FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Generování kalendářových událostí z dat dle Otevřených formálních norem
Student:	Matouš Dlabal
Vedoucí:	RNDr. Jakub Klímek, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem práce je z dat zveřejněných podle Otevřených formálních norem (OFN) [1] generovat kalendářové události, které bude možné jednoduše přidat do běžných kalendářových aplikací. Generování kalendářových událostí z datových sad publikovaných dle OFN, které obsahují časový údaj (například OFN Události), může motivovat poskytovatele dat ke zveřejňování dat právě podle OFN. Navrhněte, implementujte, zdokumentujte a otestujte:

1) Knihovnu pro generování kalendářových událostí ve formátu iCalendar z dat dle specifikace OFN.
2) Webovou službu, která bude zprostředkovávat funkce knihovny.
3) Webovou stránku, která bude nabízet uživatelské rozhraní pro využití funkcí knihovny a webové služby.

–

[1] Otevřené formální normy, Ministerstvo vnitra, https://ofn.gov.cz

---

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 17. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Generování kalendářových událostí z dat dle Otevřených formálních norem

Matouš Dlabal

Katedra softwarového inženýrství
Vedoucí práce: RNDr. Jakub Klímek, Ph.D.

12. května 2021

Poděkování

Chtěl by poděkovat RNDr. Jakubovi Klímkovi, Ph.D., vedoucímu této práce, za cenné rady a připomínky. Zmínku si zaslouží i neobvykle rychlé odpovědi na emaily. Dále děkuji své rodině za podporu během studia a tvorby této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 12. května 2021 . . . . . .

Odkaz na tuto práci

Dlabal, Matouš. *Generování kalendářových událostí z dat dle Otevřených formálních norem*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato práce se zaměřuje na zpracování dat dle otevřených formálních norem (OFN), což jsou standardy pro zveřejňování dat, definované Ministerstvem vnitra. Cílem je vytvořit nástroj umožnující generování kalendářových událostí z takových dat, která jsou zveřejněná podle OFN a navíc obsahují časovou specifikaci. Práce obsahuje analýzu, návrh a dokumentaci pro implementační část – knihovnu, HTTP službu a webovou stránku – a také popis některých formátů a norem. Knihovna a HTTP služba, které je možné dále využít i v rámci jiných nástrojů, umožňují samotné generování událostí. Webová stránka, dostupná na adrese `https://matous-dlabal.gitlab.io/ofn-to-ical-web/`, nabízí formulář pro pohodlnou práci s knihovnou a HTTP službou.

Klíčová slova otevřené formální normy, OFN, iCalendar, kalendářové události, JavaScript, Node.js

Abstract

This thesis focuses on processing data conformant with Formal Open Standards (FOS), which are the standards for data publication, as defined by the Ministry of the Interior. The aim is to create a tool capable of generating calendar events from data, that are both conformant with FOS and contain a time specification. The thesis contains analysis, design and documentation for implementation – library, HTTP service and website – and also a description of several formats and norms. The library and HTTP service, which can be further used by other applications, allow for the generation of events itself. The website, accessible at `https://matous-dlabal.gitlab.io/ofn-to-ical-web/`, provides a form for easy use of the library and HTTP service.

Keywords Formal Open Standards, FOS, iCalendar, calendar events, JavaScript, Node.js

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Úvod

Otevřené formální normy (OFN), vydávané Ministerstvem vnitra, jsou pravidla, která specifikují konkrétní strukturu, kterou je doporučené použít při publikování otevřených dat. Samotný termín je definován zákonem č. 106/1999 Sb., o svobodném přístupu k informacím.

Publikování dat podle jednotných norem přináší řadu výhod. Všechna takto zveřejněná data mají daný formát a strukturu, díky čemuž se s nimi dá jednotně pracovat. S nárůstem nástrojů, které budou data dle OFN zpracovávat, by se mohlo praktické užívání těchto norem rozšířit.

OFN jsou definované vždy pro konkrétní typy dat. Existuje například otevřená formální norma pro *události*, *turistické cíle* nebo *sportoviště*. Tyto normy, mimo jiné, sdílí společnou část (OFN *časová specifikace*), která umožňuje různými způsoby zaznamenat nějaký časový údaj. Pro *události* reprezentuje dobu trvání, pro *turistické cíle* a *sportoviště* jejich otevírací dobu.

Cílem této práce je z dat zveřejněných podle OFN generovat kalendářové události, které bude možné jednoduše přidat do běžných kalendářových aplikací. To je funkcionalita, která zatím není pro otevřené formální normy dostupná.

Konkrétně se práce zaměřuje na ty otevřené formální normy, které obsahují časový údaj v podobě OFN *časová specifikace*. Dílčím cílem je umožnit tuto funkcionalitu použít i jinými nástroji, které s daty dle OFN pracují.

Cílem rešeršní části práce je informovat čtenáře o otevřených formálních normách a seznámit ho s detaily OFN *časová specifikace* a formátu iCalendar. Pro splnění hlavního cíle je stěžejní vytvořit knihovnu, která bude z dat publikovaných dle OFN generovat kalendářové události právě ve formátu iCalendar. Půjde tedy o převod dat odpovídajících OFN *časová specifikace* do formátu, který je podporovaný mnoha kalendářovými aplikacemi, včetně těch od společností Google, Microsoft a Apple [1, 2, 3].

Dále má tato práce za cíl vytvořit HTTP službu, která bude zprostředkovávat funkce knihovny. Umožní tak její využití v rámci programů, které jsou napsány v jiných programovacích jazycích.

Pro jednoduché použití knihovny koncovým uživatelem je dalším dílčím cílem vytvořit webovou stránku, která umožní využití funkcí knihovny (potažmo HTTP služby) pomocí grafického uživatelského rozhraní. Webová stránka se dá navíc použít bez nutnosti instalace či manuálního stahování programu. Jediné, co si uživatel stáhne budou vygenerované kalendářové události.

Pro každou ze tří výše zmíněných praktických částí je nutné provést analýzu, sestavit návrh řešení a výslednou implementaci patřičně zdokumentovat.

V první kapitole jsou definovány používané pojmy a čtenář je blíže seznámen se strukturou otevřené formální normy *časová specifikace* a formátu iCalendar. Následuje zhodnocení existujících řešení (kapitola 2), analýza praktických částí práce (kapitola 3), návrh řešení (kapitola 4), popis odlišností implementace a návrhu (kapitola 5), testování výsledků (kapitola 6), dokumentace (kapitola 7) a závěr.

Kapitola 1

Definice pojmů

V této kapitole jsou vysvětleny termíny otevřená data, otevřená formální norma a kalendářová událost. Bližší pozornost je věnována otevřené formální normě *časová specifikace* a její struktuře. Také je zde blíže popsán formát iCalendar a způsob, jakým se pomocí něj dají reprezentovat kalendářové události.

1.1 Otevřená data

Otevřená data jsou podle § 3 odstavce 11 zákona č. 106/1999 Sb. definovaná, jako *„. . . informace zveřejňované způsobem umožňujícím dálkový přístup v otevřeném a strojově čitelném formátu, jejichž způsob ani účel následného využití není omezen a které jsou evidovány v národním katalogu otevřených dat.“* [4]

1.2 Otevřené formální normy

Otevřená formální norma (OFN) je podle § 3 odst. 9 zákona č. 106/1999 Sb. *„. . . pravidlo, které bylo vydáno písemně a obsahuje specifikace požadavků na zajištění schopnosti různých programových vybavení vzájemně si poskytovat služby a efektivně spolupracovat.“* [4] Jde o doporučení, jakým způsobem zveřejňovat konkrétní datovou sadu [5]. Kromě celých sad jsou definovány také základní datové typy a sdílené specifikace, které jsou součástí více datových sad [5]. Pro tuto práci je stěžejní OFN *časová specifikace*, podrobněji popsaná v kapitole 1.2.1.

Práce se jinak nezaměřuje na žádnou konkrétní datovou sadu. Reprezentativní příklady, obsahující *časovou specifikaci*, jsou OFN *události*, OFN *turistické cíle* a OFN *sportoviště*. *Časová specifikace* u posledních dvou jmenovaných sad informuje o otevírací době [6, 7].

1.2.1 OFN Časová specifikace

Časová specifikace určuje množinu časových úseků, které se dají definovat několika různými způsoby. Pokud je uvedeno více úseků, reprezentuje *časová specifikace* jejich průnik. [8]

Základními prostředky pro určení časového úseku jsou *časový interval* a *časový okamžik*. Okamžik určuje buď jeden konkrétní den, nebo bod v čase (na sekundu přesně). Interval, který začíná a končí časovými okamžiky, může být také určený oběma těmito možnostmi, nelze je však kombinovat. Dalšími vlastnostmi jsou *den v týdnu* a *časové období*. Jejich hodnoty jsou z českého číselníku (seznam přípustných hodnot [9]) pro dny v týdnu a z evropského číselníku pro časové období. *Časové období* může *časovou specifikaci* omezit například na konkrétní měsíc, jaro nebo pracovní dny. [8, 10]

Pro definování událostí, které se s určitou frekvencí opakují existuje v OFN *frekvence*. Jde o evropský číselník, který nabízí denní, čtrnáctidenní, dvouměsíční, čtvrtletní, pětileté a další frekvence. Například časový interval ohraničující rok 2020 znamená v kombinaci se čtrnáctidenní frekvencí opakování každých čtrnáct dní v roce 2020. Pro opakování bez přesně dané frekvence je možné určit počet opakování číslem. Lze tak zaznamenat např. opakování 3× během nějakého intervalu. Naopak pro přesnější určení opakování je možné použít typ *specifická frekvence*. Ta umožňuje určit, přesně které minuty, hodiny, dny, týdny nebo roky se událost opakuje. Například vždy 8 a 20 hodin, 1. den v měsíci. Dny se určují v rámci měsíce. Týdny je možné specifikovat v rámci měsíce, nebo v rámci roku. Rok se váže na desetiletí, nebo století (např. vždy 20. rok ve století). [8, 11]

Dalším nástrojem pro bližší určení *časové specifikace* je *časová doba*, která *„určuje opakující se čas nebo dobu ohraničenou různými údaji o čase. Příkladem může být otevírací doba v určitý den, nebo čas vývozu kontejneru.“* [8]

Poslední číselník použitý v této otevřené formální normě je český číselník pro jinou časovou specifikaci. *Jiná časová specifikace* slouží pro situace, kdy je potřeba omezit platnost za podmínky dobrého/špatného počasí. [8]

Poslední dva parametry – výjimka a časová platnost – jsou typu *časová specifikace*. Výjimka určuje časové úseky, které se mají od hlavní *časové specifikace* odebrat. Časová platnost určuje, kdy je *časová specifikace* platná. [8]

1.3 Kalendářová událost

Kalendářovou událostí se myslí časové období, doplněné o další informace, jako jsou název a popis. Zaznamenává se v rámci kalendáře a může sloužit například k vyznačení veřejné akce, nebo otevírací doby.

Pokud se kalendářová událost zmiňuje, jako výsledek nějakého procesu generování, jsou tím myšlena data, která událost jednoznačně definují. Takovými daty mohou být název, popis, počátek, konec, opakování apod.

1.4 Formát iCalendar

Formát iCalendar slouží k výměně kalendářových informací, mimo jiné i kalendářových událostí, nezávisle na konkrétní aplikaci. Je definován standardem RFC 5545. [12]

Jde o textový formát, který má data rozdělena do řádek, které končí sekvencí znaků `CRLF`. Řádky přesahující délku 75 znaků (včetně dvouznakového zalomení řádky) musí být rozděleny. Hlavní iCalendar objekt začíná řádkem `BEGIN:VCALENDAR` a končí `END:VCALENDAR`. Povinnými parametry jsou identifikátor `PRODID`, verze použité specifikace a alespoň jedna komponenta – v kontextu této práce konkrétně `VEVENT` určující kalendářovou událost. Identifikátor `PRODID` určuje produkt (program), který kalendář vytvořil. [12]

Komponenta kalendářové události je určena parametry uvnitř bloku, který začíná řádkem `BEGIN:VEVENT` a končí `END:VEVENT`. Povinnými parametry jsou identifikátor `UID`, časová značka (`DTSTAMP`) a ve většině případů i počátek události (`DTSTART`). Unikátní identifikátor `UID` globálně určuje kalendářovou událost. Časová značka určuje datum a čas vzniku, případně poslední úpravy, kalendářové události. Parametr `DTSTART` udává počátek události. V případě celodenních událostí jde jen o datum, jinak o datum i čas. Mezi nepovinné, v této práci využívané, parametry patří název události (`SUMMARY`), popis události (`DESCRIPTION`) a konec události (`DTEND`). Hodnota `DTEND` musí být stejného typu (datum / datum a čas), jako hodnota `DTSTART`. Musí však hodnotu `DTSTART` časově následovat. [12]

Formát data je založený na specifikaci ISO 8601. Čtyřmístný rok je bez žádného oddělovače následován číslem měsíce (zarovnaným na dvě číslice) a poté, opět bez oddělovače, číslem dne v měsíci (opět zarovnaným na dvě číslice). Datum 15. března 2021 je tedy zapsáno, jako `20210315`. [12]

Zápis data spolu s časem opět vychází z normy ISO 8601. Na zápisu data se oproti předchozímu odstavci nic nemění. Časová složka je 24hodinová a přidává se za datum, od kterého je oddělena velkým písmenem `T`. Zleva doprava je postupně uvedena hodina, minuta a sekunda dne. Jednotlivé složky jsou zarovnané na dvě číslice a nejsou od sebe nijak odděleny. Čas 14:30 dne 15. března 2021 je zapsán jako `20210315T143000`. [12]

Jelikož na planetě Zemi není ve stejnou chvíli stejný čas, je potřeba možnost přidat k času i informaci o časovém pásmu. Formát iCalendar nepovoluje zápis pomocí posunu oproti UTC (koordinovaný světový čas). Místo toho nabízí tři možnosti zápisu času. [12]

První z nich je zápis pomocí „plovoucího“ místního času. Tento zápis přesně odpovídá zápisu popsanému výše a neobsahuje žádnou dodatečnou informaci o časovém pásmu. Čas `070000` označuje 7 hodin ráno v daném místě, nezávisle na časové zóně. [12]

Druhou možností je zápis pomocí „absolutního“ UTC času. V tomto případě stačí za čas doplnit velké písmeno `Z`. Pokud se nacházíme v časovém

pásmu posunutém oproti UTC o +1 hodinu, čas `070000Z` označuje 8 hodin ráno místního času. [12]

Poslední možností je zápis pomocí místního času a reference časového pásma. Časové pásmo je označené pomocí parametru `TZID`. Tento parametr identifikuje časovou zónu definovanou komponentou `VTIMEZONE`. Pokud máme v objektu iCalendar definované časové pásmo pro střední Evropu (s příslušným identifikátorem), pak čas `TZID=Europe/Prague:070000` značí 7 hodin ráno středoevropského času. [12]

Pro opakující se události se využívá parametr `RRULE`. Jeho hodnota je strukturovaná a obsahuje jednu, nebo více částí pravidla pro opakování. Jednotlivé části se zapisují jako `<název části pravidla>=<hodnota>` a jsou od sebe oddělené znakem středník. Jednotlivé části pravidla se ve výčtu nesmí vyskytnout více než jednou, na jejich pořadí však nezáleží. Povinnou částí je `FREQ`, která určuje frekvenci opakování. Platné hodnoty jsou

- `SECONDLY` (událost se opakuje každou sekundu),
- `MINUTELY` (událost se opakuje každou minutu),
- `HOURLY` (událost se opakuje každou hodinu),
- `DAILY` (událost se opakuje denně),
- `WEEKLY` (událost se opakuje týdně),
- `MONTHLY` (událost se opakuje měsíčně),
- `YEARLY` (událost se opakuje ročně).

