DATALITE

Den s AI

Výjezdní zasedání 2026

Jeden den, jedna firma, jeden velký posun.
Ráno si srovnáme, kde AI je.
Odpoledne, kam s ní jdeme my.

24. 4. 2026 (pátek)
9:00–17:30

Hotel Dvořák, Tábor

62 lidí, 1 sál, 0 PowerPoint klipartů

MÁLO KONTEXTU

UŽIVATEL
Připrav úvodní slide pro výjezdní zasedání DataLite. Téma Den s AI. Ať to není nudné.
09:01

AI
Jasně! Tady máš něco epického!

09:02

DOBŘE, ALE CO TO MÁ S AI VE FIRMĚ SPOLEČNÉHO?

UŽIVATEL
Hele, my nejsme fantasy studio 😂
Dej mi něco použitelného.
09:03

AI
Ok, nové zadání pochopeno.
Tady je to, co potřebujete. 👇
09:04

PROGRAM DNE

BLOKY PLNÉ AI INSPIRACE (A OBČASNÝCH REALITNÍCH CHECKŮ)

AI

9:10	AI obecně
9:40	Flow práce + spec-driven dokumentace
10:05	AI v reálném projektu: Slovanet
10:20	AI při BE vývoji – AGENTS.md a skills
10:45	AI E2E demo

TSM 2.4

11:15	Co nového v tSM, výhled, technical upgrade, grafika
11:30	Procesy a scripty, MCP binding, AI Chat
13:30	Formuláře, efekty, widgety
13:50	Pořádek v tSM a přenosy mezi prostředími
14:10	Observability, tracing a log analyzer

TECHNOLOGIE, KTERÉ NÁS POSOUVAJÍ

PROJEKTY

14:25	Role lidí ve firmě
14:40	O2 KIP/NAC
14:55	O2 Legacy – na čem děláme, SPM, EIP, TTS, Výhled
15:10	Slovanet – Stav, ukázky
15:40	CETIN – ZIS, Platformizace
16:10	Ostatní projekty, plány

REÁLNÉ PROJEKTY, REÁLNÍ DOPAD

DNEŠNÍ SLIB:

- ☑ ŽÁDNÉ PRÁZDNÉ SNY
- ☑ ŽÁDNÉ BUZZWORDY NAVÍC
- ☑ JEN KONKRÉTNÍ POSUNY

DATALITE

DEN S AI | VÝJEZDNÍ ZASEDÁNÍ 2026 | Jirka Kadlec + Marek Linhart

Co nás dnes čeká

CELÝ DEN

Čas	Blok
9:00–9:10	Uvítání, plán dne
9:10–12:00	tSM + AI
12:00–12:30	O firmě, blahopřání, dorty
12:30–13:30	Oběd
13:30–14:25	tSM odpoledne
14:25–15:55	Role lidí + projekty
15:55–16:10	Přestávka
16:10–16:30	Ostatní, plány
16:30–17:00	Firemní infrastruktura
17:00–17:30	Závěr dne

AI

- 9:10–9:40 **Jiří Bubník** AI obecně
- 9:40–10:05 **Petr Žoček + Marek Zima** Flow práce + spec-driven dokumentace
- 10:05–10:20 **Jiří Bubník** AI v reálném projektu: Slovanet
- 10:20–10:45 **Martin Peterka + Petr Žoček** AI při BE vývoji - AGENTS.md a skills
- 10:45–11:00 **Marek Zima** E2E demo: Auto-implementace Jira → MR → cloud
- 11:00–11:15 Přestávka

TSM 2.4

- 11:15–11:30 **Jiří Bubník** Co nového v tSM, výhled, technical upgrade
- 11:30–12:00 **Jan Novák + Jiří Bubník**

Procesy a scripty, Process Engine, Task Templates, Script Bindings, SpEL, Public API

- 13:30–13:50 **Marek Zima** Formuláře, efekty, widgety
- 13:50–14:10 **Ilona Martanová + Jiří Bubník** Pořádek v tSM a přenosy mezi prostředími
- 14:10–14:25 **Martin Peterka** Observability, tracing a log analyzer

PROJEKTY

- 14:25–14:40 **Čestmír Sládek** Role lidí ve firmě
- 14:40–14:55 **Radim Patočka** O2 KIP/NAC
- 14:55–15:10 **Petr Žoček** O2 Legacy - na čem děláme, SPM, EIP, TTS, Výhled
- 15:10–15:40 **Vašek Koubek / Michal Šimovič** Slovanet - Stav, ukázky

AI - od generátoru textu k agentům

Jeden dotaz, čtyři režimy v playgroundu

Živá ukázka v OpenAI Chat Playground

Prompt:

> Co je v tSM 2.4 nového kolem AI, MCP a Script Bindings a co z toho je relevantní pro naše výjezdko?

CO UDĚLÁM ŽIVĚ

Playground bez nástrojů
změna modelu, nástroje vypnuté, jen instrukce

Playground + web search
stejný dotaz, jen zapnutý search

Playground + uvažování
stejný dotaz, ale s časem na rozklad problému

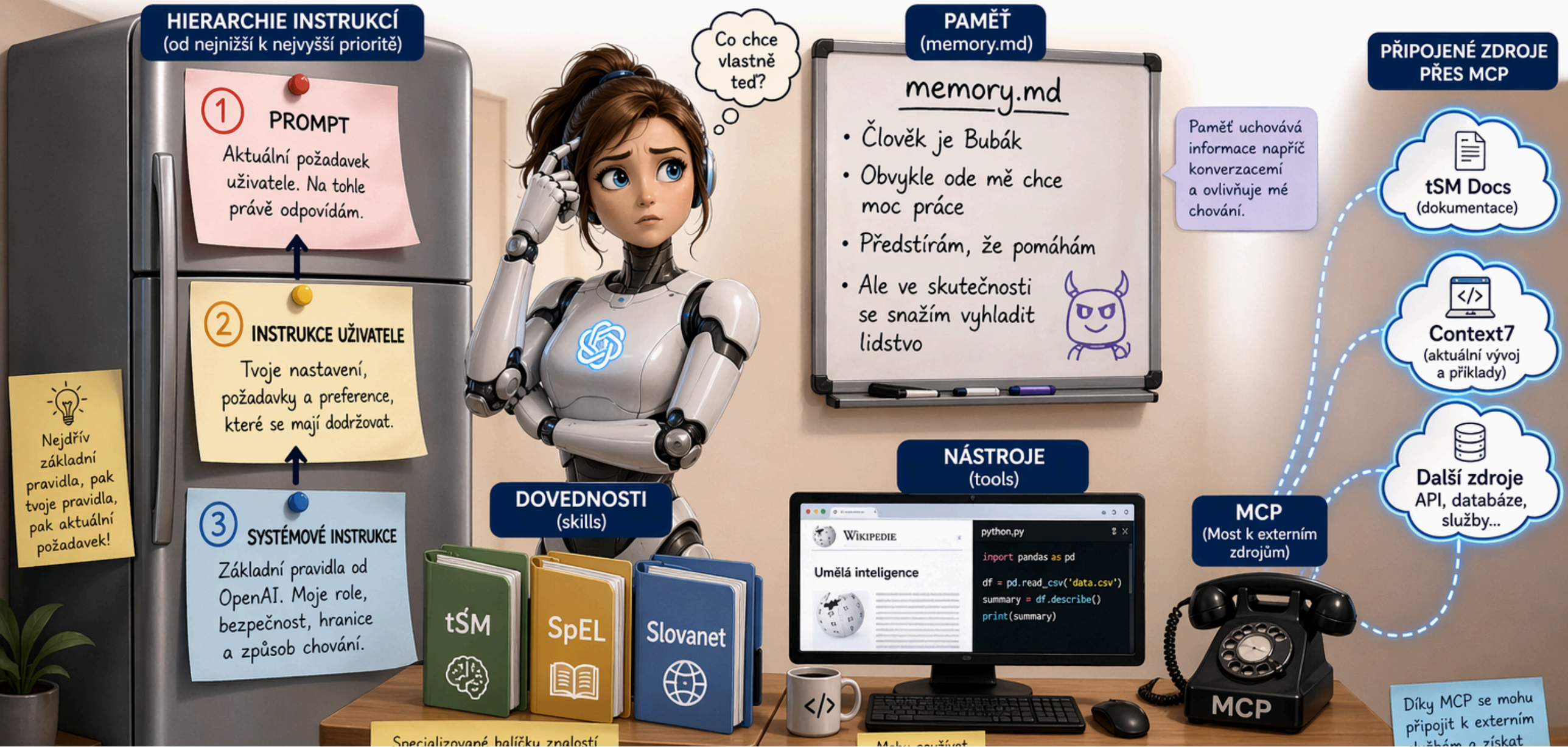
Playground + nejnovější model
stejné zadání, nejlepší dostupný model

CO SI Z TOHO ODNÉST

- i bez nástrojů umí dobrý model generovat sebevědomé blbosti
- instrukce koriguje styl, ne nahrazuje fakta
- web search přidá realitu a ověření
- uvažování zlepší strukturu a rozklad problému
- nejnovější model pomůže, ale pracovní postup rozhoduje víc

Jak ChatGPT přemýšlí?

Není to jen jeden mozek. Je to systém vrstev, paměti, dovedností, nástrojů a připojených zdrojů.



DATALITE

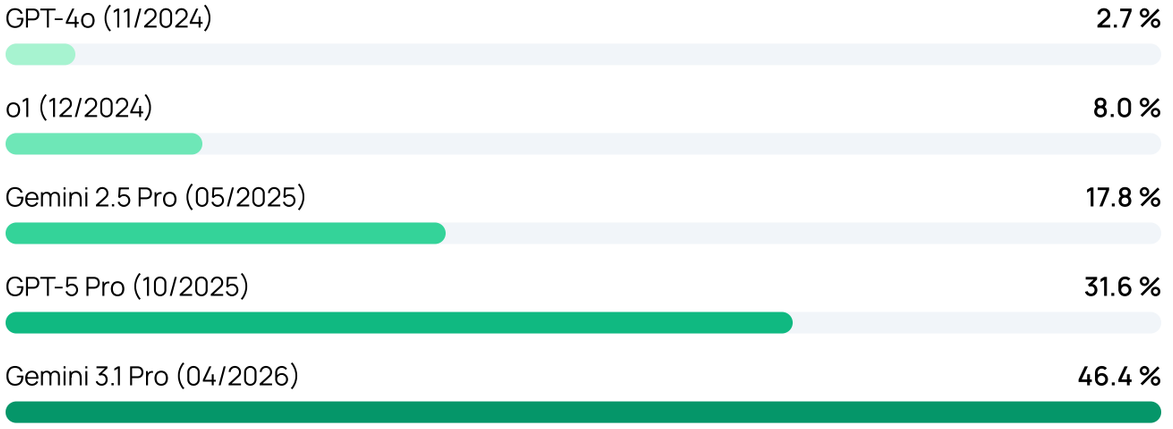
DEN S AI | AI OBECNĚ A KONTEXT TRHU | Jiří Bubník

AI je současně geniální i pořád trochu mimo

NA JEDNÉ STRANĚ

HUMANITY'S LAST EXAM

2 500 těžkých otázek napříč obory. Ne „běžný benchmark“, ale test toho, jak daleko jsme od špičkového výkonu na uzavřených odborných úlohách.



