

Sem vložte zadání Vaší práce.



**FAKULTA
INFORMAČNÍCH
TECHNologiÍ
ČVUT V PRAZE**

Bakalářská práce

Věnná města českých královen - API pro předávání grafických modelů

Martin Čapek

Katedra softwarového inženýrství

Vedoucí práce: Jiří Chludil

2. dubna 2019

Poděkování

Děkuji především své rodině za podporu, svému vedoucímu za časté konzultace a za směřování mě k práci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 2. dubna 2019

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2019 Martin Čapek. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Čapek, Martin. *Věnná města českých královen - API pro předávání grafických modelů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Práce je věnována analýze funkčním a nefunkčním požadavkům editoru Virtuálního historického průvodce, který má zjednodušit práci odborníkům (zvláště historikům).

Hlavním přínosem této práce není vytvořit plně funkční REST API, ač je implementován funkční prototyp, ale sestavit obsáhlý manuál, základní kámen činnosti pro následující generace řešitelů projektu.

Klíčová slova Node.js, RestAPI, PostgreSQL, virtuální realita, historické modely budov

Abstract

abstractEN

Keywords Nahraďte seznamem klíčových slov v angličtině oddělených čárkou.

Obsah

Úvod	1
1 Cíl práce	3
2 Analýza	5
2.1 Požadavky editoru virtuální reality	5
2.2 Použité technologie	6
2.3 Nástroje pro návrh REST API	9
2.4 Výběr nástroje pro návrh a dokumentaci API	10
3 Návrh	13
4 Vývojářské prostředí pro OS Linux	15
4.1 Příprava vývojářského prostředí	15
4.2 Verzování	15
5 Implementace	17
6 Testování	19
Závěr	21
A Seznam použitých zkratk	23
B Obsah příloženého CD	25

Seznam obrázků

2.1	JWT	7
2.2	Proces autorizace	8

Seznam tabulek

3.1	Pokrytí funkčních požadavků	13
-----	---------------------------------------	----

Úvod

Věnná města českých královen je rozsáhlý projekt, jehož cílem je vytvoření specializovaného historického průvodce věnnými městy a jejich městskou krajinou. Momentálně se v tomto projektu vyvíjí editor virtuální reality, který má sloužit jako nástroj historikům a jiným odborníkům, ve kterém budou moci spolupracovat na vytváření modelů, zasazování modelů do krajiny a tím vytvářet historická věnná města. Účelem editoru bude sloužit i návštěvníkům si tato města prohlížet.

V mojí práci se věnuji analýze požadavků editoru virtuální reality, návrhem API pro editor, implementací jeho prototypu a otestováním prototypu.

Uživatelé v editoru budou potřebovat API pro práci s velkým množstvím informací a grafických modelů, uložených v databázi. API bude také potřeba k řešení přístupových práv, protože některé informace jsou citlivé a s některými modely mohou manipulovat jen určití lidé. Výsledek této bakalářské práce bude vzorem pro vývoj API v rámci projektu Věnná města českých královen.

Cíl práce

Cílem teoretické části práce je analyzovat funkční a nefunkční požadavky editoru virtuální reality. Při analýze se zaměřit na datové úložiště a způsob propagace modelů. Dále provést analýzu nástrojů pro návrh a dokumentaci API a z těchto nástrojů vybrat nejvhodnější a s jeho použitím navrhnout prototyp API, které umožní komunikaci mezi datovým úložištěm a editorem virtuální reality. Dalším cílem práce je popsat vývojové prostředí, které využiji v implementaci. V popisu budou použité technologie, příprava vývojářského prostředí a doporučený způsob verzování zdrojových kódů.

Cílem praktické části práce je implementace a otestování prototypu REST API za použití technologie Node.js a využitím technologie continuous integration.

Analýza

2.1 Požadavky editoru virtuální reality

2.1.1 Funkční

Následující sekce popisuje funkční požadavky editoru virtuální reality. Tedy požadavky, které se vztahují k funkcionalitě na cílovou aplikaci. Tyto požadavky jsem vytvořil s pomocí Patrika Křepinského.

- F1

2.1.2 Nefunkční

Tato sekce se zabývá nefunkčními požadavky na REST API. Tedy požadavky, které se zaměřují na nároky cílové aplikace na software a to například z hlediska bezpečnosti, spolehlivosti, či výkonu.