Hodnoty části `INTERVAL` nabývají kladných celých čísel a indikují, v jakém intervalu se události opakují. Pro frekvenci `DAILY` značí implicitní hodnota `"1"`, že se událost opakuje každý den. Hodnota `"2"` značí obden a hodnota `"5"` určuje opakování události každých 5 dní. [12]

Formát iCalendar nabízí dvě možnosti, jak určit, do kdy se má událost opakovat. Pokud se neuvede ani jedna z dále zmíněných částí pravidla, opakuje se událost do nekonečna. První částí je `UNTIL`, která udává datum, do kterého (včetně) se má událost opakovat. Druhou je `COUNT`, která udává, kolikrát se má událost opakovat. Specifikovat obě části zároveň není povoleno. Hodnota `UNTIL` musí být stejného typu, jako hodnota parametru `DTSTART` (datum / datum a čas). [12]

Specifikace také umožňuje omezit dny, kdy se má událost odehrávat. Slouží k tomu pravidlo `BYDAY`, kterému se přiřadí čárkou oddělený seznam hodnot reprezentující dny v týdnu (`MO`, `TU`, `WE`, `TH`, `FR`, `SA`, `SU`). V případě měsíční nebo roční frekvence je možné před hodnotu přidat celé číslo n (kladné i záporné), které udává n-tý výskyt daného dne v měsíci. `+1MO` (nebo `1MO`) při měsíčním opakování značí první pondělí v měsíci, `-1MO` pak pondělí poslední. Pro roční

frekvenci má hodnota jiný význam. Pokud je nastaveno pravidlo `BYMONTH`, určuje číslo posun v rámci měsíce. Pokud není nastaveno pravidlo `BYMONTH`, určuje posun v rámci roku. Číslo se u pravidla `BYDAY` nesmí nastavit, pokud je při roční frekvenci nastaveno pravidlo `BYWEEKNO`. [12]

Pravidlu `BYMONTH`, které určuje měsíce, se také přiřazuje čárkou oddělený seznam. Hodnoty jsou čísla v rozmezí 1 až 12.

Pravidlo `BYMONTHDAY` se opět nastavuje seznamem. Hodnoty jsou od ± 1 do ± 31. Například -10 značí desátý den od konce měsíce. Pravidlo nesmí být nastaveno zároveň s týdenní frekvencí. [12]

Pravidlo `BYWEEKNO` obsahuje seznam hodnot, které udávají týdny v roce (± 1 až ± 53). Týden je sedm po sobě jdoucích dnů. První den v týdnu se nastavuje pravidlem `WKST` (implicitně pondělí). První týden v roce je takový, který jako první obsahuje alespoň čtyři dny daného roku. Pravidlo nesmí být použito, pokud je frekvence jiná, než roční (`YEARLY`) [12].

Pro vymezení času, kdy se událost opakuje, jsou určená pravidla `BYSECOND`, `BYMINUTE` a `BYHOUR`. Všechny tři přebírají čárkou oddělený seznam hodnot – čísel. Platné sekundy jsou v rozmezí 0–60, minuty 0–59 a hodiny 0–23. Tyto pravidla nesmí být specifikována, pokud je parametr `DTSTART` pouze datum bez časové složky. [12]

Kapitola 2

Existující řešení

Tato kapitola analyzuje existující řešení problému generování kalendářových událostí z dat dle OFN.

Použitím vyhledávače Google a klíčových slov OFN, otevřené formální normy a otevřená formální norma je možné najít webovou stránku `https://data.gov.cz`, která je spravovaná Ministerstvem vnitra a obsahuje informace o OFN i jejich konkrétní popis, oficiální GitHub účet *Otevřená data ČR @ MVČR* a bakalářskou práci *Aplikace pro podporu otevírání dat o dřevinách pomocí otevřené formální normy*. Bakalářská práce se zaměřuje na návrh nové OFN a tvorbu webové aplikace pro zobrazení dat na mapě [13].

GitHub účet pro OFN obsahuje dva relevantní repozitáře – *zobrazeni-dat-nkod-dle-ofn* a *app-ofn-plakaty*. Interní vyhledávače repozitářů GitHub a GitLab objevili za použití výše zmíněných klíčových slov (rozšířených o obměnou otevrene-formalni-normy) jediný další související projekt – *otevrene-formalni-normy-dts*. Ten se však soustředí jen na OFN *rozhraní katalogů otevřených dat*, která neobsahuje OFN *časová specifikace* a není tedy pro tuto práci relevantní [14].

Projekt *app-ofn-plakaty* se soustředí na generování plakátů z dat dle OFN *turistické cíle* a *sportoviště*, ale neřeší jejich otevírací dobu, na kterou se zaměřuje tato práce [15].

Projekt *zobrazeni-dat-nkod-dle-ofn* obsahuje, mimo jiné, zárodek zpracování dat dle OFN *časová speficikace*. Jde ale o převod na text, ne na kalendářová data. [16]

Rozšíření výše zmíněných klíčových slov o iCalendar, iCal a kalendář, nevedlo k žádným dalším výsledkům, které by souvisely s otevřenými formálními normami.

Tento průzkum potvrzuje, že se tato práce soustředí na funkcionalitu, která zatím není pro OFN dostupná.

Kapitola 3

Analýza

V této kapitole je popsána analýza požadavků, která vznikala především konzultacemi s vedoucím práce. Kapitola je rozdělena do čtyř podkapitol. Každá z nich se zaměřuje na jeden samostatný celek a popisuje funkční požadavky, případy užití a jejich scénáře.

3.1 Knihovna

Úkolem knihovny je samotné generování kalendářových událostí. Díky tomu tvoří základ pro všechny následující části práce, které ji – ať už přímo, nebo nepřímo – využívají.

3.1.1 Požadavky

Následující požadavky ohraničují rozsah funkcionality knihovny.

3.1.1.1 Funkční požadavky

R1.01 Knihovna musí umět zpracovat data odpovídající specifikaci OFN, která obsahují *časovou specifikaci* [8].

R1.02 Knihovna nemusí zohledňovat více než jednu výjimku OFN *časová specifikace*.

R1.03 Knihovna musí umět zpracovat data ve formátu JSON-LD 1.1 [17].

R1.04 Knihovna musí umět ze zadaných dat generovat kalendářové události.

R1.05 Knihovna generuje název kalendářové události z názvu hlavního – hierarchicky nejvýše postaveného – objektu. Každý objekt tento atribut dědí z OFN *věc* [18].

R1.06 Knihovna generuje popis kalendářové události z popisu hlavního – hierarchicky nejvýše postaveného – objektu a z popisu *časové specifikace*. Oba objekty tyto atributy dědí z OFN *věc*.

R1.07 Knihovna generuje časovou složku kalendářové události z dat OFN *časová specifikace*.

R1.08 Knihovna musí umět kalendářovou událost uložit ve formátu iCalendar [12].

3.1.1.2 Nefunkční požadavky

R1.09 Funkčnost knihovny bude demonstrovaná formou HTTP služby. Knihovnu bude využívat server poskytující službu.

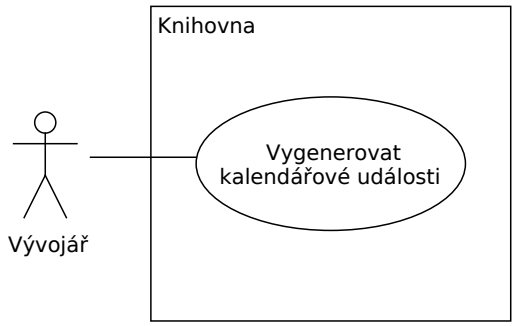
R1.10 Funkčnost knihovny bude demonstrovaná formou webové stránky. Knihovna se bude využívat lokálně – v prohlížeči uživatele.

R1.11 Ke knihovně bude dodána dokumentace.

R1.12 Kód knihovny bude verzovaný systémem git a dostupný na git repozitáři.

3.1.2 Případy užití

Diagram na obrázku 3.1 zobrazuje případy užití knihovny. Jediný aktér diagramu je vývojář, uživatel knihovny, respektive program, který knihovnu využívá.



Obrázek 3.1: Use case diagram – knihovna

3.1.2.1 Vygenerovat kalendářové události

Knihovna ze zadaných dat vygeneruje jednu, nebo více kalendářových událostí ve formátu iCalendar.

```
{
    "@context": "https://ofn.gov.cz/události/2020-07-01/kontexty/událost.jsonld",
    "typ": "Událost",
    "iri": "https://data.brno.cz/zdroj/události/2019/vinobraní",
    "název": {
        "cs": "Brněnské vinobraní",
        "en": "Brno Wine Festival"
    },
    "popis": {
        "cs": "Město Brno Vás zve na již 21. vinobraní.",
        "en": "City of Brno invites you to the 21th Wine Festival."
    },
    "doba_trvání": [{
        "typ": "Časová specifikace",
        "časový_interval": {
            "typ": "Časový interval",
            "začátek": {
                "typ": "Časový okamžik",
                "datum_a_čas": "2019-01-01T09:30:00+02:00"
            },
            "konec": {
                "typ": "Časový okamžik",
                "datum_a_čas": "2019-01-07T18:00:00+02:00"
            }
        }
    }, {
        "typ": "Časová specifikace",
        "časový_interval": {
            "typ": "Časový interval",
            "začátek": {
                "typ": "Časový okamžik",
                "datum_a_čas": "2019-01-14T09:30:00+02:00"
            },
            "konec": {
                "typ": "Časový okamžik",
                "datum_a_čas": "2019-01-20T18:00:00+02:00"
            }
        }
    }]
}
```

Toto jsou redukovaná data ve formátu JSON-LD obsahující pro knihovnu potřebné informace. Plné znění ukázkové události je součástí specifikace OFN *události* [19].

Výpis kódu 3.1: Redukovaný příklad dat dle OFN *události*

Základní scénář

1. Vývojář předá knihovně následující vstupní argumenty:

    i. Řetězec obsahující korektní vstupní data ve formátu JSON-LD. (např.: událost 3.1)

ii. Řetězec obsahující dvoupísmenný kód jazyka dle normy ISO 639-1 [20]. Ten určí jazyk, který se má při generování kalendářové události využít. (např.: kód `cs` pro češtinu, nebo `en` pro angličtinu)

iii. Jakým způsobem se zachovat v případě, budou-li vstupní data tvořena seznamem objektů OFN.

a) Vytvořit jeden kalendář ve formátu iCalendar.

b) Vytvořit pro každý prvek seznamu samostatný kalendář.

2. Knihovna ze zadaných dat vygeneruje kalendářové události.

3. Knihovna vrátí kolekci kalendářů ve formátu iCalendar. Jejich počet závisí na vstupu a na volbě argumentu iii.

Alternativní scénář – neplatný vstup

1. Vývojář předá knihovně argumenty s neplatnými hodnotami.

2. Knihovna ohlásí neplatnost zadaného vstupu.

3.1.3 Převod dat dle OFN na kalendářové události

Podle požadavku R1.05 se bude název události brát z hodnoty klíče název. Proto bude tato hodnota vyžadována. Popis události bude odvozován z hodnoty klíče popis (R1.06). Na rozdíl od názvu nebude popis vyžadován. Kromě názvu bude povinný i klíč typ, který určuje typ OFN a bude proto sloužit pro hledání *časové specifikace* v zadaných datech.

3.1.3.1 Časová pásma

Veškeré časy budou do formátu iCalendar zadávány pomocí místního času, tedy bez žádné informace o časovém pásmu. Přestože hodnoty data a času OFN mají informaci o posunu oproti UTC, není z nich možné zjistit, jestli se má použít normální, nebo letní čas a kdy (pokud vůbec) dochází k jeho změně. Navíc *časová doba* informaci o posunu vůbec nemusí mít.