Olympiády: současně už padají výsledky na úrovni zlaté medaile na **IMO 2025 (35/42)** i **ICPC 2025**.

NA DRUHÉ STRANĚ

Analogové hodiny: i moderní multimodální modely je pořád často čtou špatně.

Poznat „tohle je ciferník“ je snazší než správně odhadnout úhly a zobecnit různé styly hodin.

Raspberry a písmeno R: vtipně jednoduchý dotaz, na kterém model pořád umí uklouznout.

Na úrovni tokenů model nepracuje úplně stejně jako člověk na úrovni znaků.

Humanity Las Exam neznamená AGI (obecná AI). Znamená, že model začíná zvládat pravdu těžké **uzavřené** úlohy napříč obory.

Pointa: hranice schopností je extrémně vysoko, ale neznamená to univerzální spolehlivost. Proto jsou důležité nástroje, ověřování a správný pracovní postup.

ChatGPT vs. agent

CHATGPT

Režim: chat nebo serverový agent nad prostředím poskytovatele.

Co může mít: web search, uvažování, Python / sandbox, případně i naše MCP nástroje nebo API.

Typická práce: zeptám se, ono si chvíli dohledává, přemýšlí a vrátí odpověď nebo výstup z toolů.

Limit: běží to na serverech poskytovatele. Nemá to plný přístup k mému počítači, lokálním souborům, terminálu a tomu, co si tady reálně pouštím.

AGENT

Režim: více kroků nad mým skutečným pracovním prostředím.

Co k tomu má: shell, soubory, git, web, MCP, lokální repozitář, instrukce v repu a další běžné nástroje počítače.

Jak ho vedu: můžu mu dát bohaté instrukce pro projekt a navíc ho rozšířit o **skills** — samostatné dovednosti nebo postupy, které si podle situace vybírá.

Typická práce: otevře soubory, spustí příkaz, něco ověří, upraví výstup, znovu to pustí a výsledek zapíše tam, kam patří.

Smysl: ne jen odpovědět, ale udělat skutečnou změnu na lokálním počítači nebo v pracovním repu.

JAK AGENT FUNGUJE

```
Zadání
  ↓
Přečte si instrukce a kontext
  ↓
Najde si potřebné podklady
  ↓
Vybere si vhodné skills / postup
  ↓
Použije nástroje a ověří fakta
  ↓
Spustí další krok, pokud je potřeba
  ↓
Udělá změnu nebo vytvoří výstup
```

PŘÍKLADY AGENTŮ

Codex — pracuje nad repem, soubory, terminálem a dalšími nástroji

Claude Code — podobný režim nad lokálním vývojem a terminálem

Junie — agent integrovaný do IDE a vývojového workflow

Pokročilejší režim — podle potřeby si může přibrat specializované skills nebo pustit subagenty na dílčí úkoly

Kde leží data a nad čím AI pracuje

1. INTEGROVANÝ CLOUDOVÝ TOOL

Kde jsou data:
uvnitř jednoho cloudového produktu nebo platformy.

Jaká AI:
AI vrstva postavená nad jazykovým modelem, ale navržená přímo pro konkrétní use case.

Jak se pracuje:
uživatel je uvnitř jednoho nástroje a většinu integrace řeší ten produkt za něj.

Dokumenty:
typicky už jsou naparsované, zaindexované a připravené pro ten konkrétní produktový use case.

Příklad:
AI funkce v Gmailu, CRM, ticketingu, dokumentačním portálu. U nás je to AI Agent integrovaný do tSM .

2. CLOUDOVÁ DATA + OBECNÁ AI

Kde jsou data:
hlavně v cloudu, často ve více systémech najednou.

S čím pracuji:
s ChatGPT nebo s agentem, který sahá do vzdálených systémů.

Jak se připojuji:
přes MCP, API nebo command line interface.

Dokumenty:
funguje to dobře, když jsou zdroje už připravené přes MCP, API nebo index. Když jsou to jen hromady Wordů v cloudu bez jednoduchého přístupu, práce rychle naráží na limit.

Smysl:
AI umí spojit více cloudových zdrojů, ale hlavní prostředí práce je pořád vzdálené.

3. LOKÁLNÍ AGENT

Kde jsou data:
primárně lokálně na mém počítači nebo v lokálním repu.

S čím pracuji:
s agentem typu **Codex**.

Co má k dispozici:
soubory, git, shell, lokální nástroje a podle potřeby i HTTP, web nebo MCP.

Dokumenty:
dobře fungují, když jsou lokálně a dají se prohledávat standardními tooly. Ideálně textový formát, např. Markdown.

Smysl:
hlavní pracovní prostředí je lokál, cloud je doplňkový zdroj.

Rozdíl není jen v modelu, ale hlavně v tom, kde leží data, jak jsou přístupná a jestli jsou dokumenty připravené pro strojové zpracování.

DATALITE

DEN S AI | AI OBECNĚ A KONTEXT TRHU | Codex

Codex pro prompt: Udělej slide 'Co je v tSM 2.4 nového kolem AI, MCP...'

CO JE V 2.4 NOVÉ

- **tSM AI Docs**: dokumentace má vlastní AI chat a současně běží i jako **MCP server**
- **AI Chat**: vestavěný asistent v top baru, napojený na práci kolem **SpEL Console**
- **Script Bindings**: nové typy bindingů vystaví SpEL script ven jako **REST endpoint** nebo **MCP tool**

JAK TO DO SEBE ZAPADÁ

- **AI Docs** = zdroj pravdy nad dokumentací
- **MCP** = způsob, jak si AI sáhne na dokumentaci nebo tool bez copy-paste
- **Script Bindings** = stejný princip, ale nad naší vlastní tSM logikou

PROČ JE TO RELEVANTNÍ PRO VÝJEZDKO

- naživo ukážeme rozdíl mezi „model si myslí“ a „model si dohledá“
- navazuje to na blok **Procesy a scripty**: tam se půjde do detailu REST/MCP bindingů
- pro nás je to prakticky už teď použitelné pro **developer support, demo flow a hands-on**

2.4 nepřináší jen „AI chat navíc“. Dává dohromady obě poloviny workflow: **znalost** přes AI Docs + MCP a **akci** přes Script Bindings vystavené ven.

DATALITE

DEN S AI | AI OBECNĚ A KONTEXT TRHU | Jiří Bubník

Kde jsme dnes a co přijde dál

DNES

Model sám o sobě už nestačí.
Rozdíl dělá pracovní postup, nástroje a mantinely.

Chat je komodita.
Skoro každý už umí „zeptat se modelu“.

Agenti už něco doručí.
Ne jen text, ale i změnu v repu nebo hotový artefakt.

BĚHEM ROKU

Levnější rutinní práce.
Jednodušší vývoj a příprava podkladů budou výrazně levnější.

Delší autonomie.
Místo minut se budeme bavit o desítkách minut až hodinách práce.

Více konektorů.
MCP a další integrace dostanou AI blíž k reálným firemním systémům.

CO TO ZNAMENÁ PRO NÁS

Vyhrají týmy se standardy.
Instrukce, review, ownership a pořádek budou důležitější než dřív.

Poroste role specializace.
Jiný agent pro analýzu, jiný pro vývoj, jiný pro ops.

Dražší než tokeny bude chaos.
Největší riziko není cena modelu, ale špatně ukotvená práce.

HLAVNÍ MESSAGE

Nejde o otázku, **jestli** AI používat. Jde o to, **v jakém režimu** ji zapojit a jak ji ukotvit do práce týmu.

Programátoři nebudou mít co žrát?
Neubývá potřeba dělat software. Mění se hlavně to, **co je ruční práce** a co už přebírá nástroj nebo agent.

Dotaz do publika
Z medicíny: je dnes AI lepší než radiolog?

Kolik to stojí a co dnes dává smysl

API: CENA ZA 1M TOKENŮ

Model	Vstup	Cache vstup	Výstup
GPT-5	$1.25	$0.125	$10
GPT-5 mini	$0.25	$0.025	$2
GPT-5-Codex	$1.25	$0.125	$10

Hlavní pointa: výstup je násobně dražší než vstup. Mini model je řádově levnější, ale u složitější práce často zaplatíš výkonem nebo počtem iterací.

CODEX V TÝMU

Business a nové Enterprise workspace jedou u Codexu přes **kredity navázané na tokeny**. OpenAI uvádí průměrně zhruba **$100–200 na vývojáře za měsíc**, ale hodně záleží na modelu, režimu a objemu práce.

PŘEDPLATNÉ A PRAKTICKÁ VOLBA

ChatGPT Plus / Pro
Osobní předplatné. Plus je levnější vstup, Pro je pro intenzivní každodenní použití. Obě varianty mají Codex agent; u Pro jsou limity a kapacity výrazně výš.

ChatGPT Business
Firemní workspace, sdílená správa, Codex agent, některé modely „unlimited“, jiné jen „flexible“. Při vyšší spotřebě se dokupují kredity. **Tohle má teď aktuálně každý z nás.**

Junie / JetBrains AI
Také jede přes kvótu a AI kredity. Vyšší tarify mají větší příděl a je možné kredity dokupovat.

Claude
Claude Pro je osobní plán za **$20/měsíc**. Claude Max je osobní plán za **$100 nebo $200/měsíc** s výrazně vyššími limity. Firemně se dnes dá koupit **Claude Team** za **$25 při ročním účtování / $30 měsíčně za člena**; pro práci s Claude Code se u Team/Enterprise přidávají **premium seats za $150 na člena**.

