezovanju parametrov SQL in rezultatov poizvedbe s polji na vnosni maski. Ker je orodje funkcionalno zaokroženo, smo ga prepoznali kot idealnega kandidata za pilotni projekt.

QueryTool je napisan v programskem jeziku Java s pomočjo javanskega ogrodja GWT (Google Web Toolkit).

Action code	SQL
CALL_LOGIC.CREATE_CONTACT_MANUAL	CALL LOGIC.CREATE_CONTACT(:PART_ID , :WJPO_ID , :NAME , :SURNAME , :PHONEJOB , :PHONEHOME , :PHONEMOBILE , :MAILJO
CHECK_PCON_MAIN_QUERY_EXTENSION	SELECT PCCA.PCCA_CODE, GEND.GEND_CODE, GEND.GEND_NAMESMALL, POST.POST_CODE, POST.POST_CITY, WJPO.WJPO_CC
EXECUTE_ITEM_LIST_MANUAL_QUERY	SELECT ITEM_ACTIVE, ITEM_CODE, ITEM_CREATED, ITEM_DESIGNOWNER, ITEM_DESCRIPTIONLARGE, ITEM_DESCRIPTIONSMALL
PARTNER_SEARCH_ALL_MANUAL	SELECT PART.PART_ID, PART.PART_CODE, PART.PART_TITLES SMALL, PART.PART_TITLELARGE, PCON.PCON_STREET AS PART_STRE
EXECUTE_T101_NOTEXISTING_QUERY_EXT...	SELECT T101.T101_AGEFROM, T101.T101_AGETO, T101.T101_ASSORTMENT, T101.T101_BARCODE, T101.T101_BRAND, T101.T101_BRANDNA
EXECUTE_USER_AUTO_QUERY	SELECT USER.MODU_ID, USER.MODU_MODU_CODE, USER.MODU_MODU_ID, USER.MODU_MODU_NAMESMALL, USER.MODU_PART_I
EXECUTE_T101_MANUAL_QUERY	SELECT T101.ITEM_ID, T101.ITEM_CODE, T101.ITEM_NAMESMALL, T101.T101_AGEFROM, T101.T101_AGETO, T101.T101_ASSOR

Param. code	Type name	Default value	F1 where fix	Equals sign (IN...	Field code	Type name	Default value	F1 update field
BONUSVALUE...	DECIMAL				BUT_QUANTITY	DECIMAL		
BONUSVALUE...	DECIMAL				COUN_CODE	VARCHAR		
BUTY_ID_LOY...	BIGINT				COUN_ID	BIGINT		
FETCH	BIGINT				COUN_NAMESMALL	VARCHAR		
MAIN	SMALLINT				GEND_CODE	VARCHAR		
					GEND_ID	BIGINT		
					GEND_NAMESMALL	VARCHAR		
					NUMBER_OF_CHILDREN	INTEGER		
					PART_ACTIVE	SMALLINT		
					PART_CODE	VARCHAR		
					PART_CODEOLD	VARCHAR		

Slika 1: QueryTool

Vse podatke, ki jih vidite na sliki 1, uporabljamo v ERP aplikaciji pri izvajanju akcij, na katere je ta poizvedba vezana, in za preverjanje podatkov.

Ta rešitev ima tudi omejitve, ki jih sčasoma nismo odpravili iz različnih razlogov. Ena od takšnih omejitev je tudi, da orodje ne zna pridobiti ustreznih podatkov iz procedure v podatkovni bazi.

2.2. Razvojni proces s pomočjo umetne inteligence

2.2.1. Kako deluje velik jezikovni model

Preden opišemo sam potek dela, na kratko povzemimo, kako deluje velik jezikovni model (LLM), saj to pojasni tako njegove prednosti kot omejitve, na katere smo naleteli. LLM je model, naučen na velikih količinah besedila, ki deluje po enem osnovnem načelu: na podlagi dosedanjega besedila napoveduje najverjetnejši naslednji »žeton« (angl. token) – delček besede. Besedilo se najprej razbije na žetone, iz njih pa model zaporedoma gradi odgovor, žeton za žetonom.

Ključni gradnik je mehanizem pozornosti (angl. attention): pred izbiro vsakega novega žetona model presodi, kako pomemben je vsak od dosedanjih žetonov. Ta pozornost je omejen proračun – njena skupna teža je stalna in se porazdeli med vse žetone. Ko je konteksta malo, pomemben podatek dobi znaten delež pozornosti; ko je konteksta veliko, se isti podatek utopi med tisoči drugih [3]. To je neposreden vzrok za razpadanje konteksta in razlog, da se je pri prepisu obneslo delo z majhnimi, jasno omejenimi enotami.

Model nima baze dejstev, temveč statistične vzorce iz učnih podatkov. Zato je lahko izjemno tekoč in prepričljiv, hkrati pa lahko samozavestno navede napačen podatek (halucinacija) ali spregleda pomenske podrobnosti, na primer zanikanje [7]. Prav tako nima resničnega razumevanja izvirne kode ali poslovne logike; deluje po vzorcih, ki jih je videl. Za nas iz tega sledita dve praktični posledici: navodila in kontekst morajo biti konkretni in omejeni, vsak rezultat pa je treba preveriti glede na dokazano pravilno referenco – kar podrobneje opišemo v nadaljevanju.

2.2.2. Izbrani izdelki

Po tehtnem premisleku in posvetovanju smo v podjetju izbrali izdelek Claude podjetja Anthropic.

Claude ni vezan na en sam model, temveč omogoča izbiro med več modeli iste družine. To je koristno, ker za vsako vrsto naloge – zahtevno sklepanje, hitro rutinsko delo ali delo z dolgim kontekstom – izberemo najprimernejši model in s tem uravnovesimo kakovost, hitrost in stroške. V času pisanja so na voljo naslednji modeli [11]:

- **Claude Opus 4.8** - najzmogljivejši model za zahtevno agentno programiranje in poslovne naloge; kontekstno okno milijon žetonov,
- **Claude Sonnet 5** - najboljše razmerje med hitrostjo in zmogljivostjo; primeren za vsakodnevno razvojno delo; kontekstno okno milijon žetonov,
- **Claude Haiku 4.5** - najhitrejši model z zmogljivostjo blizu vrhunske; primeren za hitre, pogoste in cenovno občutljive naloge; kontekstno okno 200 tisoč žetonov,
- **Claude Fable 5** - najzmogljivejši širše dostopen model, zasnovan za dolgotrajne (angl. long-running) agente in najzahtevnejše naloge; kontekstno okno milijon žetonov.