3.1.3.2 Převod OFN časová specifikace do formátu iCalendar

Zde jsou popsány podrobnosti převodu OFN *časová specifikace* do formátu iCalendar. Jelikož *časová specifikace* popisuje časové údaje jinými prostředky, než formát iCalendar, nebude převod pouhým „překladem“ do jiného formátu.

Časová specifikace se dá definovat pomocí typů *časový okamžik*, *časový interval*, *den v týdnu*, *časové období*, *frekvence*, *specifická frekvence*, *časová doba*, *jiná časová specifikace* a také pomocí číselného parametru *počet opakování* nebo parametrů *výjimka* a *časová platnost* (oba typu *časová specifikace*) [8]. Mezi nejzákladnější způsoby určení *časové specifikace* patří použití

typů *časový interval* a *časový okamžik*. Tyto typy mají ve formátu iCalendar jasný ekvivalent – `DTSTART` a `DTEND`. V případě, že je nutné použít některé z pravidel pro opakování, je parametr `DTEND` nahrazen `UNTIL`, částí pravidla pro opakování. Například *časový interval* 3.2 se transformuje na kalendářovou událost 3.3. Formát umožňuje uvést jen atribut `DTSTART` a definovat tak jednostranný interval, který nemá určený konec. Naopak to však možné není. Proto je potřeba zavést hodnotu `DTSTART` pro interval bez začátku – počátek 20. století (`1900-01-01`).

```
{
    "typ": "Časový interval",
    "začátek": {
        "typ": "Časový okamžik",
        "datum": "2021-01-01"
    },
    "konec": {
        "typ": "Časový okamžik",
        "datum": "2021-01-15"
    }
}
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.2: Příklad dat typu OFN *časový interval*

```
BEGIN:VEVENT
DTSTART:20210101
DTEND:20210115
END:VEVENT
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.3: Příklad intervalu ve formátu iCalendar

Časový okamžik se bere, jako interval s identickým začátkem a koncem. Pokud je v časové specifikaci určen interval i okamžik, knihovna provede jejich průnik. Pokud je průnik prázdný, vyhodí výjimku. V případě, kdy je okamžik definován jako datum a čas, se další části časové specifikace ignorují.

Převést *den v týdnu* je relativně jednoduché. Součástí pravidel pro opakování událostí formátu iCalendar je i část `BYDAY`, určující právě dny v týdnu [12]. Nastavením denního opakovaní a výběrem dnů se dosáhne žádaného omezení události. *Časová specifikace* 3.4 se tak převede na kalendářovou událost 3.5.

Časové období má hodnoty dvou typů – hodnoty vymezující, kdy se má událost konat, a hodnoty určující délku konání. Je možné zvolit dny v týdnu, pracovní dny, měsíce a roční období, ve kterých se událost koná. Obdobně, jako pro dny, existuje ve formátu iCalendar část pravidla pro opakování, která vymezuje měsíce (`BYMONTH`) [12]. Za pracovní dny jsou zpravidla považovány dny v rozmezí Po–Pá [10]. Jelikož jsou pracovní dny navíc určené například státními svátky, bude o tom knihovna přidávat poznámku do popisu události.

```
{
    "typ": "Časová specifikace",
    "den_v_týdnu": [
        "https://data.mvcr.gov.cz/zdroj/číselníky/dny-v-týdnu/položky/pondělí",
        "https://data.mvcr.gov.cz/zdroj/číselníky/dny-v-týdnu/položky/středa",
        "https://data.mvcr.gov.cz/zdroj/číselníky/dny-v-týdnu/položky/pátek"
    ]
}
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.4: Příklad OFN *časová specifikace* s určenými dny v týdnu

```
BEGIN:VEVENT
DTSTART:19000101
RRULE:FREQ=DAILY;BYDAY=MO,WE,FR
END:VEVENT
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.5: Příklad omezení dnů ve formátu iCalendar

Teoreticky mohou být zároveň s pracovními dny zvoleny i další dny týdne. V takovém případě se budou buď krýt a výsledná kalendářová událost se tím nezmění, nebo půjde o víkend a generování skončí neúspěchem. Roční období bude pro jednoduchost chápáno jako meteorologické roční období severní hemisféry. Každé roční období tak odpovídá právě třem měsícům – jaro začíná počátkem března, ne jarní rovnodenností [21].

Délka trvání události neurčuje přesně, kdy se má událost konat. Pokud je jinak událost určená na rozmezí týdnů, či měsíců, není možné s informací, že bude trvat například týden nebo den, provést nic lepšího, než ji přidat do popisku události. Není totiž jasné, přesně který týden/den se bude konat. Na ukázkové události 3.6 je vidět kombinace více hodnot, které se převedou na událost 3.7. Datum počátku události je posunuté, protože formát iCalendar nedefinuje chování, kdy hodnota `DTSTART` neodpovídá pravidlům parametru `RRULE` (20. května 2021 není ani sobota, ani neděle) [12]. Pokud by bylo zvoleno více hodnot určujících délku trvání události (např. rok, týden, den), použije se jen ta nejkratší.

Pro vlastnost *frekvence* má formát iCalendar dvě části pravidla pro opakování (`FREQ` a `INTERVAL`). Jelikož se ale frekvencí v *časové specifikaci* neudává přesný časový údaj, není její převod do formátu iCalendar tak přímočarý. Už interpretace zdánlivě prosté frekvence „jednou za dva měsíce“ naráží na několik nejasností. Jak dlouho událost trvá? Kdy v měsíci se událost odehrává? Který z měsíců se koná a který je vynechaný? I pokud je zvolen nějaký interval, nemusí jeho počátek nutně znamenat první uskutečnění události. Jsou v zásadě dvě možnosti, jak řešit první dva problémy. V obou případech by byla frekvence zaznamenána i do popisu události.

```
{
    "typ": "Časová specifikace",
    "časový_interval": {
        "typ": "Časový interval",
        "začátek": {
            "typ": "Časový okamžik",
            "datum": "2021-04-20"
        },
        "konec": {
            "typ": "Časový okamžik",
            "datum": "2021-12-31"
        }
    },
    "časové_období": [
        "http://publications.europa.eu/resource/authority/timeperiod/SAT",
        "http://publications.europa.eu/resource/authority/timeperiod/SUN",
        "http://publications.europa.eu/resource/authority/timeperiod/FEB",
        "http://publications.europa.eu/resource/authority/timeperiod/SPRING",
        "http://publications.europa.eu/resource/authority/timeperiod/HOUR"
    ]
}
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.6: Příklad OFN *časová specifikace* s časovým obdobím

```
BEGIN:VEVENT
DTSTART:20210424
RRULE:FREQ=DAILY;BYMONTH=2,3,4,5;BYDAY=SA,SU;UNTIL=20211231
DESCRIPTION:Událost trvá hodinu.
END:VEVENT
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.7: Příklad omezení dnů a měsíců ve formátu iCalendar

Událost může vyplnit celý týden/měsíc/rok, což by u frekvence například jednou za dva měsíce fungovalo i vizuálně (ve chvíli, kdy by byl kalendář zobrazován na úrovni měsíců). Bohužel v případě většiny ostatních frekvencí (např. měsíční, týdenní nebo dvakrát/třikrát za měsíc) by to znamenalo, že by se událost konala vlastně nepřetržitě.

Druhou možností je, že se událost může konat jen jeden den, například na začátku týdne, na začátku měsíce apod. V takovém případě by byla v kalendáři frekvence vidět, jen by událost nezobrazovala přesný den/dny, kdy se má odehrát, ani jak dlouho trvá. Tato varianta má problém s nepřesnými frekvencemi. Například dvakrát za měsíc nemusí znamenat rovnoměrné rozdělení po dvou týdnech, ale i dvakrát v jednom týdnu. U frekvencí, které pracují s delším intervalem navíc hrozí, že se jednodenní událost ztratí.

Obě varianty mají své problémy, ale pro většinu frekvencí z číselníku se hodí spíše první varianta. Pro frekvence s rozsahem rok a více je omezení

na jeden den příliš hrubé. Většina ostatních je typu x-krát za nějaké období, u kterých není jasné rozprostření. Denní frekvence by v obou případech splývala a u hodinové frekvence je zbytečné souvislou událost tříštit. Tím zbývá měsíční a týdenní frekvence. U těch jediných by mohla dávat smysl druhá varianta. Jelikož by stejně nešlo o přesnou reprezentaci, budou se i tyto dvě frekvence převádět pomocí první varianty. Při převodu *frekvence* se tedy jen přidá poznámka o frekvenci do popisu události.

Číselně určený *počet opakování* není možné převést jinak, než poznámkou v popisu kalendářové události.

Jak již název napovídá, *specifická frekvence* je konkrétnější, než výše popsaná *frekvence*. Pro výběr konkrétní minuty a hodiny, kdy se má událost opakovat, se dá použít `BYMINUTE` a `BYHOUR` (části parametru `RRULE`). Pro den v měsíci existuje pravidlo `BYMONTHDAY`. Pomocí pravidla `BYWEEKNO` se dá specifikovat týden v roce. Týden v měsíci bohužel ve formátu iCalendar žádnou obdobu nemá. Rok v desetiletí, respektive století, se dá specifikovat roční frekvencí a intervalem opakování 10, respektive 100. Protože jde takto nastavit jen jednu hodnotu, je v případě více hodnot použita poznámka v popisu události. Pokud je specifikovaný rok v desetiletí i rok ve století, nastaví se opakování pro desetiletí a století se zaznamená do poznámky. Konkrétní rok se převezme z hodnoty `DTSTART`. Proto bude hodnota posunuta tak, aby rok odpovídal *specifické frekvenci*. Může také dojít k tomu, že bude zadán *časový interval*, který daný rok neobsahuje, a kalendářovou událost nebude možné vygenerovat. Aby se událost s roční frekvencí neopakovala jen jeden den (určeným parametrem `DTSTART`), je potřeba mít nastavené některé z pravidel `BY-`. Pokud takové pravidlo chybí, nastaví se pro frekvenci `YEARLY` všechny dny týdne (`BYDAY`). *Specifická frekvence* bude tedy převedena částečně na různé části pravidla pro opakování (`RRULE`), částečně jen na poznámky v popisu kalendářové události. Transformaci příkladů 3.8, 3.10 a 3.12 zobrazují postupně události 3.9, 3.11 a 3.13

Časová doba je údaj, který se dá do formátu iCalendar převést relativně snadno. Stačí nastavit opakování s denní frekvencí a použít `DTSTART` a `DTEND` se stejným datem a časové složky nastavit podle *časové doby*. Parametr `UNTIL` určuje případné datum konce intervalu a časová složka bude stejná, jako u `DTEND`. Přestože *časová specifikace* umožňuje specifikovat několik *časových dob*, takto je možné do kalendářové události zaznamenat jen jednu. Knihovna bude tedy vždy zpracovávat jen první ze seznamu.

Další omezení je, že všechny dny, kdy se událost odehrává, mají nastavené stejné časové rozmezí. Proto je situace, kdy je nastaven interval s krajními hodnotami času jinými, než je *časová doba*, problematická. Pokud je celá *časová doba* uvnitř prvního/posledního dne intervalu, stačí omezit interval. Pokud je ale začátek/konec intervalu uvnitř *časové doby*, není možné tuto jednodenní výjimku do formátu iCalendar zanést. Jelikož je kombinace *časového intervalu* a *časové doby* myšlená spíše pro déle trvající intervaly, drtivá většina dnů bude mít rozsah času správný a nepůjde tedy o příliš velký problém. Jelikož se při

```
{
    "specifická_frekvence": [
        {
            "typ": "Specifická frekvence",
            "rok_v_desetiletí": 3
        }, {
            "typ": "Specifická frekvence",
            "den_v_měsíci": 1
        }, {
            "typ": "Specifická frekvence",
            "den_v_měsíci": 2
        }, {
            "typ": "Specifická frekvence",
            "den_v_měsíci": 3
        }, {
            "typ": "Specifická frekvence",
            "den_v_měsíci": 4
        }, {
            "typ": "Specifická frekvence",
            "den_v_měsíci": 5
        }
    ]
}
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.8: Příklad dat obsahujících OFN *specifická frekvence*

```
BEGIN:VEVENT
DTSTART=19030101
RRULE:FREQ=YEARLY;INTERVAL=10;BYMONTHDAY=1,2,3,4,5
END:VEVENT
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.9: Příklad specifické frekvence ve formátu iCalendar

převodu v případě intervalů trvajících méně než den může zcela změnit časové rozpětí (a nepůjde o průnik), je v takových situacích lepší nastavit rozmezí času přímo *časovým intervalem* a *časovou dobu* vůbec nepoužívat. Pokud interval začíná až po konci *časové doby*, respektive končí před jejím začátkem, bude při převodu do formátu iCalendar den začátku/konce posunut.

Ukázka 3.14 se převede na kalendářovou událost 3.15. Začátek události je posunutý o den dopředu, protože interval začíná až po 09:15:00. Konec (`UNTIL`) se bohužel oproti původnímu intervalu prodloužil. Nejedná se o chybu, důvod je popsán v předchozím odstavci.