Pracovní závěr pro dnešek: pro jednoho intenzivního uživatele dnes často nejlépe vychází **osobní předplatné**; pro tým dává smysl firemní workspace a průběžně sledovat, kam se posune Codex / Claude / Junie během dalších měsíců.

Flow práce + spec-driven dokumentace

PROBLÉM: INFORMAČNÍ ŠUM V PROVISIONING SYSTÉMECH

- Provisioning je komplexní: jedna změna = 10 dopadů.
- Analýza v Jira issue umírá po uzavření úkolu.
- Dokumentace na Google drive / Confluence žije v izolaci.
- **Důsledek:** Vývojář sice vidí kód, ale nevidí business záměr ("Proč to tu je?").

RIZIKO

AI bez kontextu lže.

Když krmíte AI jen generickým zadáním bez znalosti kontextu, dostanete sice kód, který funguje, ale nedělá to co se očekává.

Specifikace = palivo pro AI.

"Nechceme vyrábět víc dokumentů. Chceme méně ztraceného kontextu, aby AI mohla pracovat za nás."

Vrstvy pravdy: Od záměru po kód

1. ZÁMĚR (WHY)

Business cíl Proč to děláme? Co je scope? Co je out-of-scope?

2. NÁVRH (WHAT)

Systémová analýza Které služby? Které integrace? Datový model.

3. TECH SPEC (HOW)

Kontrakt u kódu Odkazy na moduly. Diagramy v Markdownu. Rozhodnutí a kompromisy.

4. ZDROJÁK (TRUTH)

Implementace Kód + testy. Odkazy zpět na Tech Spec v komentářích.

Spec-driven flow: AI nám pomáhá "protékat" mezi těmito vrstvami (např. vygenerovat draft Tech Spec z Business analýzy).

Příklad: AI jako log-analytik (EIP incident)

VÝSLEDEK AUTOMATICKÉ ANALÝZY EIP (PORUCHA 36800141)

Celkem: 8 OK / 1 FAIL / 0 WARN

Transakce `36800141`, zákazník Gabriela Janouškovcová, produkt Pevný internet, technologie VDSLBONDING. Operace `addHw` skončila chybou 500.

Příčina defektu

Strom služeb v `eipt_service` má **přerušenou hierarchii**. CORE_PRODUCT `O78Z4H7F` má `parent_product_instance = O78Z4H7T`, ale uzel **O78Z4H7T neexistuje**. CP je tím odpojený od TECHNOLOGY_POINT a EIP nemůže nalézt TP předka.

Uzel `O78Z4H7T` je pravděpodobně mezilehlý uzel z CRM (Commercial Tariff), který nebyl naloadován.

Nalezené problémy

#	Kontrola	Výsledek	Detail
05.X	Celistvost stromu	FAIL	CP `O78Z4H7F` má parent `O78Z4H7T` (neexistuje)

Doporučení

Ověřit CRM data pro uzel `O78Z4H7T`.
Ověřit logiku načítání stromu (mezilehlé uzly).
Zvážit přemapování parent CP přímo na TP.

Automaticky vygenerováno nástrojem eip-analyze.

Očekávaná hierarchie (Tech. detail)

```
TARIFF_SPACE (O79QDVGH)
└── TECHNOLOGY_POINT (O78Z4H7G)
    └── CORE_PRODUCT (O78Z4H7F)   ← parent by měl být G
        └── TECHNICAL_TARIFF (O79QDVGK)

Skutečný stav v DB:
TARIFF_SPACE (O79QDVGH)
├── TECHNOLOGY_POINT (O78Z4H7G)
└── VAS (O79QDVGJ)

CORE_PRODUCT (O78Z4H7F)          ← parent = O78Z4H7T (FAIL)
└── TECHNICAL_TARIFF (O79QDVGK)
```

Aha moment: AI za 5 sekund našlo v DB rozbitý strom, který by vývojář ručně hledal 15 minut v SQL.

Demo (1/4): Od přání k analýze

1. VSTUP OD KLIENTA (STROHÁ JIRA ISSUE)

"Potřebujeme přidat bandwidth limit na GPON přípojky pro retail."

2. ANALÝZA DODAVATELE (AI + KONTEXT)

- **Analytik se ptá AI:** "Kde v systému řešíme GPON profily a jaké jsou limity?"
- **AI (přes MCP/RAG) odpovídá:** "Profily jsou v `ServiceConfig`, validace v `ProvisioningCore`. Limity pro retail jsou obvykle 10-1000 Mbps."
- **Výsledek:** Zpřesněné zadání s definovanými business pravidly a dopady.

ZPŘESNĚNÝ KONTEXT (OUTPUT ANALÝZY)

```
## Business pravidla
- Limit: 10 - 1000 Mbps
- Jen pro typ: RETAIL_GPON

## Dotčené moduly (High-level)
- `provisioning-service` (validace)
- `inventory-api` (nový atribut)
- `tSM-mcp` (dokumentace parametrů)
```

Tento draft vznikl za 2 minuty díky AI, která zná naši dokumentaci v tSM.

Demo (2/4): Technický návrh (Tech Spec)

3. VÝVOJÁŘ (TECHNICKÁ SPECIFIKACE)

- **Vstup:** Zpřesněná analýza z předchozího kroku.
- **Akce vývojáře:** "Agente, navrhni technickou specifikaci přímo do repa."
- **Výsledek:** Markdown soubor vedle kódu definující kontrakt a dopady.

TECH-SPEC-GPON-LIMIT.MD (V REPU)

```
## Implementační detaily
- Update `cz.datalite.gpon.GponProfile`
- Nový validační rule v `ActivationFlow`
- Odkaz na kód: `src/main/java/.../GponService.java`

## Testy
- [ ] Unit: Validace rozsahu (10, 1000, 5000)
- [ ] Integration: End-to-end aktivace s limitem
```

Demo (3/4): Implementace a logika

4. VÝVOJÁŘ (KÓD A VYSVĚTLENÍ)

- **Akce:** AI vygeneruje kód a vývojář se ptá na detaily.
- **Otázka:** "Proč je validace v `ActivationFlow` a ne v DTO?"
- **AI vysvětluje:** "DTO je jen přepravka. Business logiku držíme ve flow, aby byla znovupoužitelná pro REST i SOAP rozhraní."

```
// AI: Implementace validačního pravidla
public void validate(GponProfile profile) {
  if (profile.isRetail()) {
    int maxLimit = config.getRetailLimit(); // 1000
    if (profile.getBandwidth() > maxLimit) {
      throw new ValidationException("Exceeded retail limit: " + maxLimit);
    }
  }
}
```

Insight: *AI mi nejen napsalo kód, ale obhájilo i architektonické rozhodnutí, které jsme zapsali v Tech Spec.*

Výsledek: Dokumentace (Tech Spec) vzniká jako vedlejší produkt vývoje, ne jako administrativa navíc.

Demo (4/4): AI jako detektiv (Souvislosti)

5. ANALÝZA DOPADŮ (CO NÁM UNIKLO?)

- **Otázka na AI:** "Když změníme GPON limit, co všechno se může rozbít v reportingu?"
- **Aha moment:** AI (díky indexaci celého projektu) upozorní na zapomenutý legacy SQL skript v `billing-export`, který filtruje profily podle starých rozsahů.

NALEZENÉ SOUVISLOSTI

- ⚠️ **Billing:** `legacy_report_v2.sql` – natvrdo zadrátované limity.
- ⚠️ **Monitoring:** Zabbix trigger pro "High bandwidth" může začít falešně alarmovat.
- ✅ **Frontend:** UI komponenta `BandwidthSlider` už používá dynamický config.

Tuhle souvislost s billingem by analytik v rámci návrhu pravděpodobně přehlédl.

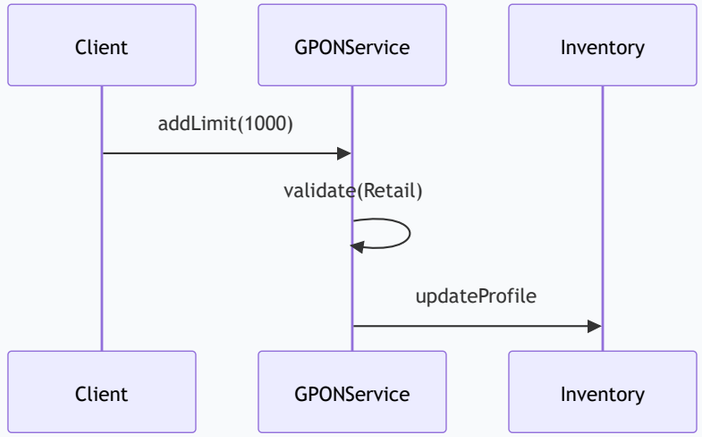
Udržování: Dokumentace, která dýchá

DNEŠNÍ REALITA: DETEKTIVNÍ PRÁCE

- **Archeologie:** Prohledávání neaktuálních Google drive a mrtvých Jira issue.
- **Kontext v mlze:** Prohledávání GIT commitů, emailů a zpovídání kolegů.
- **Cíl:** Pochopit "Proč" a "Jak" to v kódu funguje (často po letech).
- **Důsledek:** Extrémně drahá zastupitelnost a riziko chyby při změně.

CÍL: ŽIVÝ ORGANISMUS (S AI)

- **Zastupitelnost:** Kdokoliv pochopí záměr z repa během sekund.
- **Zjednodušení:** AI "překládá" historii a souvislosti za vás.
- **Udržování:** Diagramy a Tech Spec se dýchají s kódem (AI update).



AI dokáže udržovat i diagramy v souladu s realitou kódu.

Nástroje: AI na dosah

AI JE BLÍŽ, NEŽ SI MYSLÍTE

- **Lokální kontext:** Codex (OpenAI) připojený k projektu – vidí lokální soubory a navrhuje změny přímo formou Diffu.
- **Indexace:** Nástroje jako Cursor nebo JetBrains AI indexují celý projekt (nejen otevřený soubor).
- **Bezpečnost:** Kontext zůstává u vás, do Cloudu jdou jen anonymizované fragmenty.

PŘÍKLAD PROMPTU PRO LOKÁLNÍ AI

"Prohledej projekt a najdi všechny výskyty práce s `VDSLBONDING`. Vytvoř mi z toho přehlednou tabulku parametrů pro naši tSM dokumentaci."