Anthropic ponuja tudi specializiran model Claude Mythos 5 za obrambne kibernetkovarnostne naloge, ki pa je dostopen le po vabilu.

Pri prepisu QueryToola smo največ uporabljali model Claude Opus 4.8. Zanj smo se odločili, ker je zasnovan prav za zahtevno agentno programiranje in delo v poslovnem okolju: dobro razume obsežno izvirno kodo in poslovno logiko, njegovo veliko kontekstno okno (milijon žetonov) pa omogoča, da mu naenkrat predstavimo večje dele obstoječega orodja skupaj s pojasnili. Ob tem ponuja dobro razmerje med zmogljivostjo na eni strani ter hitrostjo in stroškom na drugi, kar je pomembno pri dolgotrajnem, iterativnem delu. Del najzahtevnejših nalog smo opravili z zmogljivejšim modelom Claude Fable 5, dokler njegova uporaba ni bila omejena na ameriške državljane; nato smo se v celoti vrnili na Opus 4.8.

2.3. Ponovna implementacija

Brez predhodnih izkušenj z agentnim programiranjem smo začeli po najbolj neposredni poti – z raziskovalnim, sproščenim slogom dela, ki se ga je prijelo ime »vibe coding«.

Najprej je bilo treba pripraviti spletno okolje z naslednjim pozivom:

```
V /var/www pripravi nov imenik queryTool z zadnjo verzijo Laravel-a in login  
ekranom, ki deluje preko Google prijave.
```

Po nekaj iteracijah prejmemo zadnje sporočilo Clauda:

```
Končano. V /var/www/queryTool stoji nov projekt z Laravel 13.18.1 (zadnja verzija) in delujočo Google prijavo.  
Kaj je narejeno  
Ogrodje  
• Svež laravel/laravel (v13.18.1), nameščen laravel/socialite za OAuth.  
• Baza SQLite, migracije pognane.  
Google prijava  
• GoogleAuthController.php — redirect, callback (poišče/ustvari uporabnika po google_id), logout.  
• routes/web.php — /login, /auth/google/redirect, /auth/google/callback, /logout, zaščiten /dashboard.  
• config/services.php — Google blok.  
• Migracija users: dodan google_id (unique), avatar, password je zdaj nullable.  
• User.php — google_id, avatar v fillable.  
• login.blade.php — prijavi zaslon z Google gumbom + dashboard.blade.php.  
• .env / .env.example — GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET, GOOGLE_REDIRECT_URI.  
Preverjeno (dev strežnik)  
• / → 302 na /login ✓  
• /login izriše zaslon z gumbom ✓  
• /dashboard brez prijave → 302 na /login (auth middleware) ✓  
• /auth/google/redirect → 302 na accounts.google.com ✓  
Kar moraš še narediti sam  
Vpiši Google OAuth gesla v .env (GOOGLE_CLIENT_ID + GOOGLE_CLIENT_SECRET iz Google Cloud Console → Credentials, redirect URI  
http://localhost/auth/google/callback), nato php artisan config:clear. Postopek po korakih je v README.md.  
Opomba: imenik je last clauda_██████████. Za serviranje prek www-data (nginx/apache) bo morda treba prilagoditi lastništvo storage/ in  
bootstrap/cache/.
```

Slika 2: Končno sporočilo priprave spletnega okolja

Pri poskusu odpiranja strani dobimo napako, ker prezremo opombo, Claude pa težavo uspešno odpravi.

Dobim tole napako:

```
ErrorException:
tempnam(): file created in the system's temporary directory

at
/var/www/queryTool/vendor/laravel/framework/src/Illuminate/Filesystem/Filesystem.php:22
2
at Illuminate\Foundation\Bootstrap\HandleExceptions->handleError()

(/var/www/queryTool/vendor/laravel/framework/src/Illuminate/Foundation/Bootstrap/Handle
Exceptions.php:263)
at Illuminate\Foundation\Bootstrap\HandleExceptions->
```

Show more

Napaka je klasična: Blade compiler ne more zapisati prevedenega viewa v `storage/framework/views` — torej serviajoči uporabnik (`www-data`, ker gre zdaj prek `https://[redacted]/`) nima pisalnih pravic v `storage/` in `bootstrap/cache/`. Preverim:

Inspected storage permissions and web server user >

Potrjeno: `nginx + php-fpm` tečeta kot `www-data`, `storage/` pa je last `[redacted]` z mode `775` — `www-data` spada le pod „other“ (`r-x`, brez pisanja), zato `tempnam` pade. Popravek: skupino nastavim na `www-data`, dodam `group-write` in `setgid`, da nove datoteke podedujejo skupino:

Fixed `storage/bootstrap-cache` group ownership and perms for `www-data` >

Skupina je zdaj `www-data` s `setgid`. Počistim stare cache in preverim, da prevajanje viewa deluje kot `www-data`:

Cleared caches and tested blade compile as `www-data` >

`www-data` zdaj brez težav prevaja Blade predloge. Počistim še cache, da spremembe (in nov `APP_URL`) veljajo takoj:

Cleared view/config/cache >

Rešeno.

Vzrok

Napaka `tempnam(): file created in the system's temporary directory` je nastala, ker Blade compiler ni mogel zapisati prevedenega viewa v `storage/framework/views`. Aplikacijo zdaj servira `nginx + php-fpm` kot `www-data`, mapa `storage/` pa je bila last `[redacted]` (mode `775`) — `www-data` je spadal le pod „other“ brez pisalne pravice, zato je `tempnam` padel nazaj na sistemski `/tmp` in sprožil izjemo.

(Napaka, ki si jo videl, je bila pravzaprav `renderer` izjema, ki je sam padel pri prevajanju svojega viewa — prava napaka spodaj je bila najbrž ista permission težava.)

