Engineering Leaders Community

AI SDLC PRO SAP · 10. LISTOPADU 2026

Váš návrh AI SDLC

Sestavený z vašich odpovědí: kde je váš tým dnes, co změnit jako první, jak nainstalovat nástroje a jak vaši vývojáři používají každý z 10 AI skills.

7 /10 **AI čte a kontroluje, píše člověk**

Claude Code v terminálu Eclipse, napojený na MCP server od SAP + arc1 pro čtení.

Co AI zvládne v jednotlivých krocích

Krok	Stav	Jak
Tiket v Jiře → zadání (AI ho s vámi zpřesní)	Částečně	Tiket vkládáte ručně nebo z exportu.
Zadání → IDE, AI zná váš systém	AI	AI čte živý kód, DDIC a where-used.
Nová funkcionalita	Částečně	AI navrhne kód, do SAP ho zapíše člověk.
Nové ABAP objekty (třídy, CDS, DDIC)	Částečně	AI připraví zdroják, objekt založí člověk.
Transport přes systémy	Ručně	Transporty řeší člověk.
Oprava chyby z dumpu	AI	AI přečte vložený dump a dohledá příčinu v živém kódu.
Refaktoring s testy	Částečně	AI navrhne, testy spouští člověk.
Přesun datové logiky do CDS views	Částečně	AI napíše CDS view a nový kód, v SAP je založí člověk.
Reverse engineering, hledání příčiny	AI	AI sleduje řetězec volání a where-used v systému.
Testy, code review, dokumentace	AI	AI spustí ATC a ABAP Unit a podle výsledků opraví kód.

Co změnit jako první

1. Aktualizujte ADT na 3.62 nebo novější. Od 3.62 čte a upravuje zdrojový kód pro Claude Code přímo MCP server od SAP, takže k zápisu nepotřebujete komunitní server. Aby AI mohla zapisovat, zapněte Enable Edit File tool a do seznamu balíků v Access Restrictions dejte jeden Z balík, například ZAI_*.
2. Dejte vývojářům API token do Jiry jen pro čtení. AI pak čte tikety přímo a pracuje na zadání bez kopírování.

Instalace celé sady nástrojů

Pro jednoho vývojáře, podle vašich odpovědí, v tomto pořadí.

1. Pošlete tento seznam IT

Software u vás instaluje IT. Kroky níže jsou jejich checklist, v tomto pořadí.

2. Licence Claude a Claude Code

Admin zapne Claude Code všem vývojářům v administraci Claude (Team nebo Enterprise). Každý se přihlašuje firemním účtem, nikdy soukromým: data ze soukromých účtů může Anthropic použít k trénování.

3. Eclipse 2026-09 s ADT

Stáhněte Eclipse IDE 2026-09 z eclipse.org. V Eclipse otevřete Help → Install New Software, vložte adresu níže a vyberte ABAP Development Tools. Pak File → New → ABAP Project a připojte systém DEV ze SAP Logonu.

```
https://tools.hana.ondemand.com/2026-09
```

4. Claude Code (Windows)

Nainstalujte Git for Windows, pak spusťte instalátor v PowerShellu a přihlaste se firemním účtem. Claude Code spouštějte nativně ve Windows, ne ve WSL.

```
irm https://claude.ai/install.ps1 | iex
claude
```

5. Claude Code uvnitř Eclipse

V Eclipse otevřete Window → Show View → Terminal a spusťte Claude Code tam. Kód, kontroly i AI máte v jednom okně.

6. Zapněte MCP server od SAP

Window → Preferences → ABAP Development → MCP Server → Enable ADT MCP Server. Access Restrictions nechte zapnuté a do seznamu projektů dejte jen projekt DEV. Seznam balíků nechte prázdný, AI tak jen čte. Až budete chtít zápis, přidejte jeden Z balík. Server je zdarma; podporu k jeho MCP nástrojům SAP dává jen s licencí Joule for Developers, která pro on-premise NetWeaver 7.x není v nabídce. Zkopírujte vygenerovaný token. Aby se server spouštěl s Eclipse, přidejte do eclipse.ini za -vmargs tento řádek:

```
-DadtMcpServerPrefEnabled=true
```