Jiná časová specifikace udává podmínku na počasí. Taková informace se dá zaznamenat jedině v podobě textové poznámky v popisu kalendářové události.

```
{
    "specifická_frekvence": [
        {
            "typ": "Specifická frekvence",
            "hodina": 8
        }, {
            "typ": "Specifická frekvence",
            "hodina": 9
        }, {
            "typ": "Specifická frekvence",
            "hodina": 10
        }, {
            "typ": "Specifická frekvence",
            "týden_v_roce": 1
        }, {
            "typ": "Specifická frekvence",
            "týden_v_roce": 2
        }
    ]
}
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.10: Příklad dat obsahujících OFN *specifická frekvence*

```
BEGIN:VEVENT
DTSTART:19000101T000000
RRULE:FREQ=YEARLY;BYHOUR=8,9,10;BYWEEKNO=1,2
END:VEVENT
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.11: Příklad specifické frekvence ve formátu iCalendar

```
{
    "specifická_frekvence": [
        {
            "typ": "Specifická frekvence",
            "rok_ve_století": 42
        }, {
            "typ": "Specifická frekvence",
            "rok_ve_století": 84
        }
    ]
}
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.12: Příklad dat obsahujících OFN *specifická frekvence*

```
BEGIN:VEVENT
DTSTART=19000101
DESCRIPTION:Událost se opakuje každý 42., 84. rok století.
END:VEVENT
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.13: Příklad specifické frekvence ve formátu iCalendar

```
{
        "typ": "Časová specifikace",
        "časový_interval": {
                "typ": "Časový interval",
                "začátek": {
                        "typ": "Časový okamžik",
                        "datum_a_čas": "2021-01-01T09:15:01+01:00"
                },
                "konec": {
                        "typ": "Časový okamžik",
                        "datum_a_čas": "2021-01-31T08:30:00+01:00"
                }
        },
        "časová_doba": [{
                "typ": "Časová doba",
                "od": "08:00:00",
                "do": "09:15:00"
        }]
}
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.14: Příklad dat s časovou dobou

```
BEGIN:VEVENT
DTSTART:20210102T800000
DTEND:20210102T091500
RRULE:FREQ=DAILY;UNTIL=20210131T091500
END:VEVENT
```

Tento útržek nemusí obsahovat některé povinné atributy.

Výpis kódu 3.15: Příklad intervalu a časové doby ve formátu iCalendar

3.2 HTTP služba

HTTP služba zprostředkovává funkce knihovny.

3.2.1 Požadavky

Následující požadavky ohraničují rozsah funkcionality HTTP služby.

3.2.1.1 Funkční požadavky

R2.01 HTTP služba umožní přijmout data ve formátu JSON-LD. (Z nich se budou generovat kalendářové události.)

R2.02 HTTP služba umožní generovat kalendářové události z dat, která umí zpracovat knihovna.

R2.03 HTTP služba bude posílat vygenerované kalendářové události ve formátu iCalendar.

R2.04 HTTP služba musí pomocí HTTP stavového kódu indikovat, že byla zaslána neplatná data.

3.2.1.2 Nefunkční požadavky

R2.05 Funkčnost HTTP služby bude demonstrovaná pomocí webové stránky.

R2.06 K HTTP službě bude dodána dokumentace.

R2.07 Kód bude verzovaný systémem git a dostupný na git repozitáři.

3.2.2 Případy užití

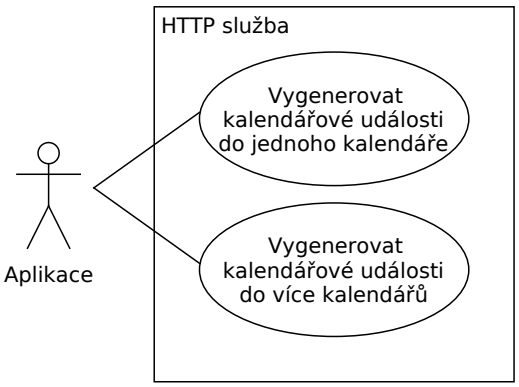
Diagram na obrázku 3.2 zobrazuje případy užití služby. Aktérem je zde aplikace, která HTTP službu využívá.

3.2.2.1 Vygenerovat kalendářové události do jednoho kalendáře

Aplikace zašle vstupní data ve formátu JSON-LD. Server odpoví jedním kalendářem ve formátu iCalendar, který obsahuje vygenerované kalendářové události.

Základní scénář

1. Aplikace pošle HTTP požadavek obsahující:
   i. Korektní data ve formátu JSON-LD v těle požadavku.
   ii. Parametr `Accept-Language` s dvoupísmenným jazykovým kódem v hlavičce.



Obrázek 3.2: Use case diagram – HTTP služba

2. HTTP služba z dat vygeneruje kalendářové události a pošle je v odpovědi jako součást jednoho kalendáře formátu iCalendar.

Alternativní scénář – neplatný vstup

1. Aplikace pošle HTTP požadavek s neplatnými daty.

2. HTTP služba pošle odpověď s HTTP stavovým kódem indikujícím, že byla poslána neplatná data.

3.2.2.2 Vygenerovat kalendářové události do více kalendářů

Aplikace zašle vstupní data ve formátu JSON-LD. Server odpoví seznamem kalendářů ve formátu iCalendar, které obsahují vygenerované kalendářové události.

Základní scénář

1. Aplikace pošle HTTP požadavek obsahující:

   i. Korektní data ve formátu JSON-LD v těle požadavku.

   ii. Parametr `Accept-Language` s dvoupísmenným jazykovým kódem v hlavičce.

2. HTTP služba z dat vygeneruje kalendářové události a pošle je v odpovědi rozdělené do jednoho, nebo více kalendářů formátu iCalendar.

Alternativní scénář – neplatný vstup

1. Aplikace pošle HTTP požadavek s neplatnými daty.
2. HTTP služba pošle odpověď s HTTP stavovým kódem indikujícím, že byla poslána neplatná data.

3.3 Webová stránka – služba

Webová stránka, která využívá HTTP službu. Prezentuje tak její funkčnost a možnost jejího využití.

3.3.1 Požadavky

Následující požadavky ohraničují rozsah funkcionality webové stránky.

3.3.1.1 Funkční požadavky

R3.01 Webová stránka umožní nahrát data z lokálního souboru.

R3.02 Webová stránka umožní nahrát data z internetového zdroje.

R3.03 Webová stránka musí umět zpracovat data, která umí zpracovat HTTP služba.

R3.04 Webová stránka musí upozornit uživatele, že nahraná data nejsou platná.

R3.05 Webová stránka umožní ze zadaných dat vygenerovat a stáhnout kalendářové události ve formátu iCalendar.

3.3.2 Případy užití

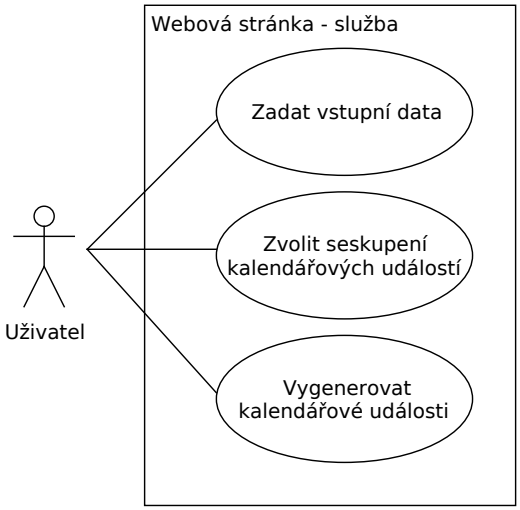
Aktérem je zde uživatel, který používá rozhraní nabízené webovou stránkou. HTTP službu tak používá pouze nepřímo. Případy užití znázorňuje diagram 3.3.

3.3.2.1 Zadat vstupní data

Uživatel zadá vstupní data ve formátu JSON-LD.

Základní scénář – data z lokálního souboru

1. Uživatel klikne na tlačítko nahrát data ze souboru.
2. Uživatel vybere soubor ve formátu JSON-LD.
3. Uživatel potvrdí výběr souboru.
4. Webová stránka načte data ze souboru.



Obrázek 3.3: Use case diagram – webová stránka – služba

Alternativní scénář – data z internetu

1. Uživatel zadá URL adresu dat, která jsou ve formátu JSON-LD.
2. Uživatel klikne na tlačítko pro nahrání data z URL.
3. Webová stránka načte data z internetového zdroje.

Alternativní scénář – data z internetu – neplatná data

1. Uživatel zadá neplatnou adresu URL.
2. Uživatel klikne na tlačítko pro nahrání data z URL.
3. Webová stránka zobrazí upozornění, že data nejdou nahrát.

3.3.2.2 Zvolit seskupení kalendářových událostí

Uživatel zvolí, jestli se má ze seznamu objektů OFN generovat jeden, nebo více kalendářů.

Základní scénář

1. Uživatel nechá zaškrtávací pole (check box) pro seskupení nezaškrtnuté.
2. Webová stránka při následujících generováních události do jednoho kalendáře neseskupí.

Alternativní scénář – seskupit události

1. Uživatel zaškrtávací pole pro seskupení zaškrtne.
2. Webová stránka při následujících generováních seskupí události do jednoho kalendáře.

3.3.2.3 Vygenerovat kalendářové události

Webová stránka pomocí HTTP služby vygeneruje kalendářové události a uloží je ve formátu iCalendar.

Základní scénář

1. Uživatel klikne na tlačítko vygenerovat kalendářové události.
2. Webová stránka pomocí HTTP služby vygeneruje kalendářové události a zahájí jejich ukládání.
3. Uživatel vybere, kam chce soubor uložit. (Tento krok závisí na nastavení prohlížeče.)
4. Uloží se buď jeden soubor formátu iCalendar, nebo archiv více takových souborů.

3.4 Webová stránka – knihovna

Webová stránka, která využívá přímo knihovnu. Prezentuje tak její funkčnost a možnost jejího využití.

3.4.1 Požadavky

Následující požadavky ohraničují rozsah funkcionality webové stránky.

3.4.1.1 Funkční požadavky

R4.01 Webová stránka umožní nahrát data z lokálního souboru.

R4.02 Webová stránka umožní nahrát data z internetového zdroje.

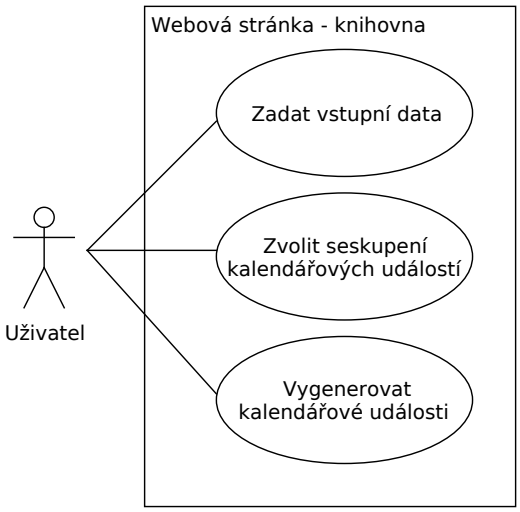
R4.03 Webová stránka musí umět zpracovat taková data, která umí zpracovat knihovna.

R4.04 Webová stránka musí upozornit uživatele, že nahraná data nejsou platná.

R4.05 Webová stránka umožní ze zadaných dat vygenerovat a stáhnout kalendářové události ve formátu iCalendar.

3.4.2 Případy užití

Aktérem je zde uživatel, který používá rozhraní nabízené webovou stránkou. Knihovnu tak používá pouze nepřímo. Případy užití znázorňuje diagram 3.4.



Obrázek 3.4: Use case diagram – webová stránka – knihovna

3.4.2.1 Zadat vstupní data

Uživatel zadá vstupní data ve formátu JSON-LD.

Základní scénář – data z lokálního souboru

1. Uživatel klikne na tlačítko nahrát data ze souboru.
2. Uživatel vybere soubor ve formátu JSON-LD.
3. Uživatel potvrdí výběr souboru.
4. Webová stránka načte data ze souboru.

Alternativní scénář – data z internetu

1. Uživatel zadá URL adresu dat, která jsou ve formátu JSON-LD.
2. Uživatel klikne na tlačítko pro nahrání data z URL.
3. Webová stránka načte data z internetového zdroje.

Alternativní scénář – data z internetu – neplatná data

1. Uživatel zadá neplatnou adresu URL.
2. Uživatel klikne na tlačítko pro nahrání data z URL.
3. Webová stránka zobrazí upozornění, že data nejdou nahrát.

3.4.2.2 Zvolit seskupení kalendářových událostí

Uživatel zvolí, jestli se má ze seznamu objektů OFN generovat jeden, nebo více kalendářů.

Základní scénář

1. Uživatel nechá zaškrtávací pole pro seskupení nezaškrtnuté.
2. Webová stránka při následujících generováních události do jednoho kalendáře neseskupí.

Alternativní scénář – seskupit události

1. Uživatel zaškrtávací pole pro seskupení zaškrtne.
2. Webová stránka při následujících generováních seskupí události do jednoho kalendáře.

3.4.2.3 Vygenerovat kalendářové události

Webová stránka pomocí knihovny vygeneruje kalendářové události a uloží je ve formátu iCalendar.

Základní scénář

1. Uživatel klikne na tlačítko vygenerovat kalendářové události.
2. Webová stránka pomocí knihovny vygeneruje kalendářové události a zahájí jejich ukládání.
3. Uživatel vybere, kam chce soubor uložit. (Tento krok závisí na nastavení prohlížeče.)
4. Uloží se buď jeden soubor formátu iCalendar, nebo archiv více takových souborů.

Kapitola 4

Návrh

Tato kapitola popisuje návrh jednotlivých částí práce. Je rozdělena do tří podkapitol, které odpovídají jednotlivým částem z analýzy (kapitola 3). Přestože má webová stránka analýzu z důvodu přehlednosti rozdělenou do dvou částí, reálně půjde o jeden web a návrh tedy rozdvojen není.

4.1 Knihovna

Knihovna přijímá data dle specifikace OFN a vytváří kalendářové události ve formátu iCalendar.

4.1.1 Technologie

Následuje popis technologií, které knihovna využívá.

4.1.1.1 JavaScript

JavaScript je interpretovaný, objektivně orientovaný jazyk určený (mimo jiné) pro webové stránky [22]. Použití JavaScriptu umožní splnit požadavek R1.10 (viz kapitola 3.1.1) a s využitím Node.js (více v kapitole 4.2.1.1) i požadavek R1.09.

4.1.1.2 Knihovna ical.js

Knihovna ical.js je JavaScriptový parser pro data ve formátu iCalendar [23]. Krom zpracování dat tohoto formátu umožňuje kalendáře i vytvářet [23]. Knihovna je licencovaná pod *Mozilla Public License Version 2.0* [23]. Použití této knihovna umožní korektně splnit požadavek R1.08 (viz kapitola 3.1.1).

4.1.2 Rozhraní

Knihovna má za úkol ze vstupních dat generovat kalendářové události. Jelikož analýza (viz kapitola 3.1) určila jen jeden případ užití, bude knihovna nabízet následující velmi jednoduché rozhraní:

