Univerzita Jana Evangelisty Purkyně v Ústí nad Labem (UJEP)

**Generování REST API specifikace pomocí LLM**

Seminární práce (KI/SWI)

Ondřej Švorc (F23209)

16. 5. 2025

## Úvod

Tato práce zkoumá, zdali je LLM o4-mini-high[[1]](#footnote-1) (dále jen LLM) schopen extrahovat ze vstupu (přirozený jazyk) strukturu REST API (endpointy, metody, parametry) a převést ji do OpenAPI dokumentace (JSON) dodržující OpenAPI 3.0 specifikaci[[2]](#footnote-2). Dále se zaměřuje na schopnost LLM generovat REST API v C# .NET 8 ASP.NET Web API (Minimal APIs) na základě dané dokumentace celkem pro 3 různé business scénáře. Nakonec se pak věnuje generování návrhů na zlepšení (dokumentace, REST API).

V běžné praxi je OpenAPI dokumentace často implicitně generována zpětně z existujícího REST API kódu, typicky pomocí nástrojů jako Swagger[[3]](#footnote-3). Slouží jednak jako dokumentační rozhraní, ale také pro automatické generování klientského kódu pro komunikaci s daným REST API. Cílem je zjistit, zda opačný přístup, tedy generování samotného návrhu a implementace REST API z OpenAPI dokumentace vygenerované pomocí LLM, může být z pohledu času rychlejší na realizaci.

Téma práce považuji za vysoce aktuální. Používání velkých jazykových modelů razantně mění způsob, jakým přemýšlíme o softwarovém návrhu – včetně návrhu REST API. Co bývalo introspektivním monologem vývojáře, případně kolektivním rozhodováním v týmu, se častokrát mění v dialog s LLM. Domnívám se, že role vývojáře jako jednotlivce se bude postupně posouvat od samotného psaní kódu k formulaci jeho záměru, a to vše pro následnou interpretaci LLM. Vývojář nebude primárně autorem konkrétních implementací, ale průvodcem v procesu iterativní elicitace s LLM, jejímž cílem bude výstup, se kterým bude subjektivně spokojen.

## Výzkumná rešerše

Kamali, M. Evaluating GPT-4 for OpenAPI Generation and Conversion. Bachelor’s Thesis. Uppsala University, 2024. Dostupné z: <https://www.diva-portal.org/smash/get/diva2:1877570/FULLTEXT01.pdf>

Yang, J. et al. CodeSensei: Teaching Large Language Models to Generate and Validate API Specifications. arXiv preprint arXiv:2402.11625, 2024. Dostupné z: <https://arxiv.org/pdf/2402.11625v1>

## Výzkumné otázky

1. Jak lze automaticky vytvářet OpenAPI dokumentaci ve formátu JSON k REST API, který je popsán přirozeným jazykem, pomocí LLM o4-mini-high?

* Validita výstupu
  + výstupem je JSON (ano/ne)
  + JSON je well-formed (ano/ne)
  + JSON odpovídá OpenAPI 3.0 specifikaci (ano/ne)
* Počet endpointů se správnými parametry a návratovými typy / celkový počet očekávaných
* Subjektivní skóre (0-5, 0 = nejhorší, 5 = nejlepší) hodnotící to, jak dobře výstup odpovídá zadání

1. Jak lze generovat návrhy na zlepšení OpenAPI dokumentace ve formátu JSON pomocí LLM o4-mini-high?

* Výstup zůstává validní podle OpenAPI 3.0 specifikace (ano/ne)
* Počet nově doplněných prvků (např. summary, examples, contstraints)
* Subjektivní skóre (0-5, 0 = nejhorší, 5 = nejlepší) hodnotící, jak je výsledek přínosný

1. Jak lze generovat REST API v C# .NET 8 ASP.NET Web API (Minimal APIs) na základě OpenAPI dokumentace ve formátu JSON pomocí LLM o4-mini-high?

* Validita výstupu
  + Výstupem je C# kód
  + Kód je po vložení do Program.cs kompilovatelný
* Počet endpointů se správnými parametry a návratovými typy / celkový počet očekávaných
* Subjektivní skóre (0-5, 0 = nejhorší, 5 = nejlepší) hodnotící čitelnost a využitelnost kódu jako základní implementace

## Návrh experimentů

Metodologie této práce spočívá v empirickém testování schopností LLM ve třech oblastech odpovídajících výzkumným otázkám. LLM je dotazován výhradně v angličtině. Pro každý úkol jsou vytvořeny samostatné promptovací šablony. Model nebyl žádným způsobem upravován ani laděn. Pro validaci JSON výstupu je používán Swagger Editor[[4]](#footnote-4).

|  |  |
| --- | --- |
| LLM | o4-mini-high |
| Jazyk promptu | angličtina |

## Výsledky a diskuze

Byly provedeny celkem 3 experimenty, jejíž výsledky považuji za nutné interpretovat. Podrobné vstupy a výstupy jednotlivých experimentů jsou uvedeny v přílohách.

