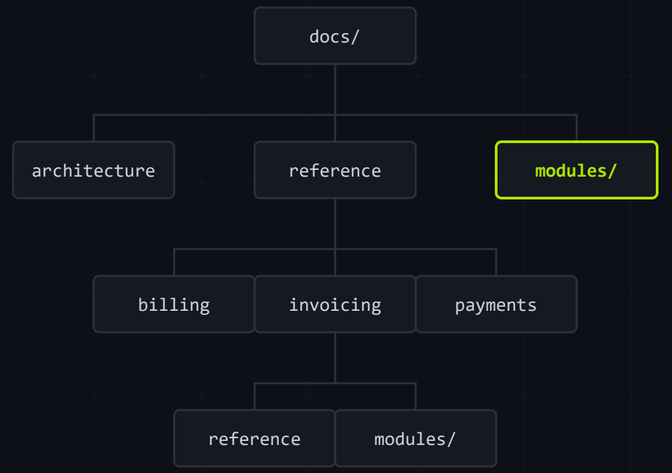
VIBECODING TALKS · 20. 4. 2026

Strukturace dokumentace pro efektivní použití s Claude Code

Od plochého `docs/` *k substrátu, který škáluje.*



Ondřej Tučný

VYTVOŘENO V CLAUDE DESIGN RESEARCH PREVIEW

O čem to bude (a o čem ne)

CO SE DOZVÍTE

- Proč dokumentace pro Claude Code potřebuje **strukturu**, ne jen obsah
- Jak oddělit **AS IS** stav systému od **TO BE** změn
- Jak škálovat **rekurzivně** — stejný vzor pro 10 kLOC i 1 MLOC
- Jak dělat dokumentaci, **kterou nemusíte udržovat ručně**

O ČEM TO NEBUDE

- *„Napište lepší CLAUDE.md“* — to už umíte
- Hotová šablona ke stažení — zásady jsou univerzální, implementace závisí na vašem stacku

Výchozí bod: co už víme

Článek Patricka Zandla na vibecoding.cz shrnul solidní základ — pravda, do určité velikosti projektu.

FUNGUJE

- Kontext na vyžádání
- `CLAUDE.md` = odkazy, ne obsah
- Tematické soubory dokumentace
- `gotchas.md` jako zápisník
- Aktualizace po každém úkolu

KDE TO PŘESTÁVÁ STAČIT

- Plochá `docs/` — stovky souborů na jedné úrovni
- `gotchas.md` jako jeden soubor = **entropická past**
- Specifikace v `docs/` míchá pending změny s aktuálním stavem
- „Aktualizuj po každém úkolu" = **naděje, ne proces**
- Žádná historie rozhodování

AS IS vs. TO BE

Dokumentace má dva úplně jiné životní cykly a nepatří do jedné složky.

AS IS **docs/**	TO BE **tasks/<EPIC>/spec/**
Popisuje, co systém **je** dnes	Popisuje, co se má **stát** v konkrétním epiku
Aktualizuje se **při mergi úkolu nebo epiky**	Vzniká **před implementací**, migruje do AS IS po dokončení
Čte se při plánování — *„jak to funguje"*	Čte se při plánování — *„co mám udělat"*
Jeden strom pro **celý systém**	Jeden strom pro **každý epik**, paralelně

Praktický dopad: agent čte TO BE jako *zadání* a AS IS jako *kontext*. Po mergi se TO BE promítne do AS IS.

Rekurzivní struktura

Stejný vzor na všech úrovních: **systém → modul → podmoduly**.

docs/ — struktura

```
docs/
├── architecture/              # Systémová architektura
├── reference/                 # Funkční referenční popis
├── modules/
│   ├── billing/
│   │   ├── README.md          # Konceptuální přehled
│   │   ├── architecture/
│   │   └── modules/             # Rekurze
│   └── invoicing/
└── …
```

PROČ TO FUNGUJE

- **Malý projekt:** použijete jen kořen. Modulů je málo nebo žádné.
- **Velký projekt:** modulů je víc — ale každý má *stejnou strukturu*.
- **Claude se neučí.** Zná jeden vzor a aplikuje ho v jakékoliv hloubce.

PRINCIP

Partial population. Substrát je *superset*. Malý repozitář instancuje jen část.

Dokumentace u kódu (ne místo něj)

Dokumentace patří i ke zdrojovému kódu — usnadňuje orientaci v kódu.

V KAŽDÉ SLOŽCE SE ZDROJÁKY

DESCRIPTION.md

Co tady žije, jaké API poskytujeme, jak se to používá. Popis *namespace* (C#), *modulu* (Python/Rust), *package* (TypeScript/Node).