7. Napojte Claude Code na MCP server

V terminálu, s tokenem z předchozího kroku. Token nikdy necommitujte do repozitáře.

```
claude mcp add --transport http --scope user abap-adt http://localhost:2234/mcp --header
    "Authorization: Bearer <TOKEN>"
claude mcp list
```

8. Doplněk arc1 pro čtení kódu

ADT 3.60 a 3.61 neumí Claude Code předat zdrojový kód; SAP to přidal ve 3.62. Než aktualizujete, komunitní doplněk arc1 přidá do stejného serveru read_source, where_used, package_contents, atc_check a run_unit_tests. Do Eclipse ho nainstalujte podle github.com/arc-mcp/arc1-adt-abap-mcp-ext a jeho nástroj arc1_sap_http_post zablokujte (viz Pravidla): umí poslat požadavek na jakýkoli ADT endpoint a Access Restrictions od SAP ho nejspíš nezastaví.

9. abapGit na DEV

V SE38 založte report ZABAPGIT_STANDALONE v lokálním balíku $ABAPGIT, vložte do něj zabapgit_standalone.prog.abap z GitHubu a aktivujte ho. Export balíku: New Offline → vyberte balík → Export ZIP, pak ZIP rozbalte do týmového repozitáře.

10. Jira přes export

Uložte export z Jiry (CSV) do týmového repozitáře, AI čte soubory. Jména a ID uživatelů předtím nahraďte iniciálami.

11. Deset skills jako plugin pro Claude Code

Skills níže jsou zdarma jako otevřený plugin (github.com/marian-kamenistak/sap-ai-skills). Každý vývojář si ho v Claude Code nainstaluje dvěma příkazy. Jeho templates/CLAUDE.md zkopírujte do týmového repozitáře a doplňte release, pravidla pojmenování Z/Y a co do AI nesmí nikdy jít.

```
/plugin marketplace add marian-kamenistak/sap-ai-skills
/plugin install sap-abap@sap-ai-skills
```

12. Pravidla

Claude Code se ve výchozím stavu ptá před každým voláním MCP nástroje. U všeho, co zapisuje, to tak nechte. V .claude/settings.json týmového repozitáře zablokujte http_post z arc1 (Access Restrictions od SAP ho nejspíš nezastaví), zakládání transportů (transporty zůstávají na lidech), čtení short dumpů (dumpy můžou obsahovat osobní údaje) a tři nástroje, které spotřebovávají Joule tokeny. S licencí Joule for Developers poslední tři řádky smažte.

```
{
  "permissions": {
    "deny": [
      "mcp__abap-adt__arc1_sap_http_post",
      "mcp__abap-adt__abap_transport_create_request",
      "mcp__abap-adt__abap_feed_query",
      "mcp__abap-adt__abap_explain_object",
      "mcp__abap-adt__abap_atc_ai_*",
      "mcp__abap-adt__abap_docs_dcp"
    ]
  }
}
```

13. Ověřte, že to funguje

V Claude Code nebo Copilotu napište „vypiš moje ABAP destinace“ a „najdi třídu ZCL_... a vysvětli, co dělá“. Odpověď musí přijít z vašeho systému DEV, ne z obecných znalostí.

Jak vaši vývojáři používají jednotlivé skills

Deset skills pokrývá celý vývojový cyklus a jsou očíslované v pořadí, jak jimi změna prochází. Kroky odpovídají tomu, co AI ve vašem nastavení smí.

`spec-grill` Zadání, které vydrží vývoj

Vstup: tiket z exportu z Jiry nebo vložený text

1. AI se autora ptá po jedné otázce: obchodní cíl a komu pomůže, dotčené SAP objekty (tabulky, funkční moduly, BAdI, rozhraní), současné a cílové chování, akceptační kritéria ve tvaru given/when/then, testovací data a hraniční případy, oprávnění a osobní údaje, závislosti (rozhraní, joby, transporty) a co do rozsahu nepatří.
2. Každý zmíněný objekt ověří v systému: jestli existuje, kde se používá a jaká má pole.
3. Zadání ohodnotí 0 až 10 podle stejné tabulky. Pod 8 ho vrátí autorovi se seznamem toho, co chybí.

Rozhoduje člověk: Autor odpovídá a rozhoduje, kdy je zadání hotové.

