reactivdevs

UKÁZKA Z KURZU · PŘIPRAVUJE SE

AI Developer

Stav produkční software s AI způsobem seniora. Kompletní osnova a ukázky ze čtyř lekcí.

@reactivdevs · reactivdevs.com

NEŽ ZAČNEME

Pro koho to je a kde navazujeme

Tenhle kurz je pro vývojáře, který už kód píše a chce s AI stavět reálný, produkční software. Ne dema, ne prototypy, které se za měsíc nedají otevřít.

Pokud jsi četl volně dostupný návod **Od vibe-codingu k produkčnímu softwaru s AI**, máš základ: kontext místo úkolu, plan mode, malé kroky, kontrola. Kurz na to navazuje a jde o dvě patra výš, do věcí, které z návodu udělat nejdou: **agenti, vlastní nástroje a reálný projekt od schématu databáze po nasazení**.

Tenhle dokument je ukázka. Dostaneš celou osnovu a čtyři lekce v plném znění, ať víš, co kupuješ.

Formát, ať víš, do čeho jdeš:

- Každá lekce je **nahraný 1:1 call.** Sdílím obrazovku a mluvím, nejsem vidět.
- Mluvím na jednoho konkrétního člověka, který mi kouká na monitor, ne do publika.
- Nezakrývám momenty, kdy model vrátí blbost a musíme couvnout. **Právě ty jsou nejcennější.**
- Vysvětluje se rozhodování, ne klikání.
- Na konci lekce úkol, na začátku další se koukáme na výsledek.

7 modulů | 35 lekcí | 12 hodin videa | AI Dev Kit v ceně

CO TĚ ČEKÁ

7 modulů od setupu po nasazení

Průběžný projekt je full-stack aplikace stavěná od specifikace po nasazení. Frontend, Node.js backend, Prisma. Jedno repo, vedené poctivě, aby byl vidět rozdíl proti splácanině.

01 **Mindset a setup.** Co je LLM a co se děje při promptu, mýty, kdy AI pomáhá a kdy škodí, instalace Claude Code, CLAUDE.md jako pravidla projektu, první pohled na AI Dev Kit.

02 **Spec a plán s AI.** Kontext místo úkolu, z nápadu ke specifikaci, plan mode, rozpad práce na kroky, prompt patterns, jak poznat slepou uličku včas.

03 **Produkční setup.** Co si necháš vygenerovat a co si pohlídáš, TypeScript a lint jako mantinely pro model, husky, CI od první minuty, anti-špageta struktura.

04 **Build a deploy end-to-end.** Prisma schéma, backend, frontend na kontraktu, autentizace, stejná feature ručně versus s AI, práce s cizí codebase, nasazení.

05 **Kvalita s AI.** Code review s AI, testy generované modelem, refaktoring bez rozbití zbytku, bezpečnost a tři úrovně přístupu, debugging, review checklist.

06 **AI Dev Kit: agenti, skills a MCP.** Průvodce hotovým repem, anatomie skillu i agenta, MCP servery, vlastní nástroj, orchestrace více agentů, hooks.

07 **Shipping a způsob práce.** AI v týmu, obhájení workflow před vedoucím, odhady, co dělat, když se model změní, akční plán na 30 dní.

LEKCE 06.2

Anatomie skillu

Skill je zabalený postup, který si model načte, až ho bude potřebovat. Nic magického: je to složka se souborem instrukcí. Hodnota není v syntaxi, ale v tom, že přestaneš pokaždé vysvětlovat to samé.

Rozdíl proti dlouhému promptu je zásadní. Prompt napíšeš jednou a zmizí. Skill je **verzovaný v repu, sdílený s týmem a spustí se sám**, když nastane situace, kterou popisuje.