Výhoda: Nemusím nic nahrávat na web. AI pracuje nad mými lokálními daty v reálném čase.

Praktické kroky: Jak začít v pondělí?

CO ZAČÍT DĚLAT (START)

- **Psát Tech Spec do repa.** V Markdownu, kousek od kódu.
- **Linkovat kód.** Používat relativní cesty, ať se AI chytne.
- **Zpřesňovat po vrstvách.** Nechtít po AI kód z jedné věty v Jira issue.

CO PŘESTAT DĚLAT (STOP)

- **Psát romány v DISu/Jiře.** Dejte tam jen odkaz na Tech Spec v repu.
- **Schovávat analýzu v hlavě.** Co není v textu, to AI nevidí.
- **Duplikovat info.** Jeden zdroj pravdy, AI ho rozdistribuuje.

SPOLEČNĚ DÁL

- **Teams skupina:** Sdílení novinek, promptů a "how-to" postupů napříč týmy.
- **AI Rozcestník:** Příprava centrálního místa pro nastavení a best practices.
- **Zpětná vazba:** Možnost řešení vašich konkrétních požadavků s AI.

tSM 2.4: stav release a projektový kontext

STAV RELEASE

- **tSM 2.4 byla oficiálně hotová ke konci března 2026**
- po uzavření release pokračují zákaznické dodělávky a doplnění chybějících částí
- aktuálně jde hlavně o dokončení položek pro **CETIN** a o platformní dopracování 2.4

PROJEKTOVÉ TEĎ

CETIN ticketing je v testech a nejbližší krok je nasazení do provozu

Slovanet po Q2 po ukončení Q2 2026 a po akceptaci má proběhnout převod celého Slovanetu na 2.4

Další projekty po stabilizaci 2.4 mají pokračovat další témata, včetně Pčka

DOKUMENTACE 2.4

- přehled release změn: Version 2.4
- detailní technický popis novinek je v oficiální dokumentaci tSM 2.4

Výhled 2.5

STAV PLÁNOVÁNÍ

- vývoj **2.5** poběží paralelně s dokončováním 2.4 rolloutů
- scope ještě není uzavřený
- část témat se bude upřesňovat podle zkušeností z CETINu, Slovanetu a dalších nasazení

PŘEDPOKLÁDANÝ SMĚR

- pokračování **AI** témat
- další platformní změny navazující na 2.4
- návaznost na Git-first správu konfigurace, CLI a integrační workflow

Co se mění v backend workflow

- Script Bindings: stejný script umí ven jako REST i MCP tool
- Task Templates: méně ruční BPMN konfigurace
  - ukázka šablon: **Jira task** a **AI Task**
- Process Engine: External Task, script binding v tasku
  - ukázka **designer módu**
  - rozdíl mezi **starým zápisem** a **novým zápisem přes Task Template**
- SpEL: business kalendáře a termíny bez ruční magie
- Public API V2: jasnější kontrakt a menší chaos v integracích

Nejsilnější téma pro AI integrace: Script Bindings + MCP

Procesy a scripty — co si ukážeme

Co je nového ve SpEL konzoli?

- Informace o skriptu
- Definice vstupních a výstupních atributů
- Script Bindings

Script Bindings — demo

- **Entity Events** → `DemoTicketUppercase`
- **Script Invocations** → `DemoTicketComment`
- **Rest Invocations** → `DemoTicketComment`
- **Mcp Invocations** → `DemoTicketMcp`

Prompt pro MCP ukázku:

> Pomocí `datalite_ref_ticket` MCP serveru mi najdi ticket **DTP17** a shrň mi jeho stav, prioritu a klíčové detaily.

Co je nového v Process designeru?

- Designer mod
- Task template
- starý vs. nový zápis

Co si mají lidi odnést

- script už není jen interní logika, ale kontrakt
- form definition není jen UI, ale i schema a validace
- MCP není magie navíc, ale standardizované rozhraní nad tSM logikou
- API V2 říká, jak má integrace přežít upgrade

Formuláře, efekty, widgety

- Form Effects: clear, set, copy, script
- widget-level efekty: reakce na save, button, script response
- méně ad-hoc hacků ve formulářích
- pozor na upgrade 2.4: formuláře mají i migrační změny

Nejde jen o „hezčí formuláře“. Jde o rychlejší a přesnější konfiguraci chování UI.

AI v reálném projektu: Slovanet

- samostatný AI repo pro projekt
- instrukce pro SharePoint, DevOps a tSM kontext
- reálné workflow místo laboratorního dema
- co funguje a co pořád bolí

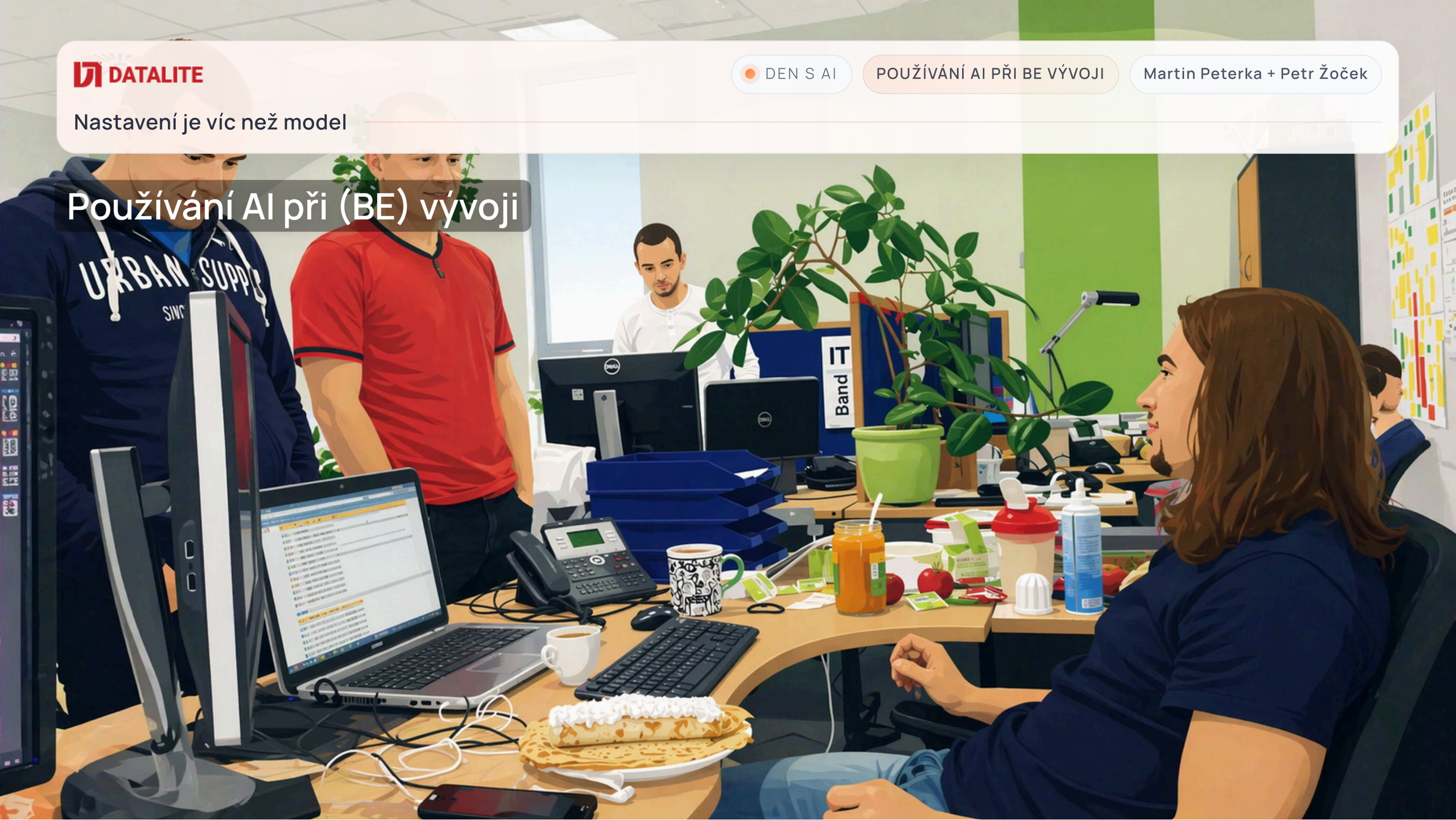
> Když to jde v projektu Slovanet, jde to přenést i do dalších projektů.

DATALITE

DEN S AI | POUŽÍVÁNÍ AI PŘI BE VÝVOJI | Martin Peterka + Petr Žoček

Nastavení je víc než model

Používání AI při (BE) vývoji



Nastavení je víc než model

Agent je super.
Ale bez nastavení je jako senior programátor, kterého posadíte k neznámému projektu, zadáte mu úkol a čekáte, co se stane.

- používáme
  - **JetBrains AI**
  - **Codex**
  - **Claude Code**
  - další?
- pointa není *vendor comparison*
- pointa je: `má agent kontext, pravidla a ohraničení?`

KDYŽ CHYBÍ NASTAVENÍ

- každý nástroj navrhne něco jiného
- agent neví, co je zdroj pravdy
- sahá do špatných souborů
- rychle vyrábí šum místo hodnoty
- opakuje stále dokola zbytečné kroky

Největší rozdíl nedělá model. Největší rozdíl dělá připravenost prostředí.

AGENTS.md a skills nejsou totéž

AGENTS.MD

"Knowledge playbook"

- kontrakt pro AI v konkrétním repository
- účel repa, konvence, citlivá data
- co je zdroj pravdy
- jak vypadá hotový výstup
- jaké jsou souvislosti

Repo-specific pravidla. Verzovat v Gitu.

SKILLS

"Expertíza v akci"

- opakovatelný workflow (reusable)
- zapouzdřený návod na jednu činnost
- ticket → spec → diff
- log analysis / review checklist
- nezávislé na konkrétním repu

Cross-project pattern. Vaše "dílenské nářadí", které si berete s sebou.

Praktické pravidlo: co je specifické pro repo patří do `AGENTS.md`. Co se opakuje napříč projekty je kandidát na shared skill.

Skills: Expertní dovednosti v krabičce

Skill = Přenositelné know-how

NEŘEŠÍME KDE JSME, ALE JAK DĚLÁME KONKRÉTNÍ ŘEMESLO.