```
OFNCAL.generate(data          :String,
                langCode      :String,
                asOneCalendar :Boolean) -> Array<String>
```

4.1.2.1 Parametr `data`

Pomocí parametru `data` se knihovně předají data ve formátu JSON-LD (požadavek R1.03 – viz kapitola 3.1.1). Pokud data nejsou ve formátu JSON-LD, knihovna (přesněji řečeno metoda `JSON#parse`, která má na starost převod textových data na objekt) vyhodí výjimku typu `SyntaxError`. Pokud data neobsahují OFN *Časová specifikace* (požadavek R1.02), knihovna vyhodí výjimku typu `TimeSpecificationNotFoundError`. Pokud struktura dat neodpovídá specifikaci OFN – chybí vlastnosti potřebné pro generování kalendářových událostí – je vyhozená výjimka typu `InvalidDataStructureError`.

4.1.2.2 Parametr `langCode`

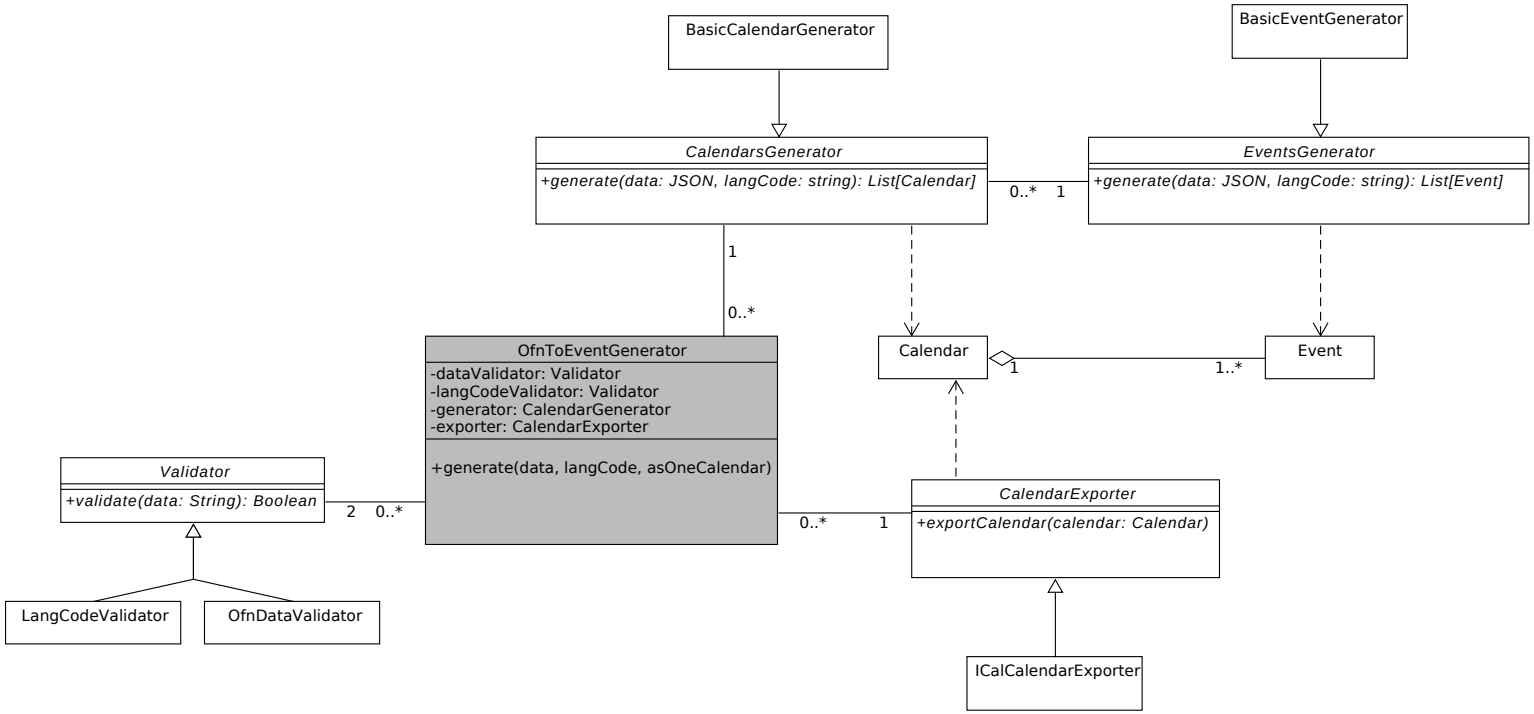
Parametrem `langCode` se určí jazyk, který má knihovna při generování využít. Je očekávána hodnota podle normy ISO 639-1 (např. `cs`, nebo `en`). V případě, že hodnota nebude této normě vyhovovat, knihovna vyhodí výjimku typu `InvalidLangCodeError` se zprávou „Language code ‚XY' not conformant with ISO 639-1." Pokud data neobsahují zvolený jazyk, vyhozená výjimka se zprávou „Language ‚XY' not found in data." je typu `LanguageNotFoundError`.

4.1.2.3 Parametr `asOneCalendar`

Rozhodnutí zda generovat právě jeden kalendář formátu iCalendar, nebo zvolit počet kalendářů podle vstupních dat, určuje parametr `asOneCalendar`. Hodnota `true` vynutí sloučení událostí do jednoho kalendáře. Výstupem bude pole obsahující jeden, nebo více řetězců, každý reprezentující jeden kalendář.

4.1.3 Diagram tříd

Diagram 4.1 ukazuje jednotlivé komponenty knihovny. Funkcionalitu hlavní třídy `OfnToEventGenerator` určují zvolené implementace rozhraní `Validator` (kontroluje korektnost předaných dat), `CalendarsGenerator` (provede samotné generování) a `CalendarExporter` (exportuje vygenerované informace (název, popis, čas, ...) do formátu iCalendar.



Obrázek 4.1: Diagram tříd knihovny

4.2 HTTP služba

HTTP služba zprostředkovává funkce knihovny pomocí HTTP API.

4.2.1 Technologie

Následuje popis technologií, které služba využívá.

4.2.1.1 Node.js

Node.js je prostředí umožňující spouštět JavaScriptový kód na serveru a asynchroně odpovídat na jednotlivé požadavky [24]. Použití Node.js umožňuje využít JavaScriptový kód knihovny a splnit tak požadavek R1.09 (viz kapitola 3.1.1) a požadavek R2.02 (viz kapitola 3.2.1).

4.2.1.2 Express

Express je framework pro tvorbu webových aplikací v Node.js [25], který je využívaný například společností IBM nebo Uber [26].

4.2.2 API

Application Programming Interface (API) je rozhraní, které udává, jakým způsobem bude služba volána a jak bude služba odpovídat. HTTP služba bude využívat bezestavové REST API.

4.2.2.1 REST

Representational State Transfer (REST) udává principy pro návrh webových služeb. Typicky je využíván protokol HTTP, který slouží pro přenos dat přes internet. [27]

4.2.2.2 Endpointy

Oba následující endpointy slouží pro získání kalendářových událostí vygenerovaných z dat, zaslaných v těle dotazu. Oba využívají metodu `POST`.

/kalendář/více
: Vrátí jeden, nebo více kalendářů s kalendářovými událostmi vygenerovanými z přijatých dat dle OFN. (Realizuje případ užití 3.2.2.2)

/kalendář/jeden
: Vrátí právě jeden kalendář s kalendářovými událostmi vygenerovanými z přijatých dat dle OFN. (Realizuje případ užití 3.2.2.1)

4.2.2.3 Vstupy

Vstup pro oba endpointy jsou data (dle OFN) ve formátu JSON-LD (například již v analýze zmíněný příklad 3.1). HTTP dotaz může vypadat například jako na ukázce 4.1.

Krom těla s vlastními daty je povinný také parametr `Accept-Language` v hlavičce dotazu, který udává požadovaný jazyk výsledných dat. Jde o obdobu parametru `langCode` (viz 4.1.2.2)

```
POST /kalendář/více HTTP/1.1
Content-Type: application/ld+json
Accept-Language: cs

{
  "@context": "https://ofn.gov.cz/události/2020-07-01/kontexty/událost.jsonld",
  "typ": "Událost",
  "iri": "https://data.brno.cz/zdroj/události/2019/vinobraní",
  "název": {
    "cs": "Brněnské vinobraní",
  },
  "popis": {
    "cs": "Město Brno Vás zve na již 21. vinobraní.",
  },
  "doba_trvání": [{
    "typ": "Časová specifikace",
    "časový_interval": {
      "typ": "Časový interval",
      "začátek": {
      "typ": "Časový okamžik",
        "datum": "2019-01-01"
      },
      "konec": {
        "typ": "Časový okamžik",
        "datum": "2019-01-07"
      }
    }
  }]
}
```

Výpis kódu 4.1: Ukázka HTTP dotazu

4.2.2.4 Výstupy

Server odpoví stavovým kódem `200 OK` a vygenerovanými daty (například ukázková odpověď 4.2). Typ dat se liší podle použitého endpointu. V případě špatného vstupu odpoví server stavovým kódem `400 Bad Request` a JSON objektem, který pod klíčem `error` obsahuje bližší vysvětlení chyby. Pokud není

typ těla požadavku `application/ld+json`, odpoví server stavovým kódem `415 Unsupported Media Type`.

/kalendář/více
: Výstupem jsou data jednoho, nebo více kalendářů ve formátu iCalendar. Jednotlivé kalendáře jsou předány jako seznam ve formátu JSON.

/kalendář/jeden
: Výstupem jsou data právě jednoho kalendáře ve formátu iCalendar.