Slika 3: Prva napaka in rešitev

Spletno stran ponovno preizkusimo in dobimo naslednjo podobno napako, ki jo Claude prav tako odpravi.

```
Sedaj dobim tole napako:
Illuminate\Database\QueryException
vendor/laravel/framework/src/Illuminate/Database/Connection.php:857
SQLSTATE[HY000]: General error: 8 attempt to write a readonly database (Connection: sqlite, Database:
/var/www/queryTool/database/database.sqlite, SQL: update "sessions" set "payload" =
eyJfdG9rZW40OiJ3WG9ZdHRWd3k5UE9aMWNtMW5pZjBHZ0s5aWY2MjZlZm91dXliX3Y3ZXZp
b3V2LjZpZj7lnVybCj6mh0dHbZCj6lWwXc9jBGF1ZGUta292aS5iaWx1bWluYS5jb20iLCJyb3V0ZSI6bnVsbH0sIj9
mbGFZaCj6eyJvbGQiOltldLCJuZCj6XCiOltldX0=, "last_activity" = 1783424282, "user_id" = ?, "ip_address" =
[REDACTED], "user_agent" = Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/150.0.0.0 Safari/537.36, "id" = rhBDlLnCrwxKCF3ifsnKG9COykNaAdTEBuz62pDc where "id" =
rhBDlLnCrwxKCF3ifsnKG9COykNaAdTEBuz62pDc)
```

Ista vrsta težave, tokrat SQLite: seje se shranjujejo v [database/database.sqlite](#), www-data pa nima pisalne pravice niti na datoteko niti na mapo [database/](#) (SQLite rabi pisati tudi journal/WAL v mapo). Preverim in popravim:

Fixed sqlite db + dir group ownership/perms for www-data >

Preverim, da `www-data` zdaj lahko piše v bazo:

Verified www-data can write to sqlite db ›

Rešeno. Isti vzrok kot prej, le drug vir: se je so shranjene v SQLite (SESSION_DRIVER=database), www-data pa ni imel pisalne pravice ne na database/database.sqlite ne na mapo database/ (SQLite pri pisanju ustvarja začasne journal/WAL datoteke v isti mapi).

Popravek

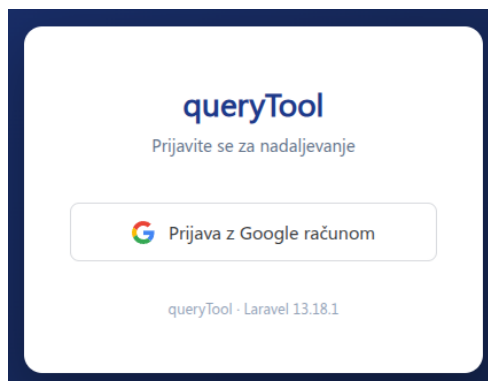
- `database/` → skupina `www-data`, `group-write` + `setgid`.
- `database/database.sqlite` → skupina `www-data`, `group-write`.
- Preverjeno: `www-data` uspešno naredi `INSERT/DROP` v bazi. ✓

Osveži stran — pisanje seje bo zdaj delovalo.

Skupni vzorec vseh treh napak je isti: projekt je last [REDACTED], servira pa ga nginx/php-fpm kot `www-data`. Če se pojavi še kje (npr. `storage/app`, nalaganje datotek), je rešitev enaka. Za trajno urejenost predlagam, da vse pisljive poti stalno pripadajo skupini `www-data` — to smo zdaj naredili za `storage/`, `bootstrap/cache/` in `database/`, kar pokriva običajne primere.

Slika 4: Druga napaka in rešitev

S to rešitvijo nas pričaka spletna stran.

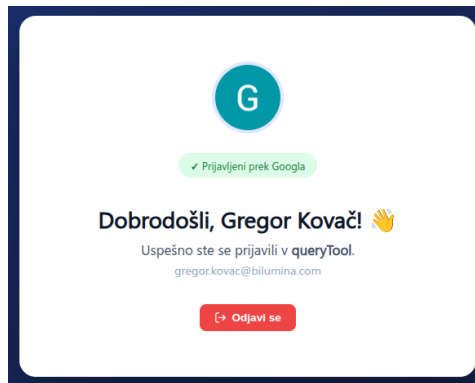


Slika 5: Prijavno okno

Ko po Claudovih navodilih pridobimo Google Client ID in skrivnost:

Nastavil sem GOOGLE_CLIENT_ID in GOOGLE_CLIENT_SECRET spremenljivke in ko se prijavim dobim 404. Naredi en welcome screen, dodaj tudi Odjavi gumb

Prijava in odjava delujeta:



Slika 6: Po prijavi

Vzpostavi dostop do baze in kreiraj stran za izvajanje poizvedb, samo SELECT-ov.

Gotovo. Povzetek.

Dostop do baze — vzpostavljen in preverjen

- `ibm_db2` je bil že nameščen (v2.3.1) — nič za namestit.
- DB2 okolje za `php-fpm` je pripravljeno prek `systemd drop-in` `db2.conf` (`clear_env=no`). Tekoči master ga ni imel → naredil `daemon-reload` + `restart php-fpm`.
- Potrjeno skozi `fpm socket`: `DB2INSTANCE=db2inst1` prisoten, `db2_connect` doseže strežnik `127.0.0.1:50000` (gonilnik + mreža delujeta). Web pot ima poln DB2 dostop.

Stran za poizvedbe (samo SELECT)

- `/query` (za prijavitelje) — `query.blade.php`: textarea, tabela rezultatov, čas izvedbe, opozorilo o skrajšanju.
- `QueryController.php` + poti v `web.php`.
- `Db2Connection.php` — native `ibm_db2`, DSN-less, `autocommit OFF` (nič se ne zapiše), meja `DB2_MAX_ROWS` (1000).
- `SelectOnlyValidator.php` — dovoli le en `SELECT/WITH/VALUES`, blokira DML/DDI, stacked queries in DB2 `FINAL/OLD/NEW TABLE(INSERT...)`. Vseh 8 testov gre skozi (vključno robni `created_orders`, ki ni napačno blokiran).
- `php artisan db2:ping` za test povezave.
- Povezava „SQL poizvedbe“ dodana na welcome screen in v navigacijo.

