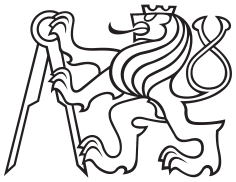
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Uživatelské rozhraní pro správu medicínské ontologie SNOMED CT

Snow UI

Filip Štěpánek

Vedoucí: Ing. Petr Křemen, Ph.D.
Studijní program: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Štěpánek** Jméno: **Filip** Osobní číslo: **483820**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Uživatelské rozhraní pro správu medicínské ontologie SNOMED

Název bakalářské práce anglicky:

User interface for SNOMED ontology management

Pokyny pro vypracování:

SNOMED-CT je mezinárodní standard pro klinické kódování nemocí, procedur, částí těla a dalších konceptů v oblasti zdravotní péče. Jeho složitost je jedním z hlavních problémů bránících širší adopci. Cílem této práce je navrhnout a implementovat systém pro správu, prohlížení a veřejné sdílení vlastního rozšíření SNOMED-CT ontologie:
1) Seznamte se se standardem SNOMED-CT, a existujícími nástroji pro správu rozšíření SNOMED-CT. Zvolte vhodný nástroj.
2) Navrhněte uživatelské rozhraní vhodné pro laiky pro správu, prohlížení, a sdílení vlastního SNOMED-CT.
3) Implementujte návrh z předchozího bodu jako webovou aplikaci ve vhodné JavaScriptovém frameworku.
4) Navrhněte testovací scénáře a testy proveďte s potenciálními uživateli.

Seznam doporučené literatury:

[1] SNOMED-CT, online, http://www.snomed.org/snomed-ct/education
[2] SnowOwl server, online https://github.com/b2ihealthcare/snow-owl
[3] SnowStorm server, online https://github.com/IHTSDO/snowstorm

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Křemen, Ph.D., skupina znalostních softwarových systémů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **11.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Petr Křemen, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Děkuji panu Ing. Petru Křemenovi, Ph.D. za asistenci při vypracování této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 21. května 2021

Abstrakt

Tato bakalářská se zabývá vývojem uživatelského rozhraní pro procházení terminologie SNOMED CT, což je lékařská terminologie sestávající se z více než 350 000 vzájemně provázaných pojmů. Uplatnění nalézá zejména v elektronických lékařských záznamech (EHR). Pro účely vývoje uživatelského rozhraní je nasazena na terminologickém serveru Snowstorm. Vyvíjená webová aplikace umožňuje procházení, vyhledávání a autorskou činnost v uživatelsky přívětivé a jednoduché formě.

Klíčová slova: snomed-ct, snomed, ui, snow-ui, nástroj

Vedoucí: Ing. Petr Křemen, Ph.D.
Praha 2,
Karlovo náměstí 13,
E-116

Abstract

This bachelor thesis deals with the development of a user interface for browsing the terminology of SNOMED CT, which is a medical terminology consisting of more than 350,000 interrelated terms. It is used mainly in electronic medical records (EHR). For user interface development, it is using the Snowstorm terminology server. The developed web application enables browsing, searching and authoring activities in a user-friendly and simple manner.

Keywords: snomed-ct, snomed, ui, snow-ui, tool

Title translation: User interface for SNOMED CT ontology management — Snow UI

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

1.1 Medicínské terminologie

Medicínská terminologie je formální, strojově čitelný jazyk, který slouží k detailnímu popisu klinických nálezů, nemocí a lidského těla, včetně jeho částí, procesů, podmínek, které jej ovlivňují a procedur na něm provedených. [4]

V porovnání s prostým textem zaručuje medicínská terminologie jednoznačnost sdělení a díky tomu zlepšuje a usnadňuje komunikaci mezi zdravotníky. Toto zároveň pozitivně ovlivňuje bezpečí pacientů pomocí mitigace rizika nedorozumění mezi personálem.

Medicínská terminologie má relativně pravidelné tvarosloví, stejné předpony a přípony se používají k přidání významů různým slovním kořenům. [4] Kořen slova často odkazuje na orgán, tkáň nebo stav. Například u poruchy hypertenze předpona „hyper-“ znamená „vysoká“ nebo „nad“ a kořen slova „tenze“ označuje tlak, takže slovo „hypertenze“ označuje abnormálně vysoký krevní tlak. Kořeny, předpony a přípony jsou často odvozeny z řečtiny nebo latiny a často se velmi liší od jejich anglických jazykových variant. Toto pravidelné tvarosloví znamená, že jakmile se naučíte přiměřený počet morfémů, bude snadné pochopit velmi přesné pojmy sestavené z těchto morfémů. Značná část lékařského jazyka je anatomická terminologie, která se týká názvů různých částí těla.