SKILL Z AI DEV KITU, ŘÁDEK PO ŘÁDKU

```
--- .claude/skills/pr-review/SKILL.md ---
name: pr-review
description: Zkontroluje otevřený PR proti konvencím projektu.
  Použij, když uživatel řekne "zkontroluj PR", "review"
  nebo když je otevřený PR před mergem.

## Postup
1. `gh pr diff` a načti celý diff, ne jen změněné řádky
2. Projdi checklist z `docs/review-checklist.md`
3. U každého nálezu uveď soubor a řádek
4. Nehlas formátování ani styl, to řeší lint

## Co nikdy nedělej
- neschvaluj PR sám
- neměň kód, jen popiš nález
```

Tři věci, na kterých to stojí:

1. **Description rozhoduje o všem.** Podle ní se model rozhoduje, jestli skill vůbec použít. Musí obsahovat spouštěče, tedy slova, která reálně napíšeš.
2. **Postup, ne esej.** Číslované kroky, konkrétní příkazy, konkrétní soubory. Vágní instrukce dá vágní výsledek.
3. **Mantinely jsou důležitější než návod.** Sekce „co nikdy nedělej" je to, co odděluje užitečný skill od stroje, který ti přepíše půl repa.

> **Praktický test kvality skillu:** dej ho kolegovi místo modelu. Pokud podle něj úkol udělá stejně jako ty, je dobrý. Pokud se musí ptát, chybí ti tam kontext.

Úkol na příště:

- Najdi postup, který jsi za poslední měsíc vysvětloval modelu víc než dvakrát.
- Zabal ho do skillu podle šablony výše a vyzkoušej ho v ostrém úkolu.

LEKCE 06.3

Agent versus chat

Většina lidí používá AI jako chytřejší Google. Napíšou dotaz, dostanou odpověď, zkopírují kód. Funguje to, ale je to strop, o kterém ani nevědí.

Agent je něco jiného: dostane zadání, vlastní sadu nástrojů a **běží samostatně, dokud není hotovo**. Sám si čte soubory, sám spouští testy, sám opravuje, co rozbil. Ty výsledek kontrolujete až na konci.

CHAT SE HODÍ NA

Rozvahu, kde nevíš, co chceš. Vysvětlení cizího kódu. Rychlou otázku. Cokoli, kde chceš být u každého kroku.

AGENT SE HODÍ NA

Práci, kterou umíš zadat, ale nechce se ti ji dělat. Průzkum velkého repa. Opakovaný postup. Věci, kde jde ověřit výsledek.

Pravidlo, kdy pustit stroj samostatně:

Pusť agenta tehdy, když **dokážeš přesně popsat, jak vypadá hotovo**, a zároveň to jde ověřit bez tebe. „Přidej testy, dokud neprojde build" ověřit jde. „Zlepši architekturu" ne, a tam agenta nepouštěj.

ZADÁNÍ PRO AGENTA, KTERÉ FUNGUJE

```
# špatně: nejde ověřit, nemá hranice
Projdi projekt a zlepši ho.

# dobře: jasné hotovo, jasné mantinely
Projdi src/api/ a nahraď ruční fetch voláními
generovaného klienta.

Hotovo znamená:
- `npm run typecheck` projde
- `npm test` projde
- žádný soubor v src/api/ neobsahuje `fetch(`

Neměň nic mimo src/api/.
Když narazíš na endpoint, který v klientu není, přeskoč ho
a napiš mi ho na konci do seznamu.
```

Poslední odstavec je to, co většině lidí chybí. Bez instrukce, co dělat při nejasnosti, si model vymyslí vlastní řešení a ty ho najdeš až v code review. Vždycky mu dej cestu ven.

Úkol na příště:

- Vyber úkol, u kterého umíš napsat tři ověřitelné podmínky „hotovo".
- Pusť ho agentem a porovnej čas s tím, kolik by ti to zabralo ručně.

LEKCE 05.6

Review checklist pro kód od AI

AI generovaný kód vypadá líp, než často je. Je konzistentní, má komentáře, drží se konvencí. Právě proto se přes review protáhne snáz než kód od unaveného juniora. A přitom má jiné, mnohem zákeřnější slabiny.