Slika 7: Vzpostavitev povezave na IBM Db2 bazo

Kar moraš narediti ti

Vpiši dostop v `.env` (ključi so že pripravljeni, prazni):

```
DB2_DATABASE=LUMINA
DB2_USERNAME=<bazni user>
DB2_PASSWORD=<geslo>
```

Nato:

```
php artisan config:clear
source /home/db2inst1/sqllib/db2profile && php artisan db2:ping
```

`db2:ping` zdaj pravilno javi „ni konfigurirano“ (dokaz, da je ožičenje pravilno — poveže se takoj ko vpišeš `credentials`).

Varnostno priporočilo: uporabi DB2 uporabnika samo z pravico branja (SELECT). Validator + `autocommit OFF` sta prva obramba, `read-only user` pa zanesljiva — poizvedbena stran je javno dostopna vsem, ki se prijavijo prek Google.

Ko vpišeš `credentials` in `db2:ping` uspe, lahko skupaj preverim še dejansko poizvedbo skozi spletni vmesnik.

Slika 8: Dopolnitev poverilnic

Sem jih vpisal

Credentials vpisani. Počistim config cache in poženem db2:ping:

Cleared config and ran db2:ping with real creds >

CLI povezava deluje (147 ms). Zdaj preverim še spletno pot (php-fpm z DB2 okoljem) z realno poizvedbo skozi aplikacijske razrede — sonda skozi fpm socket:

Tested real SELECT through php-fpm using app classes >

Vse deluje — potrjeno skozi spletno pot (php-fpm), z realnimi podatki:

- Povezava dela: db2:ping ✓ (147 ms, CLI).
- Realna poizvedba skozi aplikacijske razrede prek fpm socketja vrne prave podatke iz SYSCAT.TABLES:
 - CODE_ITEM — 214 tabel
 - TCODE_ITEM — 214
 - CHECK_CODE_ITEM — 213
- SELECT-only varovalka blokira DELETE ✓
- Sonda odstranjena, config cache počiščen.

Zaključek

Db2 dostop je vzpostavljen in stran /query je pripravljena za produkcijo. Odpri v brskalniku:

1. Prijavi se prek Googla → welcome screen.
2. Klikni „SQL poizvedbe (DB2)“ (ali navigacija → Poizvedbe).
3. Vpiši SELECT in izvedi. Rezultat je omejen na 1000 vrstic; ne-SELECT poizvedbe so zavrnjene, autocommit je izklopljen.

Ker se je povezala prek uporabnika lumina, opozorilo velja še naprej: preveri, da ima ta uporabnik na bazi LUMINA samo pravico branja — poizvedbena stran je dostopna vsem, ki se prijavijo prek Googla, in čeprav app blokira pisanje, je read-only DB user zanesljiva zadnja obramba.

Če želiš, lahko dodam še izvoz rezultatov v CSV ali zgodovino zadnjih poizvedb — povej.

Slika 9: Test povezave na bazo uspešen

SQL poizvedbe (DB2)

Dovoljene so samo SELECT poizvedbe. En stavek naenkrat.

```
select item_id, item_code from tcode_item.item limit 3
```

samo za branje Izvedi ▶

3 vrstic · 80 ms

ITEM_ID	ITEM_CODE
5	P0001
6	P0002
7	P0003

Slika 10: Uspešno izveden SELECT

Zdaj potrebujemo še orodje za analizo SQL stavkov:

```
zdej bomo pa naredil query tool - za delanje sqlov.
tabele so v shemi logic in sicer action , actionquery,
actionqueryfield in actionqueryparameter ..
analiziraj te tabele kaj so in kaj vsebinsko in mi
potem naredi query tool
```

Po nekaj iteracijah s Claudom:

Vse deluje in je preverjeno. Query tool je narejen.

Analiza tabel (kaj so)

Gre za metapodatkovni register parametriziranih poizvedb aplikacijskega ogrodja:

```
LOGIC.ACTION (9.593) → LOGIC.ACTIONQUERY (6.831) → ACTIONQUERYFIELD (76.505) [izhodni stolpci]
akcija          SQL (CLOB) + :params          ↳ ACTIONQUERYPARAMETER (16.969) [vhodni :parametri]
```

- ACTIONQUERY.AQUE_SQL = dejanski SQL z imenskimi parametri (:GLOS_CODE), pogosto WITH UR; lahko je SELECT ali CALL proc(...).
- FIELD = izhodna polja (ime+tip), PARAMETER = vhodni parametri (ime+tip), oba se ujemata z SQL-jem.

Orodje (kot si želel: vpišeš SQL → dobiš parametre s tipi + izhodne stolpce s tipi)

Nova stran /analyze ([analyze.blade.php](#)):

- Vpišeš SQL z :parametri → dve tabeli: Parametri (ime + tip + izvor) in Izhodni stolpci (ime + tip).
- Poizvedbe NE izvede — uporablja DB2 DESCRIBE (samo prepare).

Kako pridobi tipe ([Db2Describe.php](#)):

- Izhodni stolpci: DB2 CLP DESCRIBE OUTPUT → točni SQL tipi (VARCHAR(257), BIGINT, DATE, CLOB...). Parser obvlada osnovno in razširjeno (LOB) obliko.
- Parametri:
 - CALL → tipi iz signature procedure (SYSCAT.ROUTINEPARMS), pozicijsko.
 - SELECT → tip primerjanega stolpca (SYSCAT.COLUMNS, prek FROM/JOIN aliasov); prikaže tudi izvorni stolpec.
 - kjer ni mogoče → neznano (jasno označeno).

Preverjeno (skozi php-fpm kot www-data, na realnih podatkih)

- SELECT: :GLOS_CODE → VARCHAR(257), :LANG_ID → BIGINT; izhod vklj. GLOS_DESCRIPTIONLARGE → CLOB ✓
- CALL IMPORT.TEMPLATE_ITEM01: 61 parametrov pravilno preslikanih na proc signaturo ✓
- Robni primeri (CLOB razširjena oblika, več tabel) ✓

Povezave dodane na dashboard in v navigacijo (Domov · Poizvedbe · Analiza). Odpri /analyze po prijavi in vpiši SQL.

Opomba: /analyze bere le metapodatke (DESCRIBE + SYSCAT), ničesar ne izvede in ne piše. Če želiš, lahko dodam še: (a) gumb „naloži iz registra“ (izbereš ACTIONQUERY po ACTI_CODE in ti napolni SQL), ali (b) primerjavo dobljenih tipov z deklariranimi v ACTIONQUERYFIELD/ACTIONQUERYPARAMETER.