1.2 Digitalizace medicínské terminologie

Hlavní motivací pro digitalizaci medicísnké terminologie byla snaha zautomazitozovat interpretaci dat v informačních systémech, umožnit jejich strojové

porozumění a zprostředkovat vyhledavatelnost a integrovanost s podobnými daty. Dalším benefitem je snížení nejednoznačnosti těchto dat.

Rozsáhlé možnosti využití takové terminologie nalézají zejména v EHR (Electronic health record) systémech, které jsou elektronickou alternativnou klasické zdravotní karty. [19] Takto digitalizované záznamy nabízejí nejenom jednoznačnou a snadno přenositelnou formu všech záznamů o pacientovi, ale díky své elektronické čitelnosti i množství funkcí, jakými jsou například automatické upozornění na kontraindikace, aj.

Největší a nejstrukturovanější standardizovanou elektronickou terminologií zdravotní péče je SNOMED CT (Systemized nomenclature of medicine clinical terms). [21]

1.3 SNOMED CT

SNOMED CT je systematicky organizovaná, počítačově zpracovatelná sbírka lékařských termínů poskytující kódy, termíny, synonyma a definice používané v klinické dokumentaci a hlášení. [3, 20] SNOMED CT je považován za nejkomplexnější vícejazyčnou terminologii klinické zdravotní péče na světě. Primárním účelem SNOMED CT je kódování významů používaných ve zdravotnických informacích a podpora účinného klinického záznamu dat s cílem zlepšit péči o pacienta. SNOMED CT poskytuje základní obecnou terminologii pro elektronické zdravotní záznamy. Komplexní pokrytí SNOMED CT zahrnuje: klinické nálezy, příznaky, diagnózy, postupy, tělesné struktury, organismy a jiné etiologie, látky, léčiva, přístroje a vzorky. Výchozím jazykem je angličtina.

Příkladem může být například koncept reprezentující bolest hlavy. Tento konkrétní koncept má identifikátor `25064002`, úplný název `Headache (finding)` a několik synonym např. `Headache` nebo `Head pain`. Zároveň disponuje třemi vztahy `(IS-A) Head finding`, `(IS-A) Pain finding at anatomical site` a `(FINDING SITE) Head structure`. [15]

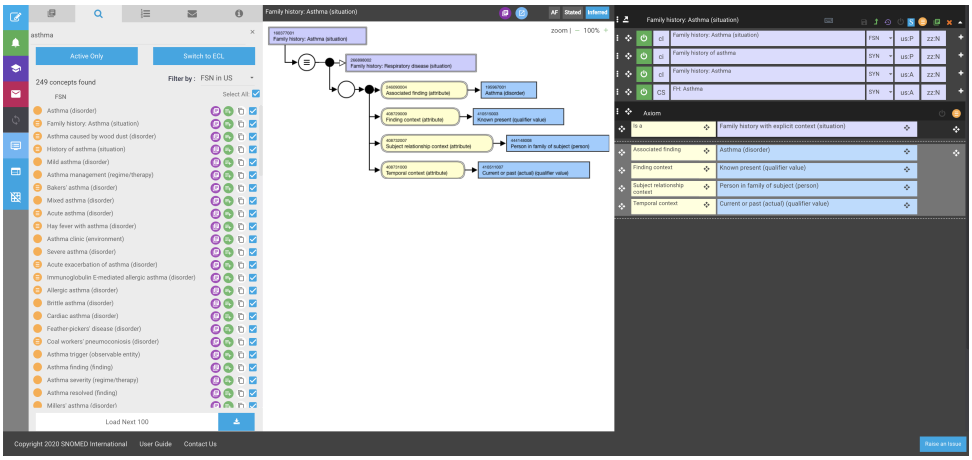
1.4 Tvorba rozšíření

SNOMED CT podporuje tvorbu vlastních rozšíření. Zjednodušeně řečeno můžeme díky tomuto přidávat nové pojmy a rozšiřovat ty stávající například o nové překlady, či mapování na další terminologické systémy.

Pro tyto účely existuje několik nástrojů, které lze k tomuto využít.