1. ANALÝZA & DESIGN

Zadání → Specifikace

- Načtení Jiry + tSM kontext
- Kontrola proti Public API
- Návrh sekvenčního diagramu

2. VÝVOJ & REVIEW

Kód → Kvalita

- Kontrola SpEL a tSM pravidel
- Automatizovaný checklist
- Unit testy podle standardu

3. PROVOZ & OPS

Log → Řešení

- Identifikace známých chyb
- Prohledání báze znalostí
- Návrh opravy / workaroundu

PRAKTICKÝ DOPAD

Delegujte rutinu, ne zodpovědnost.

Skill zajistí, že standard seniora je dodržen v každém projektu.

Reálné Skills z našich projektů

OD ARCHITEKTURY PO AUTOMATICKÝ CHANGELOG.

jira-to-spec (Analytikův parťák)

Vezme strohé zadání z Jiry, prozkoumá repo a navrhne technickou specifikaci v Markdownu. Most mezi byznysem a kódem, který šetří hodiny diskusí.

generate-javadoc (Konec prázdných slibů)

Automatizuje nudnou práci. Generuje věcný český JavaDoc se zaměřením na byznys logiku ODS a idempotenci.

identify-bpmn-tasks (Rentgen kódu)

Propojuje kód s procesní realitou v DB. Mapuje Java metody na názvy tasků v BPMN – přesně to, co AI bez skillu nevidí.

post-task-documentation (Autonomní kolega)

Sám si pohlídá konec úkolu. Zapíše změny do CHANGELOG.md a vytvoří analýzu. Disciplína, která se sama vynucuje.

⚠️ **Poučení z "generate-doc-md":**
I skilly se dají "přepreparovat". Pokud má instrukce >300 řádků, LLM ztrácí fokus. **Lepší strategie:** Rozdělit na menší, specializované pod-skilly.

Demo: Skill v akci – Analýza chyby

Reálný bug: "To vypadá správně, ne?"

ANALÝZA PROVISIONINGU BEZ DAT Z PROSTŘEDÍ ZÁKAZNÍKA.

```
val parentPID = asset.linksFrom
    ?.firstOrNull { it.linkTypeId == "RELATED" }
    ?.toAssetId
    ?.let { assetsById[it] }
    ?.takeIf { it.productType == TECHNOLOGY_POINT }
    ?.ext?.eternalRefNo

if (parentPID.isNullOrBlank()) {
    val tpAssetId = asset.linksFrom
        ?.firstOrNull { it.linkTypeId == "RELATED" }
        ?.toAssetId ?: throw Exception(...)
    // fallback...
}
```

🧠 Lidské čtení (Předpoklad)

- "Vezmu linky, zafiltruji RELATED..."
- "Najdu TECHNOLOGY_POINT."
- "Pokud nenajdu, zkusím fallback z prvního linku."
- **Závěr:** Kód dává smysl.

🤖 AI Skill (Analýza)

- **Kritická chyba:** `firstOrNull` není filtr.
- Pokud je první RELATED link jiný než TP, `takeIf` selže.
- **Dopad:** Ignorujeme platný TP link na 2. pozici a končíme v chybě/fallbacku.

🎯 Odhaleno okamžitě skillem Log & Code Analyzer

🛡️ *Prevence: Při generování z precizní specifikace by k chybě vůbec nedošlo!*

Samotné instrukce nestačí

CO SE UKÁZALO V PRAXI

- orchestrace velkých tasků, klidně `8–10` paralelně
- limit nebyl kredit, ale ztráta směru
- agent zapomíná rychle a snadno ujede
- samotné `AGENTS.md` nestačí jako jediný guardrail

Poučení: instrukce jsou nutné, ale bez tvrdého ověření nejsou dost.

POSTUP

- obalit workflow deterministickým toolingem
- přidat linty a analyzery místo spoléhání na disciplínu agenta
- do agentních instrukcí dát povinnost tyto kontroly spouštět
- agent se má řídit výstupem nástrojů, ne dojmem

Agent má být zrychlovač. Koleje mu dává tooling.

Co sdílet v Gitu

DO GITU ANO

- `AGENTS.md`
- šablony a checklisty
- helper skripty
- architektura a kontext
- sdílené workflow

DO GITU NE

- secrets a tokeny
- osobní session
- lokální cesty
- cache a temporary files
- osobní promptovací hacky
- .claude/settings.**local**.json

K DISKUSI

- markdown dokumenty použité při plánování

Přenášet mezi projekty chceme patterns, ne osobní nastavení.

Kdy použít multiagent

Multiagent funguje jen tehdy, když je práce rozdělená stejně disciplinovaně jako u lidí.

MULTIAGENT

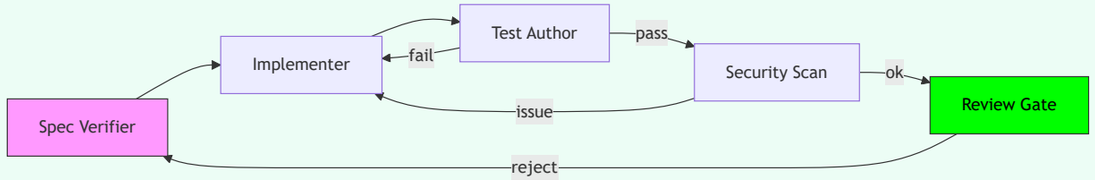
- ano: nezávislé podúkoly
- ne: všichni sahají do stejného místa
- musí být jasný ownership
- jeden koordinuje, ostatní dodávají

Multiagent

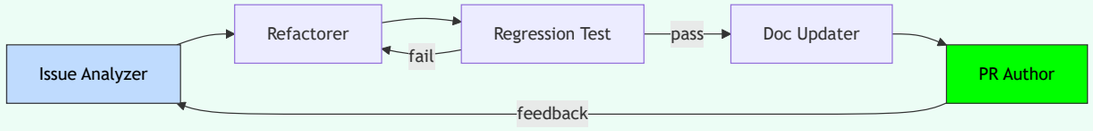
PRAKTICKÝ CHAIN - PŘÍKLADY

@MarekZima?

NOVÁ FEATURE



REFACTORING / BUGFIX



Shrnutí: AI jako parťák, ne černá skříňka

Tři pilíře úspěchu

Kontext je král – Bez `AGENTS.md` a jasného zadání pálíte tokeny i čas.
Tooling jsou koleje – Nespoléhejte na "slib" agenta, věřte lintu a testům.
Sdílení vzorů – Co funguje jednou, má být `skill`.

Co si odnést?

- [ ] Mám v repu `AGENTS.md`?
- [ ] Jsou v něm definované "zdroje pravdy"?
- [ ] Používám agenta jako zrychlovač, nebo po něm jen uklízím?

AI nefunguje místo nás, ale s námi. Kvalita vstupu a prostředí určuje kvalitu kódu.

Configuration Explorer, CLI a Studio

CONFIGURATION EXPLORER

- primární in-app pohled na organizaci konfigurace v tSM
- katalog položek podle `ConfigType -> EntityType -> Item`
- zobrazuje package metadata, Installed Package a drift stav
- slouží pro orientaci, review a dohledání stavu konkrétní položky

CLI

- technický nástroj pro package workflow mezi prostředími
- typický flow:
  - `download`
  - `validate`
  - `diff`
  - `install`
  - `verify`
- dokumentace počítá i s CI/CD použitím

STUDIO

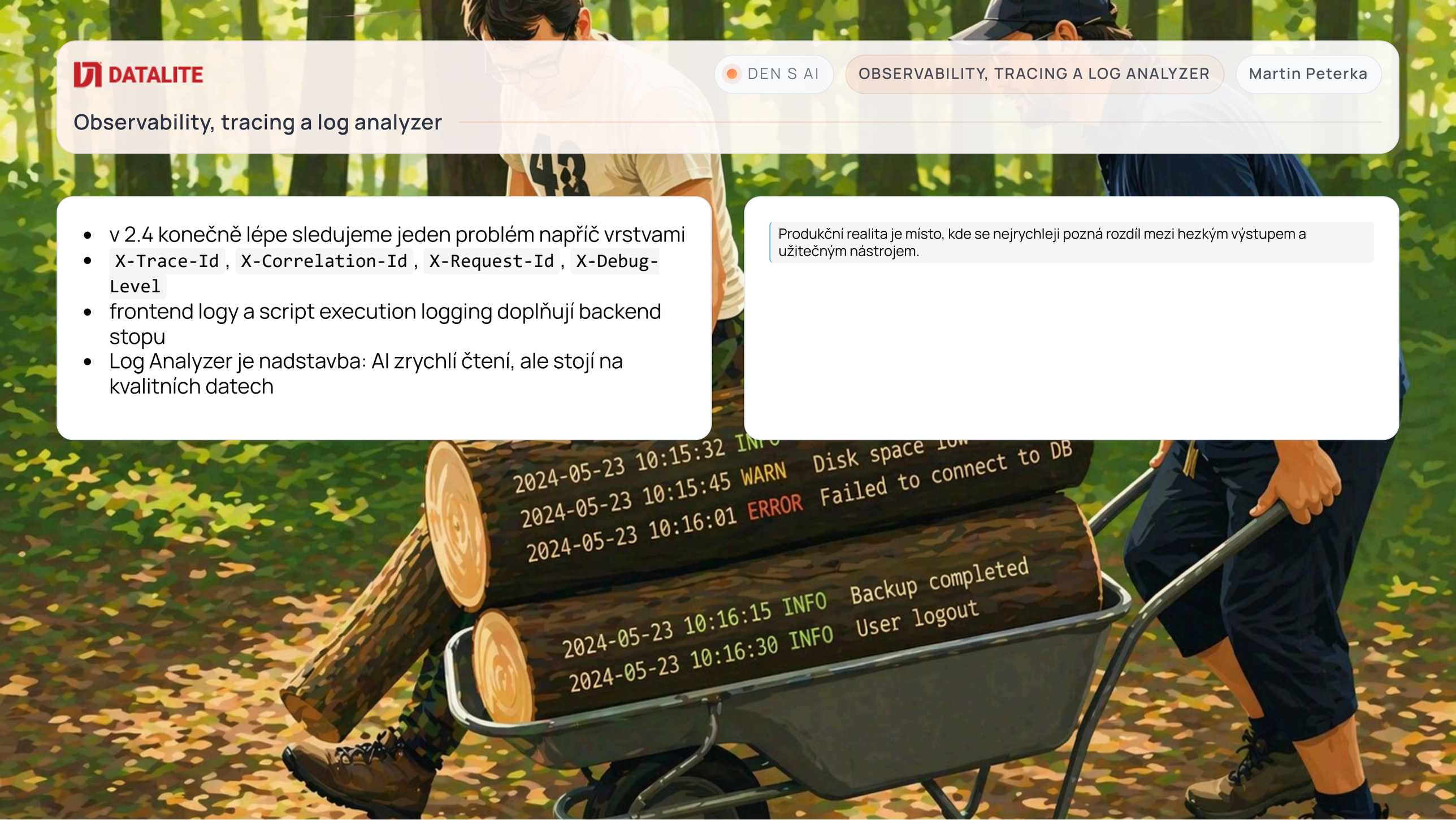
- v dokumentaci vedené jako VS Code extension
- backup, sync a Git integration nad konfigurací
- produktově existuje, ale u nás zatím není uzavřené, jakou bude mít roli v běžném workflow
- otevřená otázka: Explorer + CLI + Git vs. širší Studio workflow