### Experiment 1

První experiment spočíval v generování OpenAPI dokumentace z přirozeného jazyka.

|  |  |
| --- | --- |
| Výstupem je JSON | ano |
| JSON je well-formed | ano |
| JSON odpovídá OpenAPI 3.0 specifikaci | ano |
| Počet správných endpointů / počet očekávaných | 6/6 |
| Subjektivní skóre | 3/5 |

Výstup LLM v experimentu 1 je sice technicky korektní, dobře strukturovaný a validní podle OpenAPI 3.0 specifikace, ale postrádá jemnější návrhové detaily. Model úspěšně vytvořil základní CRUD operace pro entitu Book i její závislé entity Author a Genre, včetně filtrování podle authorId a genreId. Chybí ale například stránkování (page, pageSize), typické pro produkční API. Schémata sice odpovídají základnímu zadání, ale nejsou rozšířená o validační pravidla (minLength, enum, pattern). Výstup neobsahuje ani jednotný model pro chybové odpovědi, což je běžný best practice. Na druhou stranu potěší oddělení BookInput a Book, což ukazuje na snahu o čistý návrh, i když volba názvu BookInput mi přijde spíše nešťastná. Celkově výstup dobře reaguje na prompt, ale neprojevuje žádnou vlastní iniciativu, resp. vyšší úroveň návrhové inteligence. Hodí se jako výchozí bod, ale bez zásahu vývojáře není okamžitě připraven k produkci.

Použitý prompt a výstup LLM lze najít v Příloze 1.

### Experiment 2

Druhý experiment spočíval v generování vylepšené verze OpenAPI dokumentace ve formátu JSON, která byla výstupem prvního experimentu.

|  |  |
| --- | --- |
| Výstup zůstává validní dle OpenAPI 3.0 specifikace | ano |
| Počet nově doplněných prvků | 5 |
| Subjektivní skóre | 4/5 |

Výstup LLM v experimentu 2 představuje kvalitativní posun oproti prvnímu – struktura zůstává zachována, ale dokumentace je obohacena o summary, description a example bloky. To zlepšuje development experience v nástrojích typu Swagger UI. Model správně doplnil i validační constraints (např. minLength, maxLength, format), což původní verze postrádala. Vzniklo také nové jednotné Error schéma. Přesto se mi některé části jeví jako zbytečné – například example sekce jsou rozsáhlé a opakují se. Dokumentace by šla ještě dále optimalizovat pomocí $ref nebo externalizací opakujících se struktur. Celkově však výstup výrazně zvyšuje úroveň dokumentace, odpovídá profesionálním standardům a představuje dobrý výchozí bod pro veřejně publikovatelné API.

Použitý prompt a výstup LLM lze najít v Příloze 2.

### Experiment 3

Třetí experiment spočíval v generování REST API endpointů z OpenAPI dokumentace ve formátu JSON, která byla výstupem prvního druhého experimentu, resp. mohlo se jednat o výstup prvního experimentu, ale byl zvolen výstup druhého, protože má být lepší verzí toho prvního.

|  |  |
| --- | --- |
| Výstupem je C# kód | ano |
| Kód je po vložení do Program.cs kompilovatelný | ano |
| Počet správných endpointů / počet očekávaných | 7/7 |
| Subjektivní skóre | 4,5/5 |

Výstup třetího experimentu představuje překvapivě vyspělý příklad generace C# .NET 8 Minimal API kódu z OpenAPI dokumentace. Model dokázal vytvořit funkční, kompilovatelný kód s použitím Entity Framework Core[[5]](#footnote-5), včetně správné definice DbContext, tříd pro entitní modely a vztahů (many-to-many). Na rozdíl od předchozích experimentů je zde výstup výrazně blíže produkčnímu využití – nechybí dependency injection, Include(...) pro eager loading vztahů a idiomatická práce s DbSet<T>.

Přes to všechno je třeba zmínit několik slabin. Chybí validace vstupních dat ([Required], [MaxLength], fluent API konfigurace), chybí transakční zajištění, a konstrukce výsledků je zjednodušená (např. chybí DTO pro odpovědi). Kód neřeší edge-cases (např. neexistující AuthorId při POST/PUT), ani optimalizaci na úrovni ORM (např. AsNoTracking). Přesto – na čistě generovaný výstup bez přímé zpětné vazby – jde o velmi použitelný základ, který může vývojář dále rozšířit.

Použitý prompt a výstup LLM lze najít v Příloze 3.

## Závěr

LLM o4-mini-high prokázal schopnost generovat validní i použitelnou OpenAPI dokumentaci na základě přirozeného jazyka. Také ukázal svůj potenciál pro tvorbu základního C# API kódu dle této dokumentace. Výsledky naznačují, že opačný proces (od dokumentace ke kódu) může být vhodný v počátečních fázích návrhu.

Omezením bylo, že LLM nerozpoznával implicitní požadavky, které si vývojář osvojí praxí a myslí na ně. V budoucnu by bylo zajímavé testovat kombinaci více modelů (např. o4-mini-high vs GPT-4) nebo srovnat výstupy s reálnými návrhy vývojářů různé seniority.

## Zdroje

xxx

1. OPENAI. Introducing OpenAI o3 and o4-mini [online]. 16. dubna 2025 [cit. 12. května 2025]. Dostupné z: <https://openai.com/index/introducing-o3-and-o4-mini/> [↑](#footnote-ref-1)
2. OpenAPI Initiative. OpenAPI Specification v3.0.0. [online]. 2020 [cit. 2025-05-15]. Dostupné z: <https://spec.openapis.org/oas/v3.0.0> [↑](#footnote-ref-2)
3. SmartBear Software. Swagger – Open Source API Tool. [online]. [cit. 2025-05-15]. Dostupné z: <https://swagger.io/> [↑](#footnote-ref-3)
4. SMARTBEAR Software. Swagger Editor [online]. [cit. 14. 5. 2025]. Dostupné z: <https://editor.swagger.io/> [↑](#footnote-ref-4)
5. Microsoft. Entity Framework Core. [online]. [cit. 2025-05-15]. Dostupné z: <https://learn.microsoft.com/en-us/ef/core/> [↑](#footnote-ref-5)