Slika 11: Implementacija QueryTool-a

Analiza SQL — parametri in izhodni stolpci

Vpiši SQL z imenskimi parametri (npr. :GLOS_CODE). Orodje prikaže **tip** parametrov in **tip** izhodnih stolpcev — brez izvajanja poizvedbe (DB2 DESCRIBE).

```
SELECT ITEM.ITEM_CODE, ICOD.ICOD_CODE
FROM TCODE_ITEM.ITEM ITEM
INNER JOIN TCODE_ITEM.CODE ICOD ON ICOD.ITEM_ID = ITEM.ITEM_ID
WHERE ITEM.ITEM_CODE = :ITEM AND ITEM.ITEM_ACTIVE = :IS_ACTIVE
```

SELECT / WITH / VALUES ali CALL

Analiziraj ►

SELECT 2 parametrov · 2 izhodnih stolpcev

Parametri (vhod)

Imenski :parametri in njihovi tipi

Parameter	Tip	Vir
:ITEM	VARCHAR(30)	ITEM.ITEM_CODE
:IS_ACTIVE	SMALLINT	ITEM.ITEM_ACTIVE

Izhodni stolpci

Stolpci rezultata in njihovi tipi

Stolpec	Tip
ITEM_CODE	VARCHAR(30)
ICOD_CODE	VARCHAR(257)

Slika 12: Test SELECT-a

Preizkus procedure (CALL TEREN.OTS_2026(:NORDE_ID)) vrne seznam vhodnih parametrov, ne pa tudi informacij o vrnjenih podatkih (lahko je en nabor rezultatov, lahko jih je več), kar povemo tudi Claudu; rešitev najde, ko mu pokažemo še vzorec, po katerem so napisane naše procedure.

Poglej kako je narejena procedura `WD_FORM.SET_DOCUMENT` in poglej kateri sql je v katerem result setu. Podobno so narejeno tudi druge naše procedure.

Preberem celo telo `WD_FORM.SET_DOCUMENT` in izluščim: katere `WITH RETURN` kurzorje ima, na kateri `STATEMENT` je vsak vezan, in od kod (`SET/PREPARE`) dobi SQL. Tako razberem vzorec:

Created a file, ran a command >

`SET XSQL = 'SELECT ...'` — pri tej proceduri je SQL literal (ne kot pri `TEMPLATE_ITEM01`). To je ključno. Izvlečem celoten XSQL (od `SET XSQL` do `PREPARE`) in preverim pravo dolžino telesa:

Extracted full XSQL construction in `SET_DOCUMENT` >

Zdaj razumem vzorec. `WD_FORM.SET_DOCUMENT` (telo 6789 znakov):

```
SET XSQL = 'SELECT ... FROM ... WITH UR';    ← cel SELECT je LITERAL string
PREPARE XSQLSTATE_3 FROM XSQL;               ← pripravi iz spremenljivke XSQL
DECLARE XRESULT_SET_3 CURSOR WITH RETURN TO CLIENT FOR XSQLSTATE_3; ← 1 result set
```

Ključno: SQL je statično v telesu, le da gre prek `SET XSQL = 'literal' → PREPARE ... FROM XSQL` (spremenljivka), ne neposredno iz literala. Moja koda je iskala le literal neposredno v `PREPARE` — zato ni ujela tega vzorca. To je popravljivo (sledim spremenljivki do njenega SET).

Najprej konkretno odgovorim: opišem ta `SELECT` (result set #1):

Described `SET_DOCUMENT` result set `SELECT` >

Popolno — result set #1 (`XRESULT_SET_3`) vrne 14 stolpcev z natančnimi tipi (`VARCHAR(30)`, `TIMESTAMP`, `BIGINT` ...).

Zdaj razumem vaš splošni vzorec: `SET XSQL = 'literal SELECT' → PREPARE stmt FROM XSQL → DECLARE cursor WITH RETURN FOR stmt`. Nadgradim orodje, da sledi spremenljivki do `SET` literala in prikaže stolpce po posameznem result-setu. To pokrije `SET_DOCUMENT` in podobne procedure.

Najprej prenovim `CALL` logiko v `Db2Describe` (result-seti namesto enega seznama):

Edited `Db2Describe.php` +21-15 >

Zdaj zamenjam `procedureResultColumns` z `procedureResultSets`, ki sledi vzorcu `SET var = 'literal' → PREPARE → kurzor`:

Edited `Db2Describe.php` +91-42 >

Zdaj testiram na `SET_DOCUMENT` (1 result set) in `GET_ACTIONSAVE` (5 result setov) prek CLI:

Tested result set extraction on `SET_DOCUMENT` and `GET_ACTIONSAVE` >

`SET_DOCUMENT`: 1 result set, 14 stolpcev ✓. Ampak dva buga vidim: (1) pri `GET_ACTIONSAVE` regex prekomerno požre in najde le 1,3,5 namesto vseh 5 kurzorjev; (2) enkratni dinamični result set (npr. `TEMPLATE_ITEM01`) zdaj ne pade več na register (regresija). Popravim oboje — regex z offset-om in dopolnitev iz registra za enkratni dinamični result set:

Edited a file, ran a command >

Odlično, oba buga popravljena:

- `SET_DOCUMENT`: 1 result set, 14 stolpcev (iz telesa) ✓
- `GET_ACTIONSAVE`: vseh 5 result setov najdenih (dinamični → poštene opombe) ✓
- `TEMPLATE_ITEM01`: 1 result set, 154 stolpcev — izkazalo se je, da tudi ta uporablja `SET XSQL = 'literal'` (moja prejšnja ocena „dinamičen“ je bila napačna; zdaj ga orodje pravilno prebere iz telesa).

Zdaj posodobim UI, da prikaže stolpce po posameznem result-setu (in pripadajoči `SELECT`). Prenovim izhodni del `analyze.blade.php`:

Slika 13: Popravki za procedure

Analiza SQL — parametri in izhodni stolpci

Vpiši SQL z imenskimi parametri (npr. :GLOS_CODE). Orodje prikaže **tipe parametrov** in **tipe izhodnih stolpcev** — brez izvajanja poizvedbe (DB2 DESCRIBE).

```
CALL TERE.N.OTS_2026( :NORDE_ID )
```

SELECT / WITH / VALUES ali CALL

Analiziraj ►

CALL 1 parametrov · 2 result-set(ov)

⚠ Izhodni stolpci so razbrani iz telesa procedure (WITH RETURN kurzorji).