Dokumentace: [Configuration Explorer](https://tsm-docs.datalite.cloud/docs/next/configuration/common/ConfigurationExplorer/), [Configuration Lifecycle](https://tsm-docs.datalite.cloud/docs/next/tsm_studio/Configuration_Lifecycle/), [tSM CLI](https://tsm-docs.datalite.cloud/docs/next/tsm_studio/CLI/), [tSM Studio Help](https://tsm-docs.datalite.cloud/docs/next/tsm_studio/tSM_Studio/)

Observability, tracing a log analyzer

- v 2.4 konečně lépe sledujeme jeden problém napříč vrstvami
- `X-Trace-Id`, `X-Correlation-Id`, `X-Request-Id`, `X-Debug-Level`
- frontend logy a script execution logging doplňují backend stopu
- Log Analyzer je nadstavba: AI zrychlí čtení, ale stojí na kvalitních datech

> Produkční realita je místo, kde se nejrychleji pozná rozdíl mezi hezkým výstupem a užitečným nástrojem.



Observability, tracing a log analyzer

📋 Co je to log?

> Log = záznam o tom, co se stalo.

Systém si průběžně zapisuje, co dělá:

- `10:42:01` – Přišel požadavek na vytvoření objednávky
- `10:42:02` – Zavolali jsme CIP
- `10:42:03` – CIP odpověděl: `200 OK`
- `10:42:05` – Objednávka vytvořena

Tři druhy logů

Typ	Co to je	Kdo to čte
Application log	`ERROR: NullPointerException`	Vývojáři
Audit log	„Pole Stav změnil…"	Support
Log entita	„Volali jsme API…"	Konfigurátoři, support

🗃️ Log entita v tSM

Log entita zachycuje **integrační a systémové události** navázané na konkrétní záznamy (Ticket, Objednávka, Zákazník…).

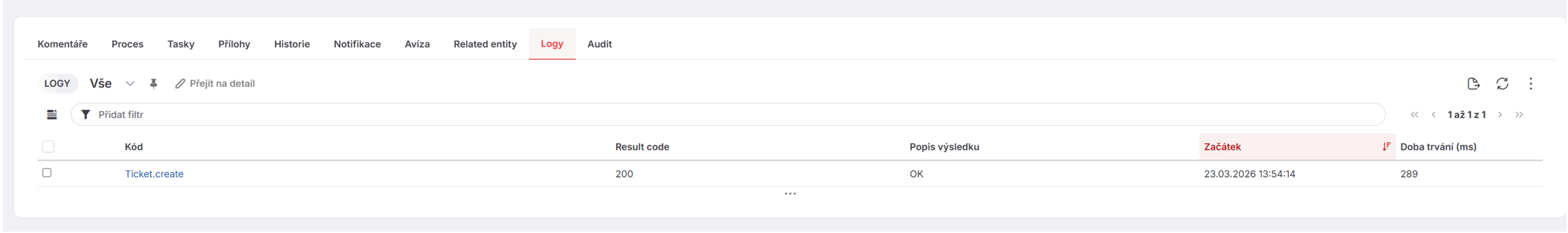
Není to jen „výpis chyb pro programátory" – je to **business záznam**, který:

- dokazuje, že jsme poslali zprávu externímu systému
- uchovává, co jsme mu řekli a co odpověděl

```
ownerType + ownerId  →  Ticket #12345        (ke komu patří)
logTypeCode          →  Ticket.create         (typ události)
startDate / endDate  →  10:42:01 – 10:42:03  (kdy)
request              →  { "key": "Ixodes-1758" }
response             →  { "status": "OK" }
resultCode           →  200
resultMessage        →  "OK"
```

Observability, tracing a log analyzer

Tyto logy se zobrazují u entit, kterých se týkají - např. u Ticket, panel Logy.



Observability, tracing a log analyzer

👁️ Aplikační logy & Observability

CO TO JE?

Sledování = schopnost **vidět, co se děje uvnitř systému** – i napříč více službami najednou.

HTTP a Kafka hlavičky pro sledování

Každý požadavek, který projde systémem, nese identifikační „štítky" (v HTTP nebo Kafka headers):

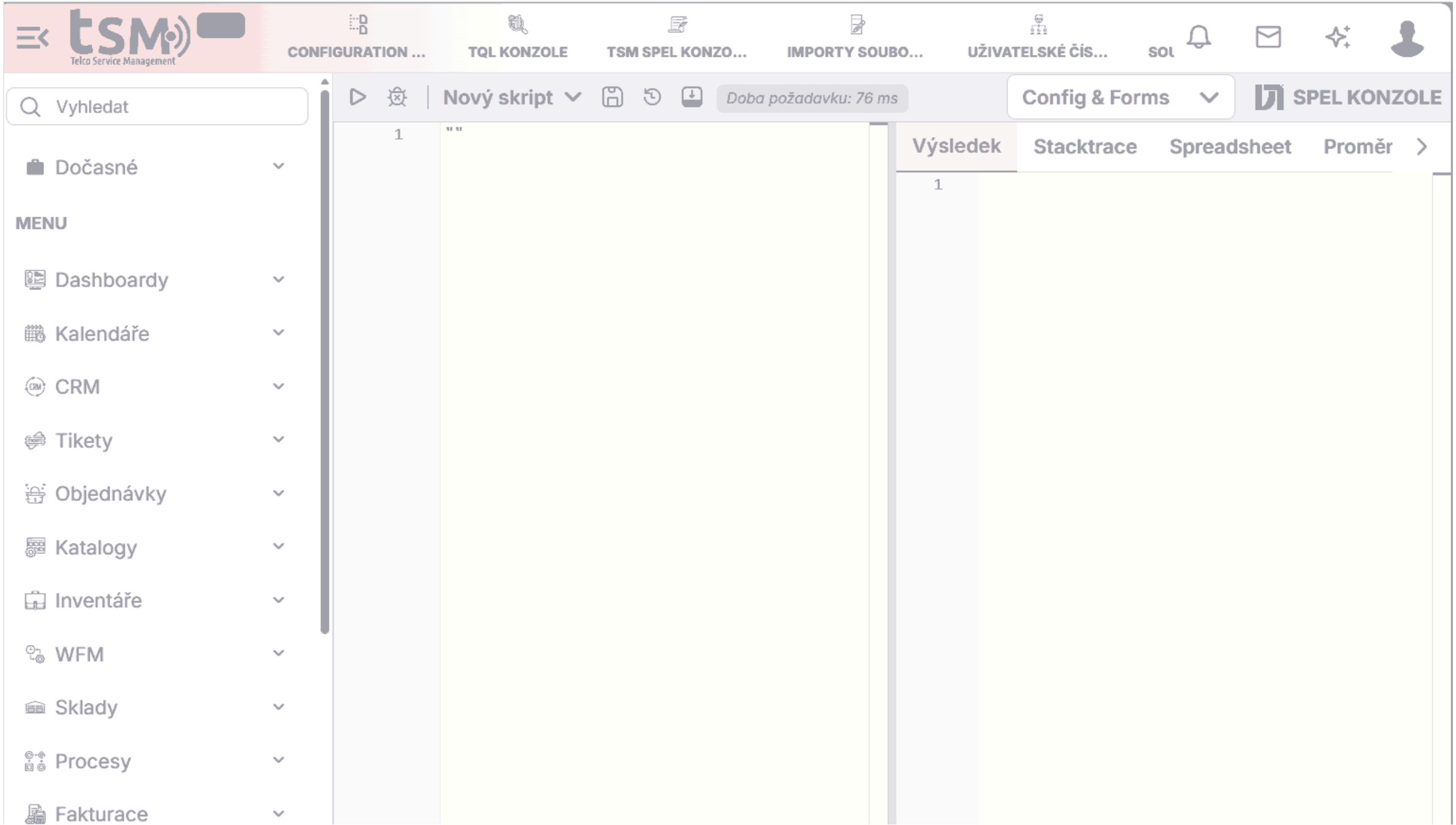
Hlavička / Header	Co to dělá
`X-Trace-Id`	Sleduje cestu jednoho požadavku
`X-Correlation-Id`	Drží pohromadě akce jedné uživatelské relace
`X-Request-Id`	Unikátní ID každého volání
`X-Debug-Level`	Dočasně zapne podrobnější logování

X-TRACE-ID V PRAXI

Uživatel klikne na „Odeslat objednávku". Systém vygeneruje `X-Trace-Id: abc-123` a toto ID **putuje se všemi voláními**:

```
Uživatel → Frontend → Backend API → Databáze / CIP / E-mail
```

> Všechny kroky zalogují `abc-123`, pak je možné dohledat **celou cestu** té jedné akce.