Výstup: Zadání v Markdownu i s hodnocením, připravené k vrácení do Jiry.

`reverse-engineer` Jak to funguje a proč to padá

Vstup: otázka typu „jak funguje X“ nebo „proč se děje Y“

1. AI najde vstupní bod (transakci, report, BAdI) a sleduje řetězec volání a where-used napříč celým systémem.
2. Vysvětlí tok dat a nakreslí ho.
3. U chyby seřadí pravděpodobné příčiny, každou s důkazem z kódu.

Rozhoduje člověk: Člověk příčinu potvrdí v debuggeru.

Výstup: Vysvětlení s diagramem, podklad pro bugfix nebo refaktoring.

`implement` Nová funkcionalita od zadání po transport

Vstup: schválené zadání s hodnocením 8 a víc

1. AI čte kód přímo z DEV přes MCP. AI najde dotčená místa a where-used.
2. Navrhne dvě technické varianty s dopady a riziky. Jednu vybere člověk.
3. AI připraví kód, člověk ho vloží do ADT a aktivuje. AI pak přes MCP spustí ATC a ABAP Unit, přečte výsledky a navrhne opravy.
4. Napíše ABAP Unit testy na akceptační kritéria ze zadání.
5. Transport vytváří a uvolňuje člověk; AI napíše jeho popis.

Rozhoduje člověk: Výběr varianty, review diffu, uvolnění transportu.

Výstup: Aktivovaný kód s testy a krátká technická poznámka do Jiry.

`test-case` Testy

Vstup: třída nebo zadání s akceptačními kritérii

1. AI napíše ABAP Unit testy ve tvaru given/when/then a databázi nahradí lokálním test double za rozhraním (CL_OSQL_TEST_ENVIRONMENT potřebuje 7.51, TEST-SEAM 7.50).
2. Testy spustí přes MCP a opraví, co neprojde.
3. Napíše i scénáře pro ruční testování v QAS.

Rozhoduje člověk: Člověk ověří, že testy dokládají zadání, ne jen kód.

Výstup: Testovací třída a testovací scénáře pro QAS.

`code-review` Code review

Vstup: diff nebo exportovaný balík

1. AI spustí ATC a jeho nálezy vezme jako základ.
2. Kontroluje Clean ABAP, chybějící AUTHORITY-CHECK, SELECT ve smyčce, SELECT *, FOR ALL ENTRIES s prázdnou tabulkou, dynamické SQL, výkon na HANA a vaše pravidla pojmenování.
3. Každý nález dostane závažnost a návrh opravy.

Rozhoduje člověk: Autor rozhoduje, které nálezy opraví.

Výstup: Nálezy seřazené podle závažnosti, jako komentář k tiketu v Jiře.

`refactor` Refaktoring

Vstup: starý Z-kód

1. AI nejdřív napíše charakterizační testy, které zafixují dnešní chování.
2. Refaktoruje po malých krocích podle Clean ABAP.
3. Po každém kroku spustí testy člověk.

Rozhoduje člověk: Review diffu po každém kroku.

Výstup: Čistší kód se stejným chováním a testy, které to dokládají.

`cds-rewrite` Přesun datové logiky z ABAPu do CDS views

Proč to dělat: Joiny, filtry a součty počítá databáze místo ABAP serveru, takže se přenáší méně dat a pomalé reporty se obvykle zrychlí. Jeden CDS view pak obslouží report, Fiori aplikaci i API, nese s sebou svá pravidla oprávnění (access control) a jde stejným směrem, kterým SAP staví S/4HANA (RAP, ABAP Cloud, clean core).

Vstup: report nebo třída, která čte data přes SELECTy ve smyčkách, FOR ALL ENTRIES nebo joiny a součty skládané v ABAPu