- Rychlost odezvy

Uživatel nesmí na obdržení nebo aktualizování grafického modelu čekat. Je třeba, aby server reagoval rychle.

- Datová nenáročnost při komunikaci

Je třeba minimalizovat množství dat posílané přes API, abychom docílili rychlosti a zbytečně nepřetěžovali spojení s koncovým uživatelem.

- Bezpečnost

Určité koncové body vyžadují oprávnění, jelikož jejich zavoláním se předávají nebo přepisují citlivá data. To se vyřeší tím, že při přihlášení dostane uživatel token, kterým se bude autorizovat u volání citlivých koncových bodů.

2. ANALÝZA

- Rozšiřitelnost

API musí umožňovat případné rozšíření o další funkcionality. Také musí být řádně zdokumentováno, aby umožnilo hladší průběh rozšiřování.

- Použité technologie

Bylo zadáno, že se bude vyvíjet v technologii Node.js za využití modulu Express.

2.2 Použité technologie

2.2.1 Node.js

Node.js je open-source multiplatformní JavaScriptové prostředí postaveno na Chrome V8 JavaScript enginu. Primární účel Node.js je tvorba serverové části webových aplikací, které vychází z paradigmatu "JavaScript everywhere".

Node.js využívá událostmi řízenou architekturu a neblokující I/O operace. Tento návrh optimalizuje výkon a škálovatelnost programů s častými požadavky na I/O operace.

Jádro celého Node.js tvoří smyčka událostí, která běží na jednom vlákně. Ta podporuje desetitisíce souběžných připojení bez nutnosti neustálého přepínání kontextu díky neblokujícímu I/O. Nedochází zde k žádnému zamykání, tudíž nemusíme mít obavy z deadlocku systému.

2.2.2 npm

Node.js využívá správce balíčků npm, jehož pomocí můžeme obecně instalovat i spravovat závislosti a spouštět skripty. Npm se chlubí tím, že je největším balíčkovacím správcem a aktuálně obsahuje skoro 800 000 balíčků. - zdroj <http://www.modulecounts.com>.

2.2.3 Express

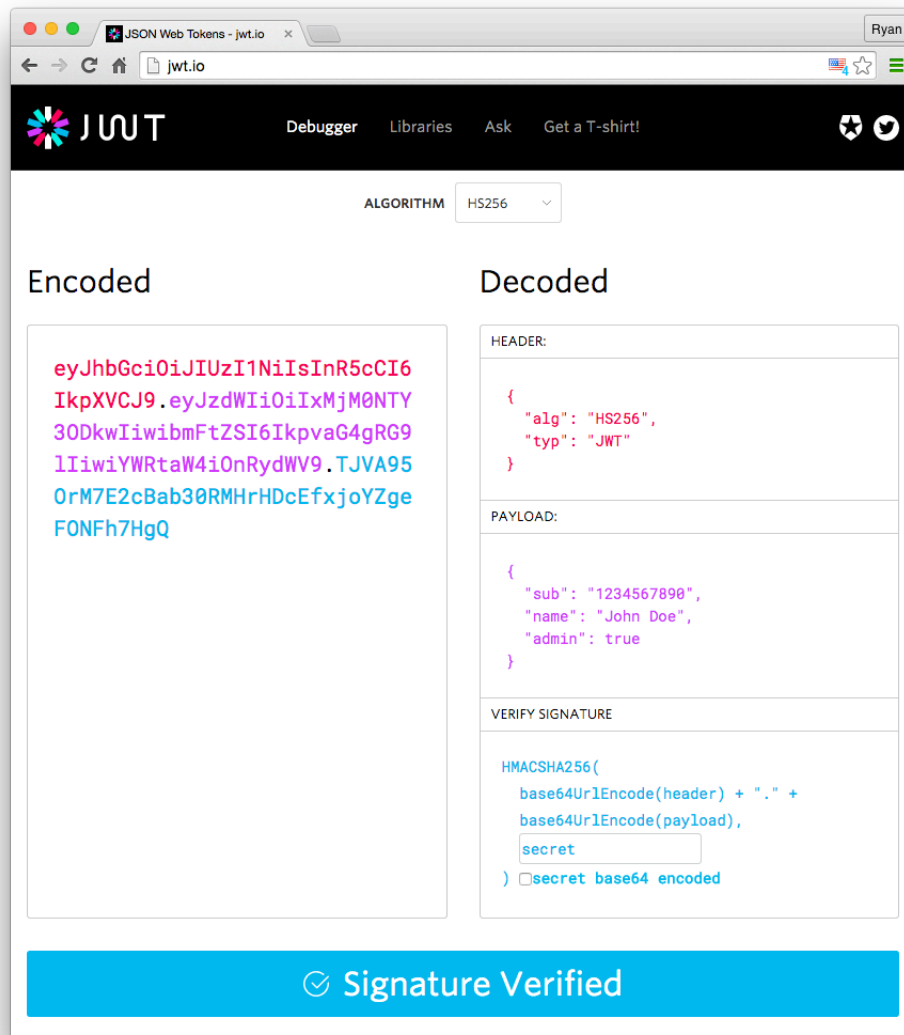
Express je framework pro Node.js, který nám umožňuje jednoduše napsat webovou aplikaci, či API. Těší se velké popularitě, momentálně na 4. místě npm rank¹.