```
HTTP/1.1 200 OK
Content-Type: text/calendar

BEGIN:VCALENDAR
VERSION:2.0
PRODID:https://gitlab.com/matous-dlabal/ofn-to-ical-lib
BEGIN:VEVENT
DTSTAMP:20210406T152117
UID:https://data.brno.cz/zdroj/události/2019/vinobraní#cs0
SUMMARY:Brněnské vinobraní
DESCRIPTION:Město Brno Vás zve na již 21. vinobraní.
DTSTART:20190101
DTEND:20190107
END:VEVENT
END:VCALENDAR
```

Výpis kódu 4.2: Ukázka HTTP odpovědi

4.3 Webová stránka

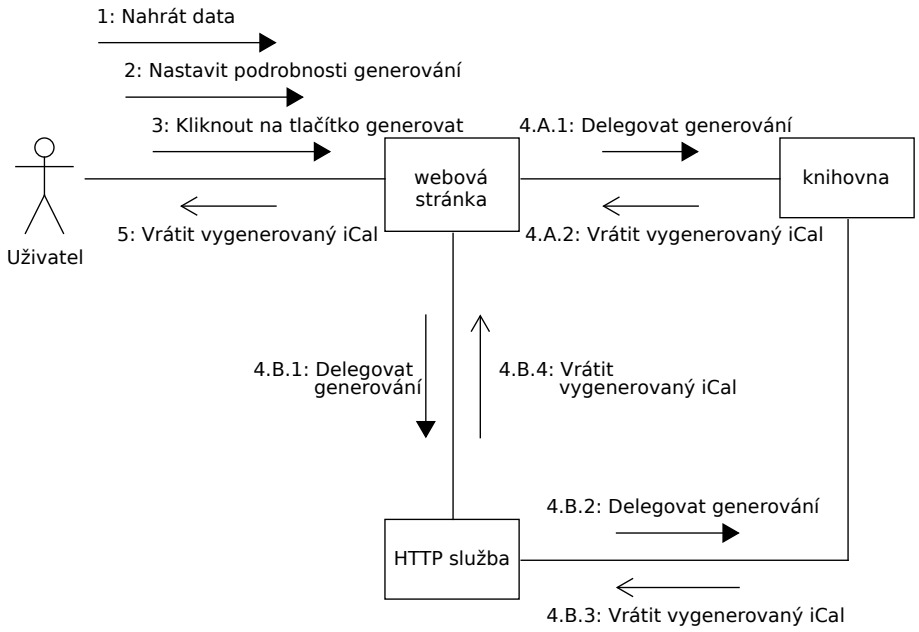
Webová stránka nabízí uživatelské rozhraní pro funkce knihovny. Využívá k tomu jak HTTP službu, tak i knihovnu samotnou. Komunikační diagram 4.2 ukazuje, že webová stránka samotná kalendářové události generovat nebude. Bude delegovat žádost o vygenerování kalendářových událostí buď knihovně (na diagramu označené jako A), nebo HTTP službě (označené písmenem B).

4.3.1 Technologie

Následuje popis technologií, které webová stránka využívá.

4.3.1.1 JavaScript

Stránka bude volat knihovnu i HTTP službu pomocí JavaScriptu (popsaný výše v kapitole 4.1.1.1).



Obrázek 4.2: Komunikační diagram webové stránky při žádosti o generování

4.3.1.2 Bootstrap

Bootstrap je framework pro usnadnění tvorby responzivního frontendu webových stránek. Byl vytvořen firmou Twitter a zveřejněn pod open-source licencí. [28]

4.3.1.3 Knihovna JSZip

Knihovna JSZip slouží, mimo jiné, k vytváření zip archivů [29]. Je zveřejněna pod dvěma open-source licencemi – *MIT License* a *GNU General Public License version 3* [29]. Knihovna je použita, protože scénář generování kalendářových událostí (kapitoly 3.3.2.3 a 3.4.2.3) určuje způsob stažení více kalendářů právě pomocí archivu souborů.

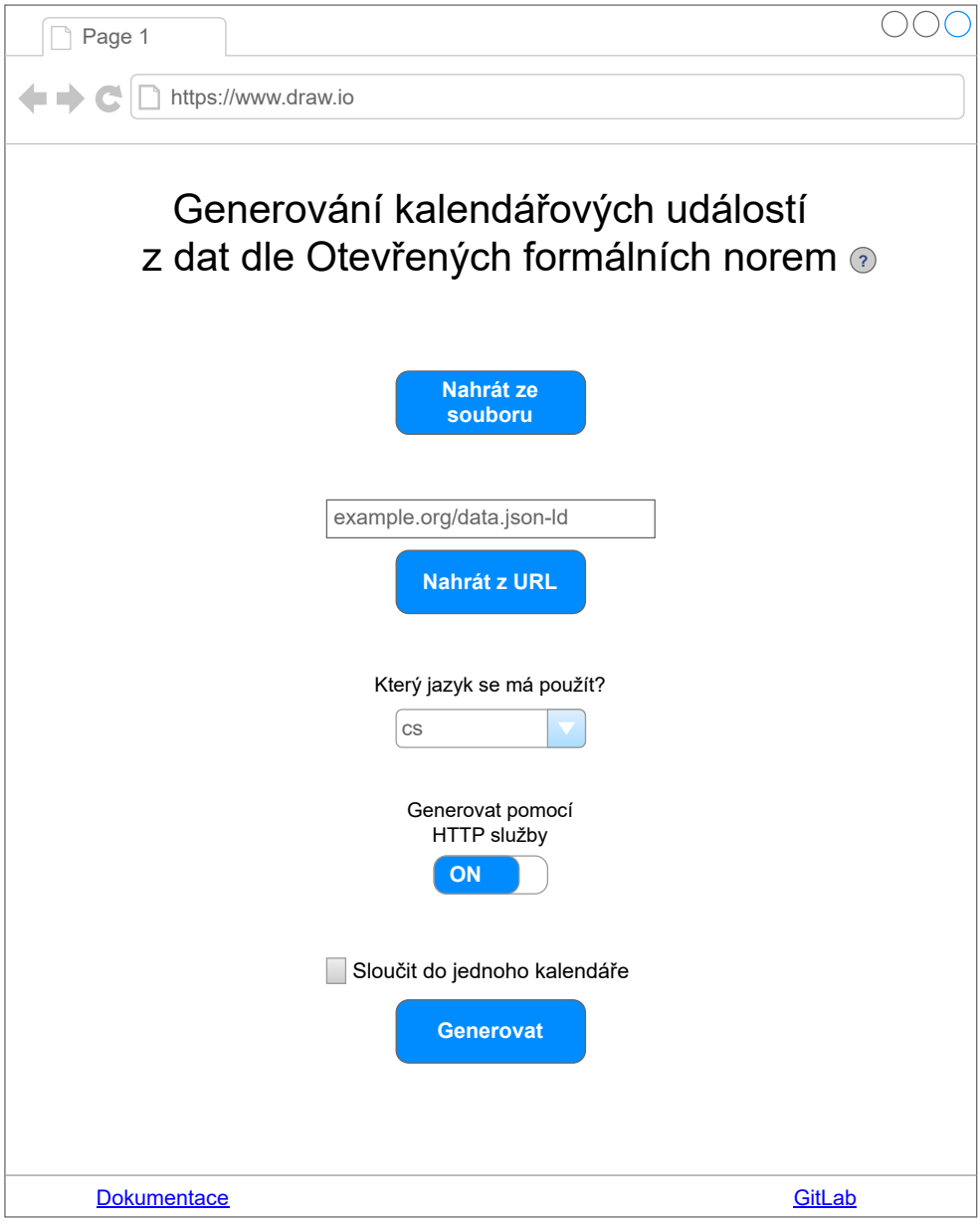
4.3.2 Mockup UI

Obrázek 4.3 s mockupem UI webu ukazuje, jak bude výsledná stránka vypadat. Především obsahuje tlačítka pro vybrání vstupních dat a pro jejich transformaci na kalendářové události. Patička obsahuje odkazy na dokumentaci a git repozitář. Otazník za nadpisem bude odkazovat na stránku s informacemi o projektu.

Tabulka 4.1 přiřazuje ovládacím prvkům příslušné požadavky (viz 3.3.1 a 3.4.1) a případy užití (popsané v kapitolách 3.3.2 a 3.4.2).

Ovládací prvky	Funkční požadavky	Případy užití (kapitoly)
Nahrát data ze souboru	R3.01, R4.01	3.3.2.1, 3.4.2.1
Nahrát z URL	R3.02, R4.02	3.3.2.1, 3.4.2.1
Generovat pomocí HTTP služby (Ano/Ne)	R3.03, R4.03	—
Sloučit do jednoho kalendáře (Ano/Ne)	—	3.3.2.2, 3.4.2.2
Generovat	R3.05, R4.05	3.3.2.3, 3.4.2.3

Tabulka 4.1: Tabulka ovládacích prvků UI



Obrázek 4.3: Mockup UI webové stránky

Kapitola 5

Implementace

V této kapitole jsou popsány odchylky implementace od analýzy a návrhu.

5.1 Knihovna

Při nahrání vygenerovaných kalendářových dat do aplikace Google Calendar se objevil problém pro opakující se události. Aplikace má limit 730 opakování, které se v kalendáři zobrazí [30]. To odpovídá 2 rokům každodenního opakování události. Na místo chybějícího začátku kalendářové události bylo původně zamýšleno zvolit počátek 20. století. Taková událost by se však v aktuálním roce vůbec nezobrazovala. Z tohoto důvodu kalendářové události, které nemají jinak určený začátek, začínají 1. ledna v roce generování události.

Výsledná implementace nepřevádí vnořené *časové specifikace*, tedy části výjimka a časová platnost. Jejich přítomnost ve vstupních datech generování událostí nebrání, ale výsledné kalendářové události jimi nejsou nijak ovlivněny.

5.2 Webová stránka

Výsledná podoba webové stránky se od té navrhované liší především přidáním ukázkových souborů, které jsou nahrané spolu s webovou stránkou na GitLab Pages. Vybráním ukázky se vyplní pole pro adresu vzdáleného souboru. Tato funkcionalita byla přidána pro snadnější vyzkoušení webové stránky, bez nutnosti hledání vstupních dat.

Kapitola 6

Testování

V této kapitole je popsáno testování jednotlivých implementačních částí práce.

6.1 Knihovna

Knihovna je testována pomocí unit testů, které se nacházejí ve složce `test/` v repozitáři knihovny. Testování probíhá pomocí nástroje Mocha. Celkový počet 405 testů pokrývá více než 99 % příkazů (statement coverage) a 97 % větví (branch coverage) kódu. Testy se spouští pomocí grunt úlohy test (případně coverage), která je blíže popsaná v kapitole 7.1.1.2.

6.2 HTTP služba

HTTP služba je testovaná pomocí nástroje Postman. Je testováno, že služba po zaslání správného dotazu odpoví podle očekávání a při špatných dotazech, kdy chybí tělo nebo parametr `Accept-Language`, odpoví patřičným status kódem.

Postman kolekce se všemi testy se nachází v repozitáři služby v souboru `./postman/ofn-cal-service.postman_collection.json`. Ve stejné složce se nachází i soubory prostředí – `Production.postman_environment.json` a `Staging.postman_environment.json` – které určují adresu API, která se bude při testech volat. Testy se dají spustit buď v aplikaci Postman [31], nebo pomocí CLI utility Newman [32], která se instaluje příkazem

```
npm install -g newman
```

Pomocí utility Newman se podle [32] testy spouští příkazem

```
newman run <soubor kolekce> -e <soubor prostředí>
```

6.3 Webová stránka

Testování webové stránky probíhá manuálně na základě testovacích scénářů. Scénáře jsou odvozeny ze scénářů případů užití z kapitoly analýza. Ve složce `src/examples/` repozitáře webové stránky se nachází soubory s daty dle OFN, které se dají při testování využít.

6.3.1 Testovací scénáře

Všechny scénáře jsou stejné v případě použití knihovny i v případě použití HTTP služby. Změna nastavení se provádí přepínačem „Generovat pomocí HTTP služby“, který je na obrázku 7.1 označený číslem 6. Pokud je ve scénáři zmíněn ovládací prvek číslem, je myšleno číslo právě z tohoto obrázku.

6.3.1.1 Generování událostí z dat z lokálního souboru

1. Uživatel klikne na tlačítko „Nahrát lokální soubor“.
2. Uživatel vybere soubor ve formátu JSON-LD.
3. Uživatel potvrdí výběr souboru.
4. Uživatel zvolí český jazyk (kód `cs`).
5. Uživatel nechá zaškrtávací pole „Sloučit do jednoho kalendáře“ nezaškrtnuté.
6. Uživatel klikne na tlačítko „Generovat“.
7. Uživatel vybere, kam chce uložit archiv s vygenerovanými kalendářovými událostmi. (Tento krok závisí na nastavení prohlížeče.)

6.3.1.2 Generování událostí z dat z internetu

1. Uživatel zadá do pole 3 URL adresu souboru s daty, která jsou ve formátu JSON-LD.
2. Uživatel klikne na tlačítko „Nahrát soubor z internetu“.
3. Uživatel zvolí český jazyk (kód `cs`).
4. Uživatel nechá zaškrtávací pole „Sloučit do jednoho kalendáře“ nezaškrtnuté.
5. Uživatel klikne na tlačítko „Generovat“.
6. Uživatel vybere, kam chce uložit archiv s vygenerovanými kalendářovými událostmi. (Tento krok závisí na nastavení prohlížeče.)

6.3.1.3 Vybrání neexistujícího souboru z internetu

1. Uživatel nechá pole 3 prázdné, nebo do něj zadá neplatnou URL adresu souboru.

2. Uživatel klikne na tlačítko „Nahrát soubor z internetu“.

3. Webová stránka zobrazí upozornění, že data nejdou nahrát.

6.3.1.4 Generování anglických událostí

1. Uživatel klikne na tlačítko „Nahrát lokální soubor“.

2. Uživatel vybere soubor ve formátu JSON-LD.

3. Uživatel potvrdí výběr souboru.

4. Uživatel zvolí anglický jazyk (kód `en`).

5. Uživatel nechá zaškrtávací pole „Sloučit do jednoho kalendáře“ nezaškrtnuté.

6. Uživatel klikne na tlačítko „Generovat“.

7. Uživatel vybere, kam chce uložit archiv s vygenerovanými kalendářovými událostmi. (Tento krok závisí na nastavení prohlížeče.)

6.3.1.5 Generování událostí sloučených do jednoho kalendáře

1. Uživatel klikne na tlačítko „Nahrát lokální soubor“.

2. Uživatel vybere soubor ve formátu JSON-LD.

3. Uživatel potvrdí výběr souboru.

4. Uživatel zvolí český jazyk (kód `cs`).

5. Uživatel označí zaškrtávací pole „Sloučit do jednoho kalendáře“.

6. Uživatel klikne na tlačítko „Generovat“.

7. Uživatel vybere, kam chce uložit soubor formátu iCalendar (koncovka `.ics`) s vygenerovanými kalendářovými událostmi. (Tento krok závisí na nastavení prohlížeče.)

6.3.1.6 Neplatná data

1. Uživatel klikne na tlačítko „Nahrát lokální soubor“.
2. Uživatel vybere soubor s neplatnými daty.
3. Uživatel potvrdí výběr souboru.
4. Uživatel zvolí český jazyk (kód `cs`).
5. Uživatel nechá zaškrtávací pole „Sloučit do jednoho kalendáře“ nezaškrtnuté.
6. Uživatel klikne na tlačítko „Generovat“.
7. Webová stránka zobrazí upozornění, že při generování nastal problém.

6.3.1.7 Nahrání ukázkového souboru

1. Uživatel vybere položku ze seznamu ukázkových souborů (prvek č. 9).
2. Uživatel klikne na tlačítko „Nahrát soubor z internetu“.
3. Pro žádnou položku by stránka neměla zobrazit chybovou hlášku. Soubor by se mělo pokaždé podařit nahrát.

Kapitola 7

Dokumentace

V této kapitole se nachází programátorská a uživatelská dokumentace k výstupům práce, které byly popsány v předchozích kapitolách. Pro některé části dokumentace bylo výhodnější zpracovat je v jiné, než čistě textové podobě, zde v práci. V takových případech je na ně odkázáno a jsou dostupné jednak online, jednak na přiloženém paměťovém médiu.

7.1 Programátorská dokumentace

Všechny tři praktické části práce (knihovna, HTTP služba a webová stránka) jsou napsané v jazyku JavaScript.

7.1.1 Knihovna

Kód knihovny je spravovaný verzovacím systémem git. Veřejný GitLab repozitář je dostupný na webových stránkách `https://gitlab.com/matous-dlabal/ofn-to-ical-lib/`. Pokud máte nainstalovaný git, lze podle [33] repozitář naklonoval pomocí příkazu

```
git clone git@gitlab.com:matous-dlabal/ofn-to-ical-lib.git
```

Pro vývoj knihovny je využíván npm balíček Grunt a jeho pluginy. Pro jejich instalaci je potřeba mít nainstalované npm [34]. Pro použití balíčku Grunt je potřeba nejdříve nainstalovat jeho textové rozhraní (CLI) [34]. To se podle [34] provede spuštěním příkazu

```
npm install -g grunt-cli
```

Grunt pluginy a další potřebné balíčky je možné podle [34] nainstalovat příkazem