V ADRESÁŘI ASSEMBLY / CRATE / BALÍČKU

MODULES.md

Auto-generovaný přehled: *„co je v tomhle modulu za namespaces"*. Pro agenta rychlý scan před zanořením do detailů.

PROČ TO MÁ SMYSL

- Claude edituje soubor → **vyjde nahoru** k dokumentaci
- Dokumentace zmíní funkci → Claude najde **zdroj**

ÚDRŽBA

Dokumentace patří i ke zdrojovému kódu — usnadňuje orientaci v kódu.

`api_hash:` `a7f3c9…`

ReadTheDocs — custom MCP

@odkazy v CLAUDE.md — ve velkém repu nepoužitelné. Dej konvenci a nástroj.

mcp · ReadTheDocs

```
ReadTheDocs({ topic: "architecture",
              module: "billing" })

ReadTheDocs({ forCode: "src/Billing/Invoicing" })
  // co dokumentuje tenhle kód

ReadTheDocs({ concept: "idempotence" })
  // najdi napříč

ReadTheDocs({ adr: "list", topics: ["auth"] })
  // ADR filtrované podle tématu
```

CLAUDE.MD SE SCVRKNE

Místo dvaceti @odkazů jediná věta: „Pro dokumentaci použij `ReadTheDocs`*."*

01 **Bez cest** — Claude zná jen konvenci.

02 **Bez kolizí** — stejné jméno souboru existuje v každém modulu.

03 **Šetří kontext** — agent dostane jen podstatné.

Reasoning, ne jen kód

Každý úkol má svoji složku — `tasks/<EPIC-KEY>/<TASK-KEY>/`.

tasks/ — struktura úkolu

```
tasks/CF-100-EPIC/
├── README.md                # Scope epiku
├── spec/                    # TO BE specifikace
├── CF-142/
│   ├── assignment.md        # Zadání — co a proč
│   ├── plan.md              # Plán — jak
│   └── changelog.md         # Co se skutečně udělalo
└── CF-143/
    └── …
```

PROČ TO MÁ SMYSL

- Multi-session vývoj — **každá session začíná bez paměti**. Plán + changelog v repu řeknou, kde skončilo.
- Rozhodnutí a důvody **se neztratí** v commit messages, ani nepřekáží.
- Když projekt dělá víc lidí / agentů, **každý vidí historii**.

DŮLEŽITÉ PRAVIDLO

`tasks/` je **cold storage**. Claude standardně nepoužívá — jen na vyžádání, plus vede dokumentaci pro aktuální task.

Substrát, ne metodika

Poslední a možná nejdůležitější princip. Docs-process je substrát pro konkrétní metodiky.

SUBSTRÁT

Stabilní základ

- **Kde co leží** — `docs/`, `tasks/`, `src/`
- **Kdy se co aktualizuje** — cadence
- **Jak se to prohledává** — `ReadTheDocs` kontrakt
- **Základní frontmatter** — schéma

METODIKY

Specializace nad substrátem

`ANALÝZA` docs/catalog/ — taxonomie

`UX` docs/ux/ — journey mapy

`DEVOPS` runbook šablony

Metodiky substrát **nepřepisují**. *Jen rozšiřují. Dvě mohou běžet v jednom repu bez konfliktu.*

PŘENOSITELNÝ

Claude chápe každý repozitář stejně.

NEZÁVISLÝ VÝVOJ

Metodiky se sdílí mezi týmy.

ROZŠIŘITELNÝ

Nová role = nová metodika. Substrát se nemění.

~/projects/cashflow · claude-code

```
$ claude

> Jak se v tomhle repu fakturuje?

◦ calling tool
  ReadTheDocs({ concept: "fakturace" })

◦ result
  docs/modules/billing/README.md
  docs/modules/billing/architecture/overview.md
  src/Billing/Invoicing/DESCRIPTION.md

> Modul Billing generuje faktury
  přes InvoiceGenerator (DESCRIPTION.md).
  Idempotence je v ADR-014…

_
```

CO UVIDÍTE

- Strukturu složek a frontmatter
- `ReadTheDocs` volání v praxi
- Jak agent naviguje mezi kódem a dokumentací
- AS IS → TO BE promítnutí po mergi

Když nic jiného, tohle

01 **Oddělte AS IS a TO BE.** Jedna změna struktury, která řeší půlku problémů se stálostí dokumentace.

02 **Nedávejte Claudeovi cesty. Dejte mu konvenci a nástroj.** `ReadTheDocs` nebo ekvivalent. CLAUDE.md se zkrátí o polovinu.

03 **Substrát, ne rovnou metodika.** Začněte obecně — struktura, stavy, frontmatter. Metodika přijde na druhý krok.

DĚKUJI

Ondřej Tučný

Celá specifikace docs-process je veřejná — odkazy v poznámkách.

boldbrick