2.2.4 JWT

JWT (JSON Web Token) je standart, který zajišťuje bezpečný přesun informací zapsaných v JSON.

JWT se skládá ze 3 částí:

¹<https://gist.github.com/anvaka/8e8fa57c7ee1350e3491>



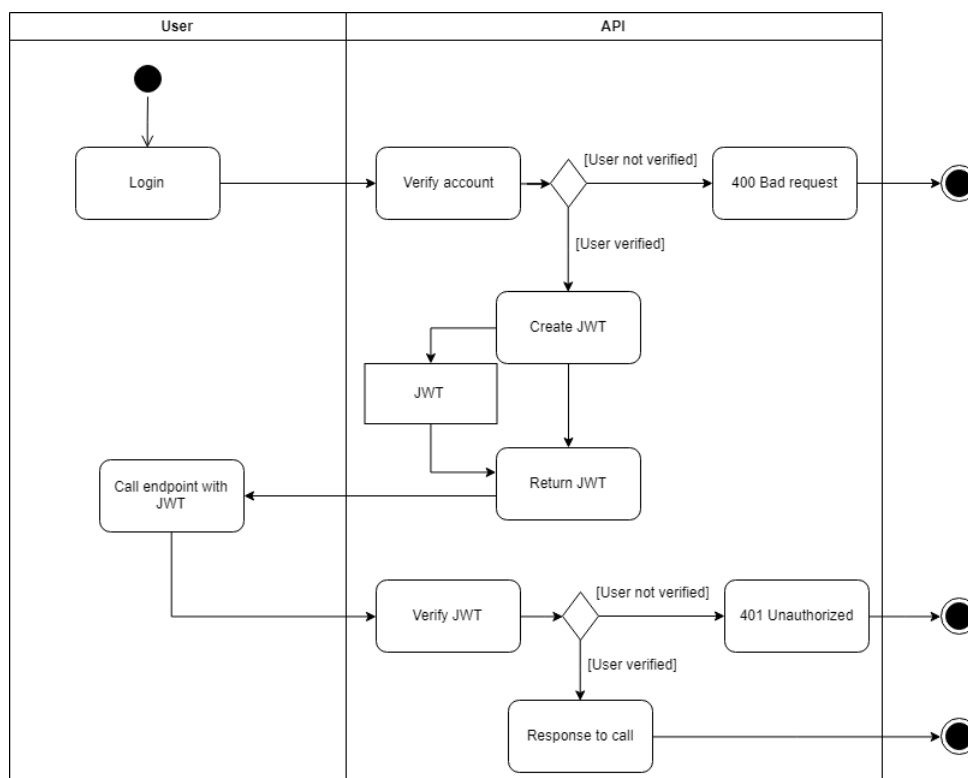
Obrázek 2.1: JWT

- Header
obsahuje typ tokenu a šifrovací algoritmus.
- Payload
obsahuje naše informace a může obsahovat ještě dodatečné informace jako expirace tokenu, vydavatel atd.
- Signature

2. ANALÝZA

obsahuje zakódované předešlé části a secret – náš tajný klíč.