Parametri (vhod)

Imenski :parametri in njihovi tipi

Parameter	Tip	Vir
:NORDE_ID	BIGINT	NORDE_ID

Result set 1 (XRESULT_SET_1)

► Prikaži SELECT (3 stolpcev)

Stolpec	Tip
ORDE_ID	BIGINT
ITEM_ID	BIGINT
OITE_QUANTITY	DECIMAL(23,7)

Result set 2 (XRESULT_SET_2)

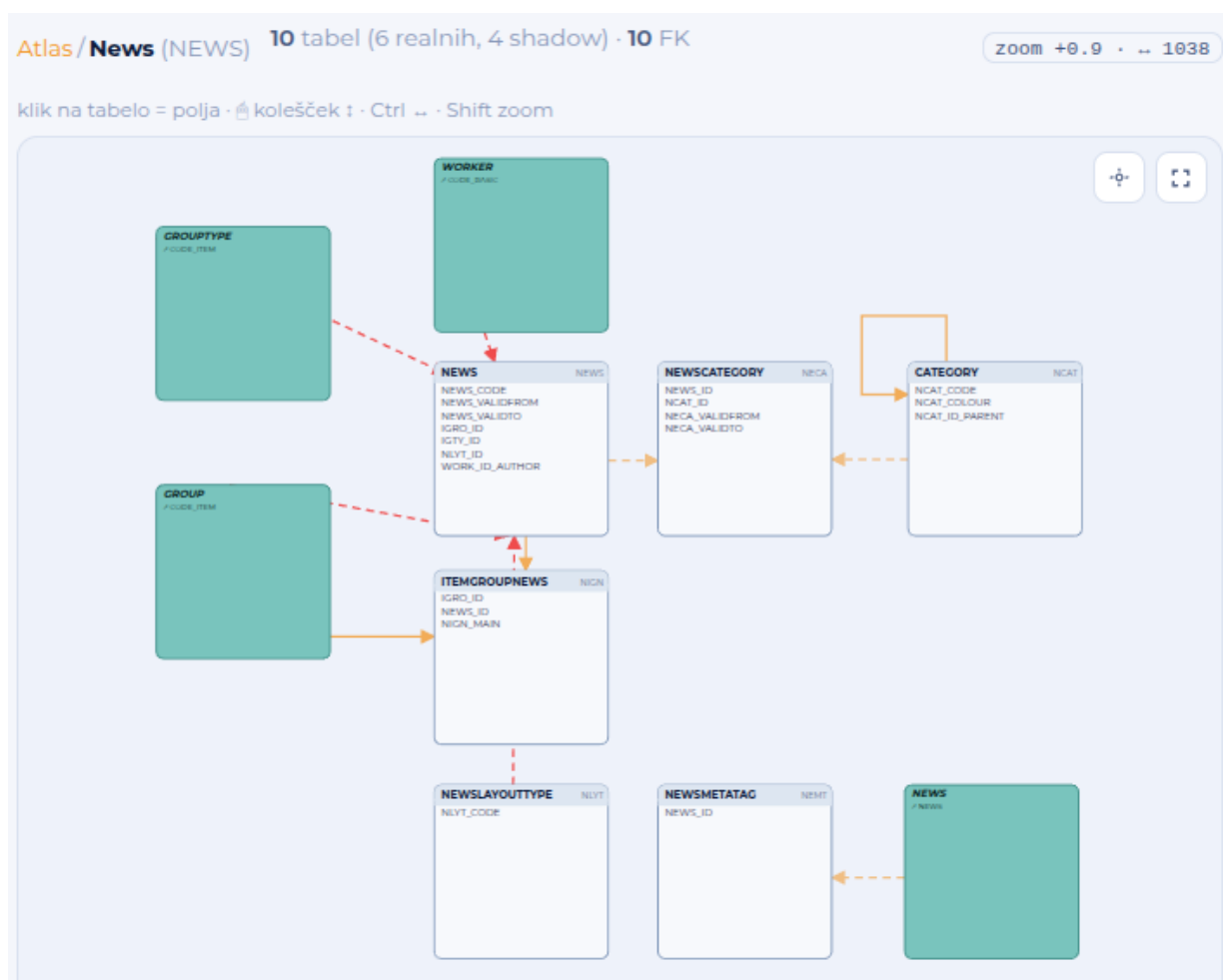
► Prikaži SELECT (3 stolpcev)

Stolpec	Tip
ORDE_ID	BIGINT
ITEM_ID	BIGINT
OITE_QUANTITY	DECIMAL(23,7)

Slika 14: Test procedure

2.4. Naslednji koraki

S pomočjo umetne inteligence smo v kratkem času (približno mesec dni) na ogrodje Laravel prenesli še nekaj drugih internih orodij, kot so Linea, monitoring baz in helpdesk.



Slika 15: »Nova« Linea

Monitor — Prikaz			
4 1 16 11			
STRANKA	METRIKA	VREDNOST	PRAG
	FORCE_PROCEDURE	2	1
	REPLICATION_CHECK_SUM_ALL_ERROR	1	1
	SCAN_WD_AVG	229	120
	STAT_2_CENT	2	1
	SCAN_AVG	306	250 700
	CONNECTION_EXECUTE	8	10 20
	CONNECTION_LOCK	0	5
	CONNECTION_TOO_MUCH_1	0	1
	CONNECTION_TOO_MUCH_2	0	1
	HCH	0	1
		0	1
	NO_DATA_CONTROL	0	1
	PRINT_AVG	182	300 700
	PRINT_MAX	1063	2000 5000
	PRINT_TOO_MUCH	0	5 10
	SCAN_MAX	1668	10.000 15.000

Slika 16: Spremljanje podatkovne baze

Helpdesk

Seznam

Tiketi v mojih procesih

Zadnja osvežitev: 08.07.2026 11:59 (pred 50s)

55 tiketov · 51 · prepis 4 · skrito 180

ODPRT

V ODELANI

REŠENO

ARHIVIRANO

Osveževanje: 30 s

Osveži zdaj

osveženo - 12:00:23

#1165

Manjkajoča evidenca prihoda

Uporabnik nima vpisane evidence prihoda za devet preteklih delovnih dni in jo mora dopolniti ali prijaviti odsotnost.