```
npm install
```

7.1.1.1 Použití knihovny

Kód knihovny (sestavený grunt úlohou build) je dostupný na adresách

- `https://matous-dlabal.gitlab.io/ofn-to-ical-lib/build/ofn-to-ical-lib.js`,
- `https://matous-dlabal.gitlab.io/ofn-to-ical-lib/build/ofn-to-ical-lib.min.js`.

Knihovna samotná neobsahuje knihovnu ical.js, která je však pro generování kalendářových událostí nutná. V případě použití knihovny v rámci prostředí Node.js, je nutné mít nainstalovaný balíček ical.js. Pokud je knihovna používána webovou stránkou, je potřeba importovat ical.js buď z lokálního souboru, staženého z GitHub repozitáře knihovny, nebo z adresy `https://unpkg.com/ical.js`.

7.1.1.2 Grunt úlohy

Pro vývoj projektu jsou v souboru `Gruntfile.js` nastavené následující úlohy. Úlohy se spouštějí příkazem `grunt <název úlohy>` [34].

build Úloha build slouží k sestavení výsledného, uceleného JavaScriptového souboru s kódem knihovny. Výsledkem je následující souborová struktura:

```
build/
├── ofn-to-ical-lib.js ..... kód přeložený programem babel
├── ofn-to-ical-lib.js.map
├── ofn-to-ical-lib.min.js ..... minifikovaná verze
├── ofn-to-ical-lib.min.js.map
└── script-concat.js ..... sloučený kód knihovny
```

První se spustí plugin grunt-contrib-concat, který sloučí kód celé knihovny do jednoho souboru `script-concat.js`. Tento soubor je poté přeložen programem babel. Tím vznikne soubor `ofn-to-ical-lib.js`, který je následně pluginem grunt-contrib-uglify minifikován do souboru `ofn-to-ical-lib.min.js`. Do souborů ve složce `build/` je také vložena hlavička s licencí knihovny.

lint Úloha lint spouští statickou analýzu kódu. Ta je provedena programem ESLint. Jde o open source utilitu pro statickou analýzu kódu, která umožňuje najít problematické části kódu, syntaktické chyby a části, které nedodržují doporučené formátování [35]. Knihovna používá lehce modifikovaná pravidla *Google JavaScript Style Guide*, poskytovaná npm balíčkem eslint-config-google [36]. Například omezení délky řádků na 80 znaků je pouze varování a v souborech s testy se ignoruje úplně.

doc Pro vygenerování dokumentace kódu slouží úloha doc. Program JSDoc z dokumentačních komentářů z kódu vygeneruje statické HTML stránky [37]. Ty jsou poté k nalezení ve složce `doc/`.

test Testy spouští úloha test, která využívá plugin grunt-mocha-test. Soubory, ve kterých jsou testy definované, se nachází ve složce (a podsložkách) `test/` a mají příponu `.test.js`. Pro přípravu testovacího prostředí slouží soubor `test/helper.js`, ve kterém jsou načteny jednotlivé části kódu ze zdrojových souborů.

coverage Podobně, jako úloha test, funguje i úloha coverage. Využívaný plugin grunt-mocha-istanbul také využívá testovací framework Mocha [38, 39]. Rozdíl je v zobrazených informacích. Úloha coverage neposkytuje detailní informace o testech samotných, ale zato generuje procentuálně vyjádřený souhrn o pokrytí kódu testy a HTML stránku s podrobnějšími informacemi, detailně rozdělenými po jednotlivých souborech [40]. Za zdrojové soubory se berou všechny JavaScriptové soubory ve složce `src/` a jejích podsložkách. Report v podobě statických stránek je vygenerován do složky `coverage/lcov-report/` [40].

7.1.1.3 GitLab CI

Konfigurace pro GitLab CI se nachází v souboru `.gitlab-ci.yml` [41]. Průběh CI knihovny je rozdělen do tří fází (anglicky stage) – build, test a deploy. Fáze build slouží k sestavení výstupu a spouští grunt úlohy build a doc. Fáze test spouští úlohy coverage a lint. Coverage spouští testy a informuje, jak velká část kódu je pokryta testy. Lint spouští statickou analýzu kódu. Poslední fáze – deploy – zajistí, že se vygenerovaná dokumentace, report pokrytí kódu a výstup grunt úlohy build zpřístupní online, jako GitLab Pages. Spouští se pouze na větvi `master`. Výstup úloh build, doc a coverage je po dokončení dostupný ke stažení.

7.1.1.4 Dokumentace kódu

Dokumentace kódu se generuje příkazem `grunt doc`. Aktuální verze pro větev `master` je dostupná na adrese `https://matous-dlabal.gitlab.io/ofn-to-ical-lib/doc/`. Popis rozhraní funkce `generate` je k nalezení v kapitole návrh (4.1.2).

7.1.2 HTTP služba

Kód HTTP služby je spravovaný verzovacím systémem git. Veřejný GitLab repozitář je dostupný na webových stránkách `https://gitlab.com/matous-dlabal/ofn-to-ical-service/`.

Pokud máte nainstalovaný git, lze podle [33] repozitář naklonoval pomocí příkazu

```
git clone git@gitlab.com:matous-dlabal/ofn-to-ical-service.git
```

Pro správu závislostí je využíváno npm, které je potřeba mít nainstalované. Služba také používá balíček Grunt, který podle [34] vyžaduje instalaci CLI příkazem

```
npm install -g grunt-cli
```

Instalace všech závislostí projektu se podle [42] provádí příkazem

```
npm install
```

Pro lokální spuštění služby slouží příkaz

```
npm start
```

respektive `npm run start`. Pro vývoj se může hodit spouštět službu příkazem `npm run dev`, který využívá npm plugin nodemon. Tento plugin restartuje server pokaždé, když se změní obsah některého ze zdrojových souborů [43].

Služba je vytvořená pomocí frameworku Express a hostovaná na platformě Heroku. Kromě produkčního serveru na adrese `https://ofn-to-ical-service.herokuapp.com/` existuje i server `https://ofn-to-ical-service-staging.herokuapp.com/`, který slouží jako testovací prostředí.

7.1.2.1 Grunt úlohy

Pro vývoj projektu jsou v souboru `Gruntfile.js` nastavené následující úlohy. Úlohy se spouštějí příkazem `grunt <název úlohy>` [34].

lint Úloha lint spouští statickou analýzu kódu. Ta je provedena programem ESLint. Jde o open source utilitu pro statickou analýzu kódu, která umožňuje najít problematické části kódu, syntaktické chyby a části, které nedodržují doporučené formátování [35]. Knihovna používá lehce modifikovaná pravidla *Google JavaScript Style Guide*, poskytovaná npm balíčkem eslint-config-google [36].

7.1.2.2 API Dokumentace

API služby je zdokumentované podle OpenAPI Specification 3.0. Soubor se specifikací ve formátu YAML se jmenuje `openapi.yaml` a je uložený ve složce repozitáře služby. Vizualizovaná podoba dokumentace (pomocí nástroje Swagger UI) je dostupná na adrese `https://matous-dlabal.gitlab.io/ofn-to-ical-service/api-doc/`.

7.1.2.3 GitLab CI

Konfigurace pro GitLab CI se nachází v souboru `.gitlab-ci.yml` [41]. Služba má CI rozděleno na čtyři fáze – lint, doc, staging a production. Fáze lint spouští statickou analýzu kódu, prováděnou nástrojem ESLint. Fáze doc je nakonfigurována tak, aby se z `master` větve generovaly statické webové stránky s dokumentací API. Stránky samotné není potřeba nijak generovat, nachází se ve složce `api-doc/`. Spuštěním skriptu `update-api-doc.sh` se popis dokumentace v souboru `openapi.yaml` převede do formátu JSON a uloží ke zbytku stránek.

Zbylé dvě fáze slouží pro nasazení služby na servery Heroku a pro spuštění testů. Fáze production nasazuje službu z `master` větve, staging operuje nad stejnojmennou větví. Nasazení se provádí pomocí nástroje Dpl. Po nasazení se spouští postman testy. Ty jsou v obou fázích stejné, liší se jen adresou API.

7.1.3 Webová stránka

Kód webové stránky je spravovaný verzovacím systémem git. Veřejný GitLab repozitář je dostupný na webových stránkách `https://gitlab.com/matous-dlabal/ofn-to-ical-web/`. Pokud máte nainstalovaný git, lze podle [33] repozitář naklonoval pomocí příkazu

```
git clone git@gitlab.com:matous-dlabal/ofn-to-ical-web.git
```

Pro vývoj je využíván npm balíček Grunt a jeho pluginy. Pro jejich instalaci je potřeba mít nainstalované npm [34]. Pro použití balíčku Grunt je potřeba nejdříve nainstalovat jeho textové rozhraní (CLI) [34]. To se podle [34] provede spuštěním příkazu

```
npm install -g grunt-cli
```

Grunt pluginy a další potřebné balíčky je možné podle [34] nainstalovat příkazem