V naší aplikaci využijeme tuto technologii pro ověřování totožnosti uživatelů. Naše aplikace vytvoří JWT po přihlášení a uživatel se jím bude nadále identifikovat



Obrázek 2.2: Proces autorizace

2.2.5 Implementační jazyk

Pro vývoj byl vybrán jazyk TypeScript, který je kompilovatelný do JavaScriptu. Tento jazyk je vyvíjen firmou Microsoft a jeho hlavní myšlenkou je být nadstavbou JavaScriptu, která přidává statické typování a další funkcionalitu.

2.2.5.1 TSLint

TSLint je nástroj pro statickou analýzu kódu TypeScriptu.

2.2.6 PostgreSQL

Pro datovou vrstvu byla vybrána databáze PostgreSQL.

PostgreSQL je objektově-relační databázový systém pod MIT licenci. Je pověstný svou spolehlivostí a vysokou bezpečností. Na

2.2.7 MVC

2.3 Nástroje pro návrh REST API

2.3.1 Swagger

Swagger je sada nástrojů postavená na OpenAPI Specifikaci. Nástroje:

- Swagger editor - webový editor umožňující psát OpenAPI Specifikace
- Swagger UI - vizualizuje API dokumentaci, kterou generuje z OpenAPI Specifikace
- Swagger Codegen - vygeneruje kód na základě OpenAPI Specifikace

OpenAPI Specifikace, dříveji Swagger Specifikace, je sada pravidel, které sémanticky popisují API. Pravidla mohou být zapsaná nebo vygenerována do souboru formátu YAML, nebo JSON. YAML je více čitelný pro lidi, kteří nejsou zvyklí na závorky. Programátorům může více vyhovovat JSON. OpenAPI Specifikace se skládá z metadat, koncových bodů a schématu.

2.3.2 Apicurio Studio

Apicurio Studio je open-source editor pro OpenAPI specifikaci. Musíte se přihlásit a vaše návrhy REST API se ukládají do vašeho repozitáře na GitHubu, GitLabu nebo Bitbucketu, což se hodí v případě, když na návrhu pracujete společně v týmu. V editoru můžete přepínat mezi interaktivní vizualizací návrhu a jeho JSON/YAML definicí.

2.3.3 RAML

RAML je akronym RESTful API Modeling Language a je to typ souboru podobný YAML. Existují různé nástroje používající RAML, vypíši některé z těch nástrojů: API Workbench je IDE pro RAML, které je v beta verzi a funguje jako balíček pro Atom, což je textový editor. Raml2html dokáže z RAML vytvořit HTML dokumentaci. Osprey dokáže z RAML vygenerovat API pro Node.js.

2.3.4 Restlet

Restlet Studio je webová aplikace, která umožňuje uživateli interaktivně upravovat REST API. Doáže definované API exportovat do OpenAPI specifikace nebo RAML. Dokáže vytvořit Server Skeleton pro Node.js.

2.3.5 Apiary

Api editor podporuje API Blueprint i OpenAPI Specifikaci.

API Blueprint je podobně jako OpenAPI Specifikace sada pravidel popisující API. API Blueprint specifikace je zapsaná v Markdown dokumentu, který je rozdělen do sekcí. Tyto sekce nejsou povinné.

Výpis základních 5 sekcí, u každé je číslo, které vyjadřuje kolikrát se sekce může v dokumentu objevit.

- 0-1 Metadata
- 0-1 API Name & overview
- 0+ Resource
- 0+ Resource Group
- 0+ Data Structures

Apiary disponuje interaktivní dokumentací, webový editor má možnost přepnutí do tmavého režimu. Dále zajišťuje propojení s GitHub nebo GitLab repozitářem.

2.4 Výběr nástroje pro návrh a dokumentaci API

V této sekci navážu na předešlou analýzu nástrojů pro návrh REST API a vyberu zde nejvhodnější nástroj. Nejvíce mě nadchnul Swagger a Apiary, popíši zde porovnání těchto nástrojů.

Oba nástroje jsem vyzkoušel, Swagger jsem zkoušel ve SwaggerHubu, což je webová aplikace, která přináší všechny základní vlastnosti Swaggeru. Oba nástroje umožňují požadovanou dokumentaci a návrh API a k tomu i generování kódu.

Co jsem mi na Swaggeru líbilo a v Apiary chybělo je, že vizualizovaná dokumentace měla označené zabezpečené koncové body, dokumentace se dá exportovat do html, editor obsahuje navigátora, který umožňuje přeskakovat do různých sekcí dokumentu a náhled dokumentace také odkazuje na části dokumentu, takže se k nim jde jednoduše "prokliknout". Zpřehlednění orientace v dokumentu, jež SwaggerHub nabízí, je velice vhodné, protože dokumentace API čítá běžně stovky až tisíce řádků.

Apiary nemá žádný export, existuje ale Apiary CLI, které umožňuje zobrazit dokumentaci v internetovém prohlížeči.

2.4.1 OpenAPI Specifikace vs API Blueprint

Markantní rozdíl mezi nimi je, že Swagger je těžce provázán s OpenAPI Specifikací, kdežto Apiary, přestože podporuje OpenAPI Specifikaci, je založeno na API Blueprintu.

2.4. Výběr nástroje pro návrh a dokumentaci API

Při porovnání repozitářů API Blueprint a OAS můžeme vidět, že OpenAPI Specifikace je více živější, co se týče commitů i řešených problémů, navíc má zhruba dvakrát více hvězdiček a forků.

OAS je založený na YAML (nebo JSON) formátu a API Blueprint má vlastní formát APIB, který má syntaxi podobnou Markdownu a využívá MSON². Rozdíl v těchto formátech je spíše otázka osobního vkusu, jelikož se liší především syntaxí.

2.4.2 Závěr

Oba nástroje jsou srovnatelné, ale Swagger nabízí více možností pracování s dokumentací a hlavně proto jsem se rozhodl ho nominovat jako vítěze tohoto pomyslného souboje.

²<https://github.com/apiaryio/mson>

Návrh

Tabulka 3.1: Pokrytí funkčních požadavků

Funkční požadavek	API endpoint
Přihlášení	GET /login
Odhlášení	nemusí se řešit v API
Smaž model	Delete /model
Vytvoř nový model	Put /model
Editovat práva k modelu	Post /model/{modelId}
Editovat práva k projektu	Post /project/{projectId}
Seskupení	Post /group/{groupId}
Žádosti o práva	Get /rights/model/{modelId}
Zobrazení miniatur po přihlášení	Get /miniatures
Třídění miniatur	nemusí se řešit v API
Zobrazení modelů projektu v určitém čase a počasí	Get /model/{modelId}/time/{time}/weather/{weather}
Ulož všechny změny v projektu	Post /project/{projectId}
Vytvoř kopii modelu	Put /model
Přidání modelu do projektu	Post /project/{projectId}
Vytvoř nový projekt	Put /project
Vytvoř kopii projektu	Put /project
Nahrání modelu z lokálního zařízení	Put /project

Vývojářské prostředí pro OS Linux

4.1 Příprava vývojářského prostředí

4.1.1 Node.js

Listing 4.1: Instalace Node.js a npm

```
sudo apt install nodejs --Yes  
sudo apt install npm --Yes
```

V implementaci je použita verze nodejs 8.10.0 a npm 3.5.2. Vaši verzi si můžete zkontrolovat pomocí následujících příkazů.

Listing 4.2: Zkontrolování verze Node.js a npm

```
nodejs -v  
npm -v
```

4.1.2 PostgreSQL

TBD

4.2 Verzování

Tady popíšu doporučený způsob verzování zdrojových kódů.

Implementace

Testování

Závěr

Seznam použitých zkratk

API Application Programming Interface

I/O Input/Output

npm Node.js package manager

IDE Integrated Development Environment

JSON JavaScript Object Notation

OAS OpenAPI Specification

CI Continuous Integration

Obsah přiloženého CD

	readme.txt	stručný popis obsahu CD
	exe.....	adresář se spustitelnou formou implementace
	src	
	impl.....	zdrojové kódy implementace
	thesis.....	zdrojová forma práce ve formátu L ^A T _E X
	text	text práce
	thesis.pdf.....	text práce ve formátu PDF
	thesis.ps.....	text práce ve formátu PS