N

Portal (helpdesk portal)

#1145

Zapiranje razknjiževanja surovin na DN v minus na lokacijah Centralnega skladišča

Zapiranje minusa artikla 0000628 čokolada bela nova (-194 kg) na skladišču 100 LOK 0013 iz 6.7.2026.

A

Mrežna razpoložljivost (koskladišča)

#1142

ID 6841369

Artikel 0003989 se na izpisu prikazuje napačno - opis se ne izpisuje kot pri običajnih členih, kar onemogoča pošiljanje računov.

A

Atributi (dinamični)

#1141

Podatki za nadaljevanje nadgradnje maske za poročanje embalaže, maska

Dopolnjeni so manjkajoči podatki za nadgradnjo maske za poročanje embalaže: artikli, nove skupine VEMB, vrste embalaže in teže.

07.07.2026 08:28

Izpis prodane embalaže (embalažnina)

#1133

RE: [] Naročilo #3213 od []

Iz predračuna 2380481 se ni general račun in ga Tanja ne more narediti sama. Prosi za ureditev.

A

Spletna odprema (fulfillment / dispatch)

Slika 17: Helpdesk

Pri helpdesku rešitev bere elektronska sporočila, določi, v kateri poslovni proces sodijo, odpre zahtevek za ustrezno osebo, med prejšnjimi rešitvami poišče najustreznejšo ter predlaga odgovor stranki. Pri razvoju sistema za pomoč uporabnikom (helpdesk) smo ugotovili, da potrebuje jasno strukturiran spomin – zbirko rešitev, primerov odgovorov na elektronska sporočila in podobnih smernic. Šele v takšnih jasno začrtanih okvirov deluje zanesljivo; brez njih namreč začne odgovore preprosto ustvarjati sama.

Umetna inteligenca nam je z uporabo orodja Ansible pomagala zgraditi tudi rešitev za samodejno postavitve strežnikov različnih tipov (bazni, aplikacijski, spletni) ter njihovo nastavitve (dostopi do baz, nameščanje aplikacij na aplikacijske strežnike itd.)).

3 Širši pogled: zmožnosti in meje umetne inteligence pri razvoju

Naši izkušnji nista osamljeni. Postavljamo ju v širši okvir razprave o vlogi umetne inteligence pri razvoju programske opreme, in sicer z dveh zornih kotov: inženirskega (kaj orodja danes zmorejo in česa ne) in etičnega (kako naj to tehnologijo odgovorno uporabljamo).

3.1. Inženirski pogled: kje pomaga in kje ne

Koristno izhodišče je razumevanje razvoja programske opreme kot sklopa treh plasti: odločanja (kaj zgraditi), izvedbe (kako to zapisati v kodo) in dostave (kako to preoblikovati v zanesljivo delujoč izdelek). Umetna inteligenca najizraziteje pospešuje srednjo, izvedbeno plast, medtem ko se odločanje in dostava avtomatizaciji izmikata [2]. To pojasnjuje, zakaj kljub hitremu napredku zmogljivosti ni prišlo do napovedanih množičnih odpuščanj razvijalcev; številni odmevni primeri, ki so jih mediji pripisali umetni inteligenci, so bili v resnici posledica drugih dejavnikov, predvsem finančnih pritiskov [2]. V nekaterih primerih [12] so podjetja celo znova zaposlila inženirje, saj kakovost rešitev, ki jih je zagotavljala umetna inteligenca, ni dosegala zahtevanih standardov. Enako se je pokazalo v obeh naših primerih: umetna inteligenca je bila najbolj koristna pri izvedbi, torej pri pisanju kode, medtem ko smo odločitve o zasnovi in merila za sprejemljivost rezultata obdržali v lastnih rokah.

Zmogljivosti teh orodij ob tem hitro naraščajo. Podjetje Anthropic poroča, da razvoj umetne inteligence vse bolj poganja umetna inteligenca sama; njihovi inženirji danes v povprečju oddajo približno osemkrat več kode na četrtletje kot v obdobju 2021–2025, v skrajni obliki pa se tak trend približuje t. i. rekurzivnemu samoizboljševanju [9]. Prav zato postajajo večšine, ki niso neposredno pisanje kode – razumevanje pravega problema, zasnova rešitve, presoja in dostava – za razvijalca še pomembnejše, ne manj [10].

Ena pomembnejših praktičnih omejitev je t. i. razpadanje konteksta (context rot): kakovost odgovorov modela tiho upada, ko se kontekstno okno polni, saj je pozornost modela omejen vir, ki se z več žetoni razprši [3]. Napaka se ne pokaže kot izjema ali na nadzornih ploščah, temveč kot postopno izgubljanje natančnosti.

Zavedati se velja tudi, da konteksti, navodila in datoteke za usmerjanje agentov predstavljajo svojevrstno vrsto tehničnega dolga. Za razliko od kode, ki ostaja stabilna, dokler se je ne dotikamo, lahko vsaka nova različica modela tiho razvrednoti prej skrbno izdelana navodila [4]. Navodila naj zato temeljijo na konkretnih in preverljivih dejstvih o projektu, izogibamo pa se splošnemu »usmerjanju vedenja« modela.

Spodbudna je ugotovitev, da koda, ki je dobro vzdrževana in razumljiva za človeka, hkrati ostaja bolj zanesljiva tudi za urejanje z umetno inteligenco. Raziskava na 5.000 datotekah je pokazala pomembno povezavo med metriko zdravja kode (CodeHealth) in ohranitvijo pomena po preoblikovanju z jezikovnim modelom [5]; organizacije lahko take metrike uporabijo za presojo, kje je poseg umetne inteligence manj tvegan in kje je potreben dodaten človeški nadzor. Hkrati pa agentno programiranje zahteva več, ne manj discipline – več strukture, jasnejša merila kakovosti in nadzorne mehanizme [6].