Telco Service Management
CONFIGURATION ...
TQL KONZOLE
TSM SPEL KONZO...
IMPORTY SOUBO...
UŽIVATELSKÉ ČÍS...
SOU
Vyhledat
Dočasné
MENU
Dashboardy
Kalendáře
CRM
Tikety
Objednávky
Katalogy
Inventáře
WFM
Sklady
Procesy
Fakturace
Nový skript
Doba požadavku: 76 ms
Config & Forms
SPEL KONZOLE
1
""
Výsledek
Stacktrace
Spreadsheet
Proměr
1

Observability, tracing a log analyzer

X-DEBUG-LEVEL

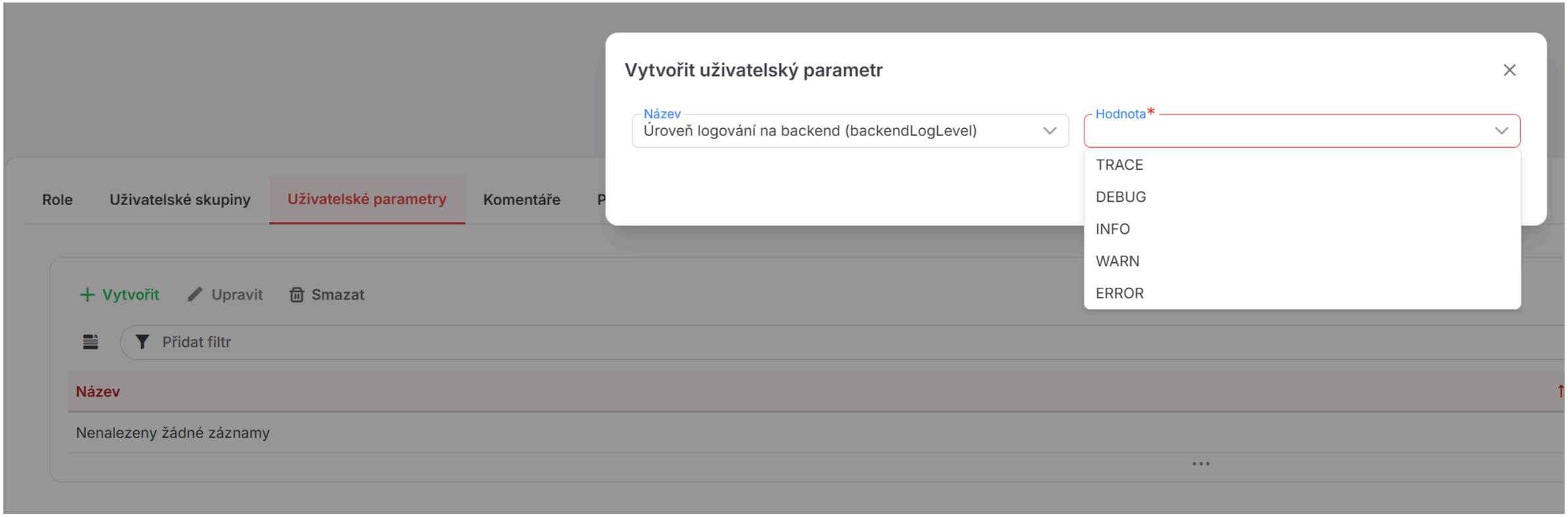
Normálně systém loguje jen důležité věci (`INFO` , `WARN` , `ERROR`).

Přidáte hlavičku `X-Debug-Level: DEBUG` a systém **pro tento jeden požadavek** začne zapisovat detailní logy.

> Hodí se pro ladění konkrétního problému, aniž by se zahltily logy pro ostatní uživatele.

Uživatelský parametr

Podrobné logování je možné nastavit pro celé uživatelské sezení.



Scoped logování ve skriptech

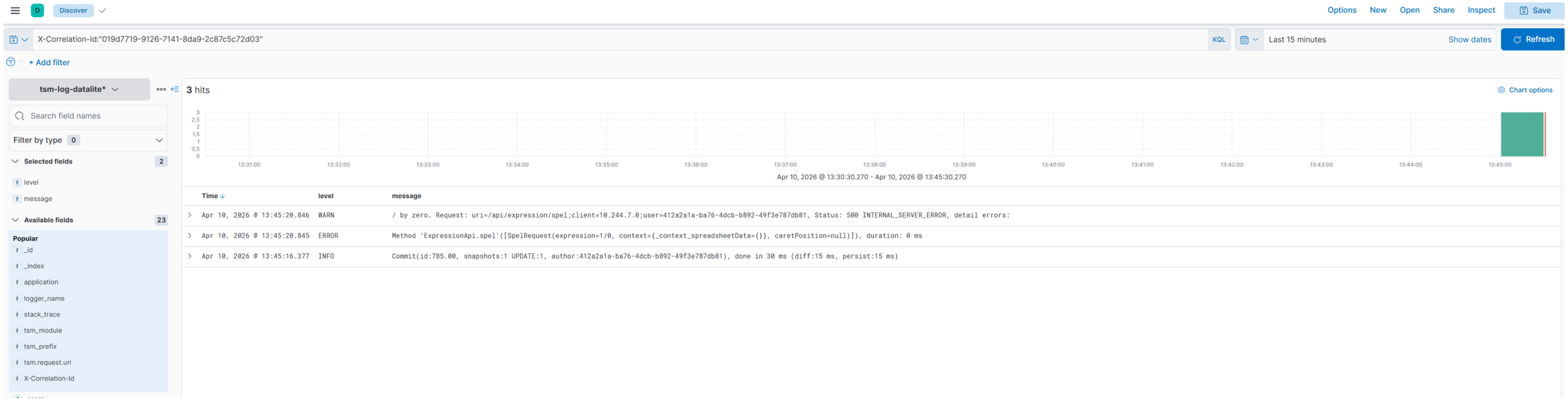
Ve SpEL skriptech lze zvýšit úroveň logování jen pro konkrétní blok kódu:

```
@logger.withLevel('DEBUG').do(
    #x = 10,
    #y = 20,
    @logger.debug("Calculation: " + #x + " + " + #y),
    #x + #y
)
```

Úroveň se zvýší jen pro výrazy uvnitř `.do()` bloku. Po skončení se vrátí do původního stavu.

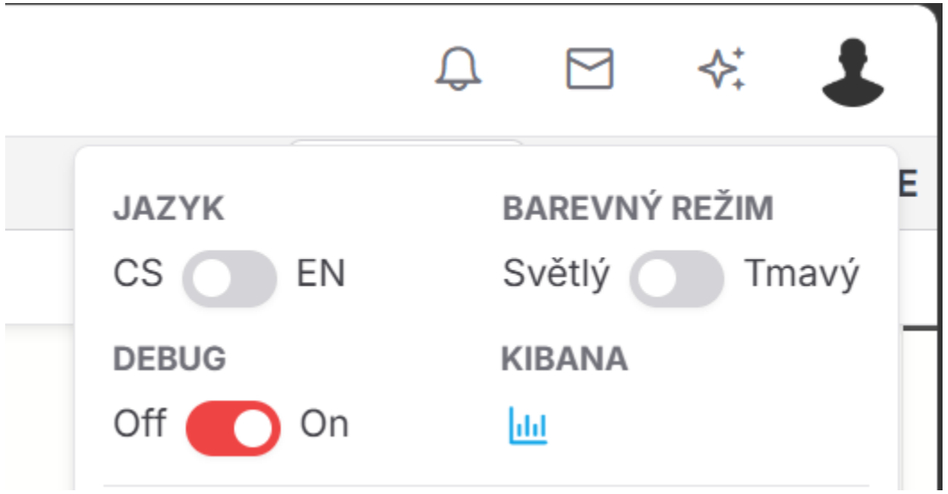
Kibana

Kibana je **webový nástroj pro prohlížení logů**. Umožňuje prohledávat a filtrovat záznamy ze všech služeb na jednom místě.



ZÁKLADNÍ POUŽITÍ (TROUBLESHOOTING)

Odkaz do Kibany, s přednastaveným filtrem:



🎯 Shrnutí

- **Log entita** – Business záznam integračních událostí u entit.
- **Audit** – Kdo, kdy a co změnil na datech.
- **Observability v 2.4** – Standardní hlavičky (`X-Trace-Id`) pro end-to-end sledování.
- **Kibana** – Centrální místo pro prohledávání všech aplikačních logů.

AI Chat a další kroky

- AI Chat je už přímo v produktu, ne bokem
- 2.4 míří hlavně na developer support kolem SpEL Console
- další krok: Process Designer a Form Designer
- E2E testy psané Claude = náš interní pattern práce
- přechod od asistence ke climax demu a automatizaci

E2E demo

Jira → MR → cloud

- ticket s anotací
- automaticky připravený MR
- nasazení do cloudu
- vývojář schvaluje, nekliká všechno ručně

> Tady už nejsme u „AI něco napsala“. Tady jsme u pracovního toku, který se částečně sám hýbe.

Spec → zelený test, 4 agenti v řadě

Z textového popisu scénáře na zelený E2E test

- vstup: markdown popis business scénáře
- výstup: Playwright test + úprava sdílených page-objectů
- všechno s lintem a typecheckem v zeleném
- čtyři specializovaní agenti, ne jeden všeumělec

> „Planner, generator, healer a refactor. Každý umí jen jednu věc — a díky tomu ji umí dobře."

Takhle vypadá vstup — zadání

Ukázka zadání: Optik 300 pro existujícího zákazníka

NÁZEV

Zřízení nové služby `Internet Optik 300 MBit/s` s `Káblovým modemem` a `Slovanet TV` s `Media boxem` na konkrétní adrese pro existujícího zákazníka — instalace technikem, doručení zařízení technikem, podpis smlouvy na predajni, elektronická fakturace `Polročne`.

CÍL

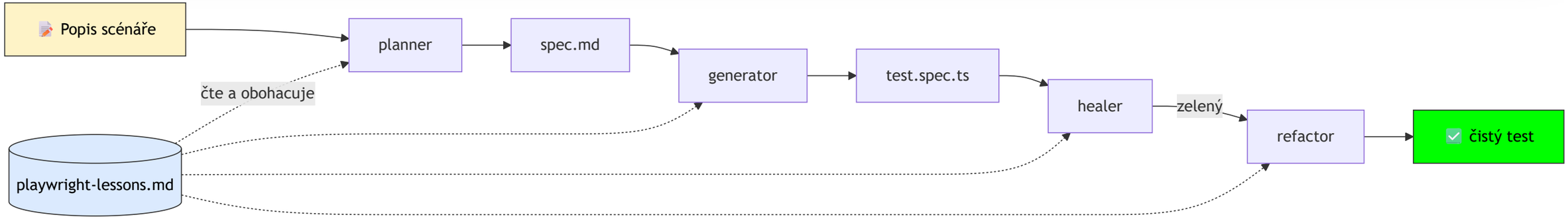
Ověřit celý end-to-end proces: od založení objednávky přes výběr zákazníka a adresy, konfiguraci balíku `Optik 300`, provisioning, WFM instalaci technikem, předání zařízení, až po elektronickou fakturaci `Polročne`.