1. AI přečte program a vypíše každý přístup k datům: každý SELECT, joiny ve smyčkách, FOR ALL ENTRIES, součty a filtry počítané v ABAPu a where-used tabulek.
2. Nejdřív napíše ABAP Unit testy, které na testovacích datech zafixují dnešní výstup (lokální test double za rozhraním), aby přepis mohl doložit stejný výsledek.
3. Navrhne CDS design: které joiny, filtry, součty a počítaná pole se přesunou do CDS view na bázi DDIC přes DEFINE VIEW (view entities potřebují 7.55), jaké asociace vystaví, co zůstane v ABAPu (zápisy, složitá procedurální pravidla) a access control, který tam, kde to sedí, nahradí roztroušené AUTHORITY-CHECKy. Design schvaluje člověk.
4. AI napíše CDS zdroják a nový kód programu, člověk je založí a aktivuje v ADT, pak AI spustí testy přes MCP a opraví, co neprojde.
5. Porovná dobu běhu před a po přes SAT nebo SQL trace v ST05 na realistických datech a uvede čísla. Bez zrychlení se nepřepisuje.
6. Transport vytváří a uvolňuje člověk; AI napíše jeho popis.

Rozhoduje člověk: Schválení CDS designu a kontrola čísel před a po.

Výstup: CDS view s access control a testy, kratší program, který z něj čte, a doba běhu před a po.

`documentation` Dokumentace

Vstup: objekt nebo balík

1. AI čte kód přímo z DEV přes MCP.
2. Popíše účel, vstupy a výstupy, použité tabulky, kdo objekt volá a co volá on.
3. Přidá sekvenční diagram v Mermaidu a známá rizika.

Rozhoduje člověk: Vlastník objektu dokumentaci potvrdí.

Výstup: Markdown pro Confluence nebo tiket v Jiře.

`transport-release` Transport přes systémy

Vstup: hotová změna v DEV

1. AI zkontroluje seznam objektů, který jí dáte.
2. Napíše popis transportu, testovací plán pro QAS a plán pro rollback.
3. Uvolnění transportu a import do QAS a PRD zůstává na lidech, podle vašeho procesu.

Rozhoduje člověk: Uvolnění transportu a import do produkce.

Výstup: Popis transportu, checklist pro QAS a PRD, plán pro rollback.

`bugfix` Oprava chyby

Vstup: bug z Jiry a text dumpu z ST22 bez osobních údajů

1. Z dumpu AI najde místo pádu a projde řetězec volání zpět ke vstupnímu bodu.
2. Popíše příčinu a ukáže řádky kódu, které ji dokládají.
3. Nejdřív napíše test, který chybu zopakuje, pak co nejmenší opravu.
4. AI připraví kód, člověk ho vloží do ADT a aktivuje. AI pak přes MCP spustí ATC a ABAP Unit, přečte výsledky a navrhne opravy.
5. Transport vytváří a uvolňuje člověk; AI napíše jeho popis.

Rozhoduje člověk: Člověk potvrdí příčinu před opravou a projde diff.

Výstup: Oprava, test, který chybu nepustí zpátky, a příčina zapsaná v Jiře.

Vaše odpovědi

Jaký AI nástroj budou vaši ABAP vývojáři používat?
Claude (Team nebo Enterprise)

Mají vaši vývojáři zapnutý Claude Code?
Ano

Kde vaši ABAP vývojáři každý den píšou kód?
Eclipse s ADT 3.60 nebo 3.61

Na jakém systému vyvíjíte?
ECC na 7.50 nebo novějším

Připojí se Eclipse k vašemu SAP systému (jsou zapnuté ADT služby)?
Ano

Jak se změny dostávají do produkce?
Transporty přes TMS (DEV → QAS → PRD)

Povolíte MCP server od SAP v Eclipse, jen proti DEV?
Ano, jen čtení a kontroly

Centrální MCP server ve vaší síti, za firewallem a dostupný přes VPN?
Ne

Postavili byste si vlastní MCP server jen s nástroji, které vyberete?
Ne

Mohou vývojáři používat open-source komunitní nástroje (abapGit, arc1, komunitní MCP servery)?
Ano

Mohou vývojáři na vývojovém systému (DEV) kopírovat kód do souborů přes abapGit?
Jen export do souborů

Jak se AI dostane k vašim tiketům v Jiře?
Jen přes export

Používáte ATC (ABAP Test Cockpit)?
Ano

Píšete ABAP Unit testy?
Občas

Co smí jít do AI?
Zdrojový kód (Z-objekty), DDIC struktury, Dumpy a logy bez osobních údajů

Na čem vývojáři pracují?
Windows

Kdo instaluje software?
IT

Test můžete kdykoli zopakovat: https://www.engineeringleaders.io/academy/cs/ai-sdlc-for-sap