Nazadnje ne gre prezreti, da jezikovni modeli ostajajo nezanesljivi na načine, ki niso vedno očitni: slabo obvladujejo zanikanja [7], vrhunski modeli pa si pri preverjanju dejstev pogosto nasprotujejo. Za nas to pomeni, da preverjanje rezultatov proti obstoječi, dokazano pravilni implementaciji ni neobvezen dodatek, temveč nujen del postopka.

Ne smemo pa spregledati stroškov umetne inteligence. Vlagatelji si bodo namreč na dolgi rok želeli povrniti stotine milijard dolarjev, ki jih trenutno vlagajo v podjetja, ki (še) ne poslujejo z dobičkom. Stroške lahko znižamo z naslednjimi ukrepi:

- **Uravnavanje vloženega napora (angl. effort).** Manjši vložen napor lahko prinese boljše razmerje med kakovostjo in ceno; model Fable 5 pri nastavitvi low doseže boljši rezultat (60 % pri 3,76 USD) kot Opus 4.8 pri nastavitvi max (59 % pri 13 USD) [13],
- **Prilagajanje modela zahtevnosti naloge.** Za vse naloge ni treba uporabljati najzmogljivejšega modela. Fable 5 lahko uporabimo za arhitekturo in načrtovanje, samo izvedbo pa glede na njeno zahtevnost prepustimo cenejšim modelom (Opus, Sonnet ali celo kateremu od modelov GPT),
- **Zmanjševanje porabe žetonov (angl. tokens).** Z uporabo namenskih veščin (angl. skills) za zmanjšanje porabe žetonov, kot sta Ponytail [14] in Caveman [15], dosežemo enak rezultat ob manjši porabi.

3.2. Etični pogled: enciklika Magnifica Humanitas

Razprava o umetni inteligenci ni le tehnična, temveč tudi etična. Dober okvir ponuja prva enciklika papeža Leona XIV. z naslovom Magnifica Humanitas: o varovanju človekove osebe v času umetne inteligence, objavljena 25. maja 2026 ob 135-letnici enciklike Rerum Novarum [8]. Enciklika tehnologije ne obravnava kot sovražne ali same po sebi zlonamerne, opozarja pa, da »tehnologija nikoli ni nevtralna, saj prevzame značilnosti tistih, ki jo zasnujejo, financirajo, urejajo in uporabljajo«. Poziva, naj umetna inteligenca služi človeku in skupnemu dobremu ter naj ne koncentrira moči; med drugim izpostavlja dostojanstvo dela in poziv, naj »ostanemo človeški« [8].

Za razvijalce iz tega sledi konkretna drža. Umetna inteligenca naj bo orodje, ki razvojno ekipo razbremeni rutinskih opravil in ji omogoči osredotočenost na zahtevnejše, ustvarjalne naloge – ne pa nadomestek za človeško presojo in odgovornost. Ta drža se ujema z našo osrednjo ugotovitvijo: pri obeh primerih je vodilno vlogo – odločanje, presojo in končno odgovornost za rezultat – ohranil razvijalec.

Ta drža postaja še pomembnejša, ker razvoj umetne inteligence vse bolj poganja umetna inteligenca sama, s čimer se povečuje tudi tveganje izgube človeškega nadzora [9]. Toliko bolj velja poziv enciklike, naj človek ostane v središču odločanja in odgovornosti.

4 Zaključek in nadaljnji koraki

Naša izkušnja kaže, da je umetna inteligenca lahko resnična pomoč pri razvoju in vzdrževanju internih orodij, če nalogo dobro omejimo, jo razdelimo na preverljive enote in rezultate dosledno preverjamo. Pri implementaciji internih orodij je bila najbolj produktivna pri izvedbi, medtem ko sta zasnova in nadzor ostala v rokah razvijalca.

Ker se je pilotni prepis QueryTool-a izkazal za uspešnega, smo s pomočjo umetne inteligence na podoben način na sodobnejšo tehnološko osnovo prenesli tudi nekatera druga interna orodja. V prihodnje načrtujemo širšo uporabo umetne inteligence, in sicer za samostojno programiranje in podporo ter za njeno vpeljavo, po potrebi tudi za nadzor, neposredno v procese sistema ERP. Prispevek tako predstavlja prvi korak v širšem premisleku o tem, kako naj se uveljavljene interne razvojne prakse soočijo z orodji, ki postajajo iz dneva v dan zmogljivejša.

Literatura

- [1] BIZJAK Tomaž, KOVAČ Gregor, KOZLAR Tomaž: Implementacija prilagodljivega ERP sistema z malo ali nič kode. Sodobne informacijske tehnologije in storitve OTS 2025: Zbornik 28. konference, Maribor, 2025.
- [2] NARAYANAN Arvind, KAPOOR Sayash: Why AI hasn't replaced software engineers, and won't. normaltech.ai, 11. 6. 2026.
- [3] BHATKOTI Vineet: Understanding Context Rot - Why Your AI Agent Gets Worse the Longer It Works. DZone, 18. 6. 2026.
- [4] GOEDECKE Sean: Prompts are technical debt too. seangoedecke.com, 2026.
- [5] BORG Markus, HAGATULAH Nadim, TORNHILL Adam, SÖDERBERG Emma: Code for Machines, Not Just Humans: Quantifying AI-Friendliness with Code Health Metrics. arXiv:2601.02200, 2026.
- [6] TORNHILL Adam: Agentic AI Coding Practices for Speed with Quality (predstavitev), 2026.
- [7] MAYNE Harry, MCKINNEY Lev in dr.: Negation Neglect: When models fail to learn negations in training. arXiv:2605.13829, 2026.
- [8] LEON XIV.: Magnifica Humanitas - enciklika o varovanju človekove osebe v času umetne inteligence. Vatikan, 15. 5. 2026.
- [9] ANTHROPIC INSTITUTE: When AI builds itself. anthropic.com, 2026.
- [10] ARJANCODER: AI Writes Code Faster Than You. Now What? (video), 2026.
- [11] ANTHROPIC: Models overview. platform.claude.com/docs/en/about-claude/models/overview
- [12] BBC: Ford rehires human engineers after AI fails to match quality checks, <https://www.bbc.com/news/articles/cgrkd41n2v9o>
- [13] <https://deepswe.datacurve.ai>
- [14] <https://ponytail.dev/>
- [15] <https://caveman.so/>