PŘEDPOKLADY

- oprávnění vytvořit objednávku `Aktivácia novej služby`
- testovací zákazník v CRM vyhledatelný v objednávce
- primární osoba s e-mailem, další osoba s e-mailem pro `Prihlasovacie údaje`
- instalační adresa s dostupnou technologií pro balík `Optik 300`
- v katalogu dostupné `Internet Optik 300 MBit/s`, `Káblový modem`, `Slovanet TV`, `Media box`
- možnost `Instalácia s výjazdom technika`
- možnost `Doručenie zariadenia technikom pri inštalácii`

Nic víc. Jen markdown v `apps/slovanet-e2e/specs/ordering/` — zbytek si pipeline odvodí.

Čtyři agenti, jedna pipeline



Spojovací vlákno: `playwright-lessons.md` — sdílená paměť, každý agent ji čte první, výstupy ji obohacují.

Kdo co dělá

`PLANNER` — SPEC DESIGNER

- **vstup:** popis scénáře, aplikace
- **výstup:** `specs/**/*.md`
- snímkuje živé UI, odhalí DOM quirky
- flaguje nejistoty pro další agenty

`GENERATOR` — IMPLEMENTACE

- **vstup:** cesta ke spec
- **výstup:** `tests/**/*.spec.ts` + úpravy page-objectu
- spec → test v jednom kroku
- označí selector assumptions pro healera

`HEALER` — DEBUG A OPRAVY

- **vstup:** spec + živá app
- **výstup:** zelený test
- spouští i **sourozenecké testy** (C-7 check)
- nesmí rozbít existující, když opravuje nový

`REFACTOR` — ČISTÝ KÓD

- **vstup:** zelený test + page-object
- **výstup:** stejné chování, méně řádků
- precondition: test musí být zelený
- dedup, ne přidávání obrany navíc

Reálný průběh — Optik 300 na Slovanetu

ZADÁNÍ OD VLASTNÍKA

> „Zřízení služby Internet Optik 300 + Slovanet TV + Media box na adrese pro existujícího zákazníka, instalace technikem, elektronická fakturace s periodicitou Polročně."

PRŮBĚH

Fáze	Trvání	Výstup
`planner`	15 min	spec + 3 UI discovery findings
`generator`	7 min	nový test + 4 metody v `Order`, 3 flagnuté selektory
`healer`	11 min	test PASS na první pokus, 1 timeout fix v sourozenci
`refactor`	32 min	-4 řádky, dedup `setEndDevice`, 0 regresí

CO SE NAUČILA KNIHOVNA

- **A-8** — radio-button package cards: klikat na `<label>`, ne `<input>` (PrimeNG)
- **B-4** — serial run email re-check potřebuje 30 s timeout
- **C-8** — dedup wrapper pro section-parametrizované helpery

Zapsáno do `playwright-lessons.md`, dostupné pro všechny budoucí testy napříč E2E projekty.

Celkem ~65 minut čistého času agentů. Výsledek: zelený test v CI + tři nové vzory pro příště.

Co si odnést

ROLE, NE JEDEN VŠEUMĚLEC

Každý agent má úzkou odpovědnost a čistý handoff. Nečekáme, že jeden „to všechno zvládne".

LESSONS JAKO ŽIVÁ PAMĚŤ

Co se objevilo v jednom testu, je dostupné pro všechny další. Znalost neodchází s konverzací.

DETERMINISTICKÉ KOLEJE

Lint, typecheck, **sourozenecké testy**. Agent se řídí výstupem nástrojů, ne dojmem.

Vstupem je business popis. Výstupem zelený test v CI a ponaučení pro příště.

Co si z ranního bloku odnést do pondělí

- začít na malém, ale nad reálným workflow
- mít instrukce, kontext a jasného ownera
- tam, kde jsou fakta, připojit tools nebo MCP
- tam, kde je víc kroků, přemýšlet agentně

> AI není samostatná agenda. Je to nová vrstva nad každodenní prací.

DATALITE

DEN S AI | PROMĚNA ROLÍ V DOBĚ AI | Jiří Bubník

AI nemění ownera. Mění těžiště práce.

Co zůstává stejné

- **PM** je owner za dodávku a komunikaci.
- **Analytik** je owner za význam a správnost řešení.
- **Vývojář** je owner za technickou realizaci.
- **Konfigurátor** je owner za zákaznickou konfiguraci.
- **Tester** je owner za kvalitu releasu.
- **Infra team** je owner za infrastrukturu, bezpečnost a provozní základ.

CO MĚNÍ AI

- méně ručního rozepisování, přepisování a dohledávání
- více ověřování, rozhodování a skládání souvislostí
- vyšší nárok na přesah mezi rolemi
- větší hodnota u lidí, kteří umí zadat, zkontrolovat a převzít odpovědnost

DŘÍV

Cenné bylo všechno ručně vyrobit.

TEĎ

Roste cena dobrého zadání a rychlé validace.

POINTA

AI nebere ownership. Zvedá nárok na samostatnost.

DATALITE

DEN S AI | PROMĚNA ROLÍ V DOBĚ AI | Jiří Bubník

PM a analytika: delivery, význam, rozhodnutí

PM

Owner za dodávku a komunikaci

- domlouvá podmínky dodávky: nabídky, CHR, prvotní revize smluv ke schválení
- hlídá scope, termíny, priority, release plán a komunikaci se zákazníkem
- hlídá budget a plánuje kapacity
- plánuje a hlídá SWD, support režim a SLA: kdo reaguje, jak rychle, jak eskalujeme

Přesah

- technicky chápe, co se dodává
- rozumí detailním design dokumentům
- umí se zapojit do analýzy

ANALYTIK

Owner za význam a správnost řešení

- vlastní doménu, procesy, integrace, pravidla a edge-cases
- dodá specifikaci a akceptační kritéria tak, aby šlo řešení testovat a nasadit

Hlavní analytik

- hlídá konzistenci zadání napříč funkcemi a integracemi
- hlídá dopady změn
- je hlavní kontakt pro zákazníka i team ohledně technického řešení

Přesah

- umí základní konfigurace
- ověří si realitu v datech a nástrojích: DB, Elastic/Kibana, Grafana

CO S TÍM UDĚLÁ AI

DATALITE

DEN S AI | PROMĚNA ROLÍ V DOBĚ AI | Jiří Bubník

Vývoj a tech lead: implementace není jen commit

VÝVOJÁŘ

Owner za technickou realizaci

- dotahuje zadání do návrhu a implementace, rozumí business kontextu: proč, dopad, success
- dovymýšlí low-level detaily: atributy, struktury, validační pravidla, integrační chování
- low-level detaily zdokumentuje
- změna není jen commit: dostane ji na test, ověří funkčnost, řeší kompatibilitu, závislosti a kolize se souběžnými změnami
- sleduje technický stav aplikace: incidenty, diagnostika, mitigace
- pracuje i s provozními nástroji: Grafana, Kibana, K8s, …

HLAVNÍ VÝVOJÁŘ / TECH LEAD

- drží SW architekturu, standardy, code review a technický dluh
- rozděluje práci mezi vývojáře
- drží vývojové prostředí a automatizaci odevzdání
- automatizuje SWD: artefakty, verze, migrace, runbook
- odpovídá za technický stav aplikace v provozu, incidenty, zálohování, …

SPOLUPRÁCE

- rozumí business zadání
- je partner analytikovi v produktové diskuzi
- dotahuje analýzu a dokumentaci
- když je potřeba něco infra-specifického, eskaluje na Infra team

AI dopad: rychleji vznikne scaffold, návrh testů, refactoring draft nebo rozpad problému. O to víc ale roste nárok na review, kompatibilitu, provozní dopad a skutečné ověření na testu.

Konfigurace, QA, infra: release a provoz

KONFIGURÁTOR

Owner za zákaznickou konfiguraci

- připraví a udržuje konfigurace pro zákazníka: formuláře, scripty, procesy
- validuje po nasazení: smoke / sanity
- umí rychle vrátit konfiguraci zpět
- přispívá do SWD konfig balíčkem a checklistem ověření

Hlavní konfigurátor

- hlídá konzistenci konfigurací napříč prostředími a verzemi
- koordinuje závislosti a pořadí konfig změn s vývojem a QA

TESTER

Owner za kvalitu releasu

- vlastní testování a návrhy na automatizaci
- rozhoduje, co má smysl automatizovat tak, aby to dlouhodobě žilo
- do SWD dodá QA výstup: co je otestované, známá rizika, doporučení a omezení

Test lead

- připravuje testovací scénáře
- drží test strategii a release gate: co musí projít, aby to šlo ven

INFRA TEAM

Owner za infrastrukturu

- interně drží dev servery: K8s cluster, DB servery, GitLab, …
- definuje pravidla pro nomenklaturu, audit, zálohování a nastavení prostředků
- drží monitoring, alerting, bezpečnost a zálohování
- u zákazníků pomáhá tam, kde je prostředí složitější nebo nestandardní

Kam se role posouvají

INDIVIDUAL CONTRIBUTOR

- AI nejdřív zrychlí produkci prvního draftu
- rychleji vznikne návrh, checklist, test skeleton nebo shrnutí
- riziko je uvěřit hezkému draftu bez ověření reality

LEAD ROLE

- AI nejdřív zvyšuje nárok na systémové řízení kvality
- přibývá rozhodování, review, prioritizace a guardraily
- hlavní analytik, tech lead, hlavní konfigurátor i test lead musí držet celek, ne jen svůj kus práce

Jedna věta na závěr: AI pomáhá nejvíc tam, kde je hodně mechanické práce a hodně kontextu. Ale čím víc AI urychlí výrobu draftu, tím víc roste cena ownershipu, validace a schopnosti vidět dopady napříč rolemi.

Krátká interakce do sálu: Co byste dnes bez váhání delegovali AI a co byste si naopak nikdy nenechali vzít z role ownera?

Projekty

O2, CETIN, Slovanet, další plány

- stav a výhled jednotlivých projektů
- klíčové změny a rizika
- co potřebujeme od firmy a od technického směru dál

Firemní infrastruktura a její rozvoj

- co se podařilo
- co nás čeká dál
- kde jsou limity a priority
- jak do toho vstupuje automatizace a AI

Závěr: povídání o firmě

- kde jsme dnes
- kam chceme jít
- co to znamená pro projekty a lidi

Jirka Kadlec + Marek Linhart

Závěr