Projdi tohle u každého PR, kde sahala AI:

- [ ] **Rozumím každému řádku?** Když ne, zpět. Kód, který neumíš vysvětlit, neumíš ani opravit ve tři ráno.
- [ ] **Sedí to na konvence projektu,** nebo si model zavedl vlastní vzor vedle toho existujícího?
- [ ] **Nová závislost?** Proč, jak je velká, kdy měla poslední release. Modely rády přidávají balíčky.
- [ ] **Chybové stavy.** Co se stane, když request selže, vrátí prázdno nebo timeoutne? Tohle je nejčastější díra.
- [ ] **Bezpečnost.** Validace vstupu na serveru, autorizace u každého endpointu, žádné klíče v kódu.
- [ ] **Testy testují chování,** nebo jen kopírují implementaci a projdou vždycky?
- [ ] **Nezmizelo něco potichu?** Projdi diff celého souboru, ne jen přidané řádky.
- [ ] **Hraniční případy.** Prázdný seznam, jeden prvek, tisíc prvků, diakritika, null.

> **Nejnebezpečnější není kód, který nefunguje.** Ten najdeš hned. Nebezpečný je kód, který funguje na tvých datech a spadne na produkčních, protože model netušil, že to pole může být prázdné.

V kurzu tenhle checklist zabalíme do skillu, takže první kolo review projede stroj a ty dostaneš seznam nálezů s odkazy na řádky. Druhé kolo ale pořád děláš ty. **Odpovědnost se delegovat nedá.**

LEKCE 04.5

Stejná feature dvakrát

Tohle je lekce, kterou v kurzu nejvíc obhajuju, protože zabere dvakrát tolik času než ostatní. A přesto ji tam nechávám, protože nic jiného rozdíl neukáže tak přesvědčivě.

Vezmeme jednu konkrétní feature, filtrování a stránkování seznamu s uložením do URL. Postavíme ji dvakrát. Jednou ručně, tak jak jsem ji psal posledních deset let. Podruhé s AI podle workflow z modulů 02 a 03. Měříme čas a pak srovnáváme výsledek.

Co v tom srovnání uvidíš:

- **Kde AI vyhraje na hlavu.** Boilerplate, typy, testy, opakující se kód. Tady není o čem debatovat.
- **Kde tě zdrží.** Rozhodnutí, kde je potřeba znát kontext projektu, který v repu není napsaný.
- **Kde ti podstrčí něco, co vypadá dobře.** A jak to poznat dřív než v produkci.
- **Kolik času sežere psaní zadání** a od jaké velikosti úkolu se to začne vyplácet.

Nechci ti prodat, že AI je vždycky rychlejší. Není. U malé změny, kterou máš v hlavě, ji nepotřebuješ. Celý smysl téhle lekce je, abys po ní poznal, kdy sáhnout po čem. To rozhodnutí je ta skutečná dovednost.

Výsledek obou verzí dostaneš jako kód v repu. Můžeš si je otevřít vedle sebe a projít diff sám.

KURZ

Co dostaneš

- **35 video lekcí** ve formátu nahraného 1:1 callu, 7 modulů, zhruba 12 hodin.
- **AI Dev Kit:** hotové repo se skills, agenty, hooks, nastavením a CLAUDE.md. Nasadíš ho hned v prvním modulu a odneseš si ho na vlastní projekty.
- **Knihovna prompt patterns** a šablon.
- **Kompletní zdrojový kód** referenční full-stack aplikace.
- **Review checklist** pro AI generovaný kód.
- **Doživotní přístup** a aktualizace obsahu.

~~8 900 Kč~~ **5 900 Kč**

Zaváděcí cena. Jednorázová platba, žádné předplatné, 14denní garance vrácení peněz.

Na kurzu se pracuje. Osnova je hotová a natáčím lekce, prodej otevřu hned po dokončení. Stav sleduj na reactivdevs.com/course/ai-developer. Mezitím je obsahově hotový kurz Production-ready Developer, u kterého se dá zapsat na čekací listinu.

Jak to spolu souvisí:

Production-ready Řemeslo: git, testy, architektura, CI, obhájení rozhodnutí. Bez toho ti AI jen rychleji vyrobí nepořádek.

AI Developer Tohle řemeslo dělané násobně rychleji s AI, agenty a vlastními nástroji.

Mentoring 1:1 vedení přímo na tvém projektu.

Otázka není, jestli tě AI nahradí. Je, jestli budeš ten, kdo ji řídí.