```
npm install
```

Zdrojové kódy stránky se nachází ve složce `src/`. Stránka je nastylovaná pomocí frameworku Bootstrap.

7.1.3.1 Grunt úlohy

Pro vývoj projektu jsou v souboru `Gruntfile.js` nastavené následující úlohy. Úlohy se spouštějí příkazem `grunt <název úlohy>` [34].

lint Úloha lint spouští statickou analýzu kódu. Ta je provedena programem ESLint. Jde o open source utilitu pro statickou analýzu kódu, která umožňuje najít problematické části kódu, syntaktické chyby a části, které nedodržují doporučené formátování [35]. Knihovna používá lehce modifikovaná pravidla *Google JavaScript Style Guide*, poskytovaná npm balíčkem eslint-config-google [36].

doc Pro vygenerování dokumentace kódu slouží úloha doc. Program JSDoc z dokumentačních komentářů z kódu vygeneruje statické HTML stránky [37]. Ty jsou poté k nalezení ve složce `doc/`.

7.1.3.2 GitLab CI

Konfigurace pro GitLab CI se nachází v souboru `.gitlab-ci.yml` [41]. Služba má CI rozděleno na dvě fáze – test a deploy. Fáze test spouští ESLint. Fáze deploy je nakonfigurována tak, aby se spouštěla jen nad větví `master`. Generuje dokumentaci JavaScritpu v podobě statických webových stránek, které přidává do složky `src` kterou publikuje na GitLab Pages.

7.1.3.3 Dokumentace kódu

Dokumentace kódu se generuje příkazem `grunt doc`. Aktuální verze pro větev `master` je dostupná na adrese `https://matous-dlabal.gitlab.io/ofn-to-ical-web/doc/`.

7.2 Uživatelská dokumentace

Tato kapitola slouží jako uživatelská dokumentace webové stránky a obsahuje popis jejího ovládání. Webová stránka je dostupná na adrese `https://matous-dlabal.gitlab.io/ofn-to-ical-web/`.

Obrázek 7.1 zobrazuje webovou stránku s očíslovanými ovládacími prvky. Prvek 1 je odkaz na stránku s informacemi o projektu. Dole v patičce jsou odkazy na jednotlivé GitLab repozitáře.

Pro generování kalendářových událostí je nejprve potřeba vybrat soubor se zdrojovými daty. Ta se dají nahrát buď z lokálního souboru (tlačítko 2), nebo ze souboru z internetu (tlačítko 4). Po stisknutí tlačítka 2 se otevře dialogové okno, ve kterém je možné vybrat lokální soubor. Aby se nahrála data ze vzdáleného zdroje, je potřeba před stisknutím tlačítka 4 nejprve zadat do pole 3 adresu vzdáleného souboru, nebo vybrat ukázku ze seznamu 9 (tím se také vyplní pole 3).

Druhá část stránky slouží k vybrání parametrů generování. Uživatel může v roletce 5 zvolit jazyk událostí, přepínačem 6 ovlivnit, zda generování proběhne lokálně, nebo dotazem na HTTP službu a zaškrtávacím polem 7 rozhodnout o sloučení vygenerovaných událostí do jednoho kalendáře.

Tlačítko 8 spouští generování a pokud vše proběhne úspěšně, zahájí stahování výsledného souboru. Typ stahovaného souboru závisí na volbě 7. Pokud je pole zaškrtnuté, je výsledkem generování jeden soubor formátu iCalendar s příponou `.ics`. Pokud uživatel pole nezaškrtl, stáhne se zip archiv s jedním, nebo více soubory formátu iCalendar. V případě chybějících zdrojových dat nebo problému při generování stránka zobrazí varování s informací, v čem nastal problém.

Obrázek 7.1: Screenshot webové stránky s očíslovanými ovládacími prvky

Závěr

Na začátku této práce byl formulován její cíl a jeho dílčí části. Čtenář byl nejprve seznámen s používanými termíny a byla mu podrobně představena otevřená formální norma *časová specifikace*. Dále byl seznámen s formátem iCalendar a s prostředky, které nabízí pro určení kalendářových událostí. Byla provedena analýza a návrh pro praktické části práce:

- knihovnu, generující kalendářové události z dat dle OFN;
- HTTP službu, která vystavuje REST API a interně závisí na knihovně;
- webovou stánku, která slouží jako grafické uživatelské rozhraní pro knihovnu, respektive HTTP službu.

V návrhu byly, mimo jiné, popsány zvolené technologie a představeny existující knihovny, nabízející hotová řešení pro části, na které se tato práce nezaměřuje. Tyto zanalyzované a navržené části byly implementovány, otestovány a v textu práce i částečně zdokumentovány. Na další části dokumentace, především dokumentace jednotlivých tříd a metod, je odkázáno a nachází se jednak v příloze práce, jednak ve veřejných GitLab repozitářích, které jsou dostupné online.

Cíl práce byl splněn, byla vytvořena knihovna, HTTP služba i webová stránka, pomocí kterých je možné generovat kalendářové události z dat dle OFN. Stále je však prostor pro vylepšení a rozšíření práce. Knihovna například při generování událostí nebere v potaz výjimku ani časovou platnost *časové specifikace*. Práce navíc odhalila, že některé části obecné *časové specifikace* nejsou pro převod do formátu iCalendar vhodné a význam takových částí je zaznamenán pouze do popisu kalendářové události. Některé, pro převod, problematické části (např. frekvence) by se do formátu iCalendar převést daly, vyžadovalo by to však dodání dalších dat, která by upřesnila původně vágní reprezentaci v *časové specifikaci*. Přestože *časová specifikace* sama umožňuje definovat takové části jiným, přesnějším způsobem, mohla by se práce rozšířit

i tímto směrem. Data dle OFN mohou být vícejazyčná a při generování se volí jazyk, který se má pro kalendářové události použít. Dodatečné poznámky do popisu události v toto chvíli existují jen v češtině a angličtině, pro ostatní jazyky se použije angličtina. Přestože jsou otevřené formální normy českou záležitostí, je doplnění dalších jazykových variant dalším z možných rozšíření práce. Uplatnit by se mohlo například pro turistické cíle, které leží v blízkosti hranic, kde mohou být další jazykové varianty žádoucí.

Literatura

1. GOOGLE. Import events to Google Calendar. In: *Calendar Help* [online]. Google, 2021 [cit. 2021-04-02]. Dostupné z: https://support.google.com/calendar/answer/37118?hl=en&ref_topic=10510645#zippy=%2Ccreate-or-edit-an-icalendar-file.
2. MICROSOFT. Share your calendar in Outlook on the web. In: *Microsoft Support* [online]. Microsoft, 2021 [cit. 2021-04-02]. Dostupné z: https://support.microsoft.com/en-us/office/share-your-calendar-in-outlook-on-the-web-7ecef8ae-139c-40d9-bae2-a23977ee58d5.
3. APPLE. Import a export kalendářů na Macu. In: *Uživatelská příručka pro Kalendář* [online]. Apple, 2021 [cit. 2021-04-02]. Dostupné z: https://support.apple.com/cs-cz/guide/calendar/icl1023/mac.
4. Zákon č. 106/1999 Sb., o svobodném přístupu k informacím. In: *Sbírka zákonů České republiky*. 1999. ISSN 1211-1244.
5. KLÍMEK, Jakub. Otevřené formální normy (OFN). In: *Portál otevřených dat* [online]. Ministerstvo vnitra České republiky, 2021 [cit. 2021-03-30]. Dostupné z: https://data.gov.cz/ofn/.
6. DVOŘÁK, Martin; SPÁL, Robert; MAREK, Jiří; KLÍMEK, Jakub. *OFN Turistické cíle* [online]. 2020 [cit. 2020-11-04]. Otevřená formální norma. Ministerstvo vnitra České republiky. Dostupné z: https://ofn.gov.cz/turistick%C3%A9-c%C3%ADle/2020-07-01/.
7. DVOŘÁK, Martin; SPÁL, Robert; MAREK, Jiří; KLÍMEK, Jakub. *OFN Sportoviště* [online]. 2020 [cit. 2020-11-04]. Otevřená formální norma. Ministerstvo vnitra České republiky. Dostupné z: https://ofn.gov.cz/sportovi%C5%A1t%C4%9B/2020-07-01/.
8. DVOŘÁK, Martin; SPÁL, Robert; MAREK, Jiří; KLÍMEK, Jakub. *OFN Časová specifikace* [online]. 2020 [cit. 2020-11-04]. Otevřená formální norma. Ministerstvo vnitra České republiky. Dostupné z: https://ofn.gov.cz/%C4%8Dasov%C3%A1-specifikace/2020-07-01/.

9. NEČASKÝ, Martin; KLÍMEK, Jakub. *OFN Číselníky* [online]. 2020 [cit. 2021-03-30]. Otevřená formální norma. Ministerstvo vnitra České republiky. Dostupné z: https://ofn.gov.cz/%C4%8D%C3%ADseln%C3%ADky/draft/.

10. ÚŘAD PRO PUBLIKACE EVROPSKÉ UNIE. *Time Period* [online]. 2019 [cit. 2021-04-02]. EU Vocabulary. Úřad pro publikace Evropské unie. Dostupné z: https://op.europa.eu/cs/web/eu-vocabularies/concept-scheme/-/resource?uri=http://publications.europa.eu/resource/authority/timeperiod.

11. ÚŘAD PRO PUBLIKACE EVROPSKÉ UNIE. *Frequency* [online]. 2019 [cit. 2021-04-09]. EU Vocabulary. Úřad pro publikace Evropské unie. Dostupné z: https://op.europa.eu/cs/web/eu-vocabularies/concept-scheme/-/resource?uri=http://publications.europa.eu/resource/authority/frequency.

12. DESRUISSEAUX, Bernard (ed.). *Internet Calendaring and Scheduling Core Object Specification (iCalendar)* [online]. 2009 [cit. 2020-10-20]. Request for Comments, RFC 5545. Dostupné z DOI: 10.17487/RFC5545.

13. PRIX, Ladislav. *Aplikace pro podporu otevírání dat o dřevinách pomocí otevřené formální normy.* Praha, 2020. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

14. KOPEČEK, Martin. otevrene-formalni-normy-dts. In: *GitHub* [online]. Otevřená data Ministerstva financí ČR, 2021-01-12 [cit. 2021-04-20]. Dostupné z: https://github.com/otevrena-data-mfcr/otevrene-formalni-normy-dts.

15. ŠKOP, Michal; NEČASKÝ, Martin. app-ofn-plakaty. In: *GitHub* [online]. Otevřená data ČR @ MVČR, 2021-03-17 [cit. 2021-04-20]. Dostupné z: https://github.com/opendata-mvcr/app-ofn-plakaty.

16. KLÍMEK, Jakub; STRAKA, Jakub. zobrazeni-dat-nkod-dle-ofn. In: *GitHub* [online]. Otevřená data ČR @ MVČR, 2021-03-15 [cit. 2021-04-20]. Dostupné z: https://github.com/opendata-mvcr/app-ofn-plakaty.

17. SPORNY, Manu; LONGLEY, Dave; KELLOGG, Gregg; LANTHALER, Markus; CHAMPIN, Pierre-Antoine; LINDSTRÖM, Niklas. *JSON-LD 1.1 – A JSON-based Serialization for Linked Data* [online]. 2020 [cit. 2020-10-20]. W3C recommendation. Dostupné z: https://www.w3.org/TR/json-ld11/.

18. DVOŘÁK, Martin; SPÁL, Robert; MAREK, Jiří; KLÍMEK, Jakub. *OFN Věc* [online]. 2020 [cit. 2020-11-09]. Otevřená formální norma. Ministerstvo vnitra České republiky. Dostupné z: https://ofn.gov.cz/v%C4%9Bc/2020-07-01/.

19. DVOŘÁK, Martin; SPÁL, Robert; MAREK, Jiří; KLÍMEK, Jakub. *OFN Události* [online]. 2020 [cit. 2020-11-04]. Otevřená formální norma. Ministerstvo vnitra České republiky. Dostupné z: `https://ofn.gov.cz/ud%C3%A1losti/2020-07-01/`.

20. ISO 639-1:2002. *Codes for the representation of names of languages – Part 1: Alpha-2 code*. ICS 01.140.20. Ženeva, Švýcarsko: International Organization for Standardization, 2002.

21. Meteorological Versus Astronomical Seasons. In: *National Centers for Environmental Information* [online]. National Centers for Environmental Information, 2021-03-18 [cit. 2021-04-23]. Dostupné z: `https://www.ncei.noaa.gov/news/meteorological-versus-astronomical-seasons`.

22. BENGTSSON, Peter; SCHONNING, Nick. JavaScript. In: *MDN Web Docs* [online]. Mozilla, 2020 [cit. 2021-01-14]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/javascript`.

23. KEWISCH, Philipp; LAL, James; SCHRÖDER, Martin et al. ical.js. In: *GitHub* [online]. Thunderbird, 2020 [cit. 2021-01-14]. Dostupné z: `https://github.com/mozilla-comm/ical.js`.

24. ROGERS, Mikeal et al. About Node.js. In: *Node.js* [online]. 2020 [cit. 2021-01-14]. Dostupné z: `https://nodejs.org/en/about/`.

25. YAAPA, Hage; MCKINNEY, Rand et al. Express – Node.js web application framework. In: *Express* [online]. expressjs, 2019 [cit. 2021-05-09]. Dostupné z: `https://expressjs.com/`.

26. WICLIFF, Praveen; MCKINNEY, Rand et al. Companies using Express. In: *Express* [online]. expressjs, 2020 [cit. 2021-05-09]. Dostupné z: `https://expressjs.com/en/resources/companies-using-express.html`.

27. RODRIGUEZ, Alex. RESTful Web services. In: *IBM Developer* [online]. IBM, 2015 [cit. 2021-01-14]. Dostupné z: `https://developer.ibm.com/technologies/web-development/articles/ws-restful`.

28. OTTO, Mark; POUPARD, Gaël et al. About. In: *Bootstrap v5.0* [online]. Bootstrap, 2020 [cit. 2021-01-25]. Dostupné z: `https://getbootstrap.com/docs/5.0/about/overview/`.

29. *JSZip* [online]. Stuart Knightley, 2019 [cit. 2021-03-09]. Dostupné z: `https://stuk.github.io/jszip/`.

30. NEIL@GCALTOOLKIT. Older Recurring Events Disappear on Import from .ics. In: *Calendar Help Community* [online]. 2020-08-19 [cit. 2021-05-07]. Dostupné z: `https://support.google.com/calendar/thread/65547353/older-recurring-events-disappear-on-import-from-ics?hl=en`.

31. SMITH, Sue; RAJAN, Sowmya; DENYER, Tristan. Using the Collection Runner. In: *Postman Learning Center* [online]. Postman, 2021-04-29 [cit. 2021-05-08]. Dostupné z: `https://learning.postman.com/docs/running-collections/intro-to-collection-runs/`.

32. SMITH, Sue; RILEY, Claire; DENYER, Tristan; RAJAN, Sowmya. Running collections on the command line with Newman. In: *Postman Learning Center* [online]. Postman, 2021-04-30 [cit. 2021-05-08]. Dostupné z: `https://learning.postman.com/docs/running-collections/using-newman-cli/command-line-integration-with-newman/`.

33. CHACON, Scott et al. Git – git-clone Documentation. In: *Git Documentation* [online]. Software Freedom Conservancy, 2021 [cit. 2021-04-06]. Dostupné z: `https://git-scm.com/docs/git-clone`.

34. ALMAN, Ben et al. Getting started. In: *Grunt* [online]. 2017 [cit. 2021-03-26]. Dostupné z: `https://gruntjs.com/getting-started`.

35. GP, Sudarsan; EDDY, Ronald Jr.; PARTINGTON, Kevin; ZAKAS, Nicholas C. About. In: *ESLint* [online]. OpenJS Foundation, 2017 [cit. 2021-04-06]. Dostupné z: `https://eslint.org/docs/about/`.

36. SORHUS, Sindre; WALTON, Philip; OSMANI, Addy; BERGER, Florian. eslint-config-google. In: *GitHub* [online]. Google, 2017 [cit. 2021-04-12]. Dostupné z: `https://github.com/google/eslint-config-google/blob/master/README.md`.

37. COLTER, Matthew; WILLIAMS, Jeff; MATHEWS, Michael. Getting Started with JSDoc 3. In: *Use JSDoc* [online]. JSDoc, 2019 [cit. 2021-04-12]. Dostupné z: `https://jsdoc.app/about-getting-started.html`.

38. HALLIDAY, Peter et al. grunt-mocha-test. In: *GitHub* [online]. Peter Halliday, 2020 [cit. 2021-04-12]. Dostupné z: `https://github.com/pghalliday/grunt-mocha-test`.

39. BELOV, Sergey; CESAR, Paulo et al. grunt-mocha-istanbul. In: *GitHub* [online]. Paulo Cesar, 2016 [cit. 2021-04-12]. Dostupné z: `https://github.com/pocesar/grunt-mocha-istanbul`.

40. CESAR, Paulo. *grunt-mocha-istanbul 5.0.2* [software]. 2016-08-07 [cit. 2021-04-13]. Dostupné z: `https://github.com/pocesar/grunt-mocha-istanbul`.

41. AMIRAULT, Marcel; SELHORN, Suzanne et al. GitLab CI/CD. In: *GitLab Docs* [online]. GitLab, 2021-02-17 [cit. 2021-04-13]. Dostupné z: `https://docs.gitlab.com/ee/ci/`.

42. npm-install. In: *npm Docs* [online]. npm, 2020 [cit. 2021-04-20]. Dostupné z: `https://docs.npmjs.com/cli/v7/commands/npm-install`.

43. SHARP, Remy et al. nodemon. In: *GitHub* [online]. Remy Sharp, 2021-03-31 [cit. 2021-04-20]. Dostupné z: `https://github.com/remy/nodemon`.

Příloha A

Seznam použitých zkratek

API Application Programming Interface.

CLI Command Line Interface.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

JSON JavaScript Object Notation.

JSON-LD JavaScript Object Notation for Linked Data.

OFN Otevřená formální norma.

REST Representational State Transfer.

UI User Interface.

URL Uniform Resource Locator.

UTC Koordinovaný světový čas.

YAML YAML Ain't Markup Language.

Příloha B

Obsah přiloženého CD

- `README` ..... stručný popis obsahu CD
- `src/`
  - `ofn-to-ical-lib/` ..... složka knihovny
  - `ofn-to-ical-service/` ..... složka HTTP služby
  - `ofn-to-ical-web/` ..... složka webu
- `thesis/` ..... text práce
  - `src/` ..... zdrojová forma práce ve formátu LaTeX
  - `bakalarska-prace-matous-dlabal.pdf` .. text práce ve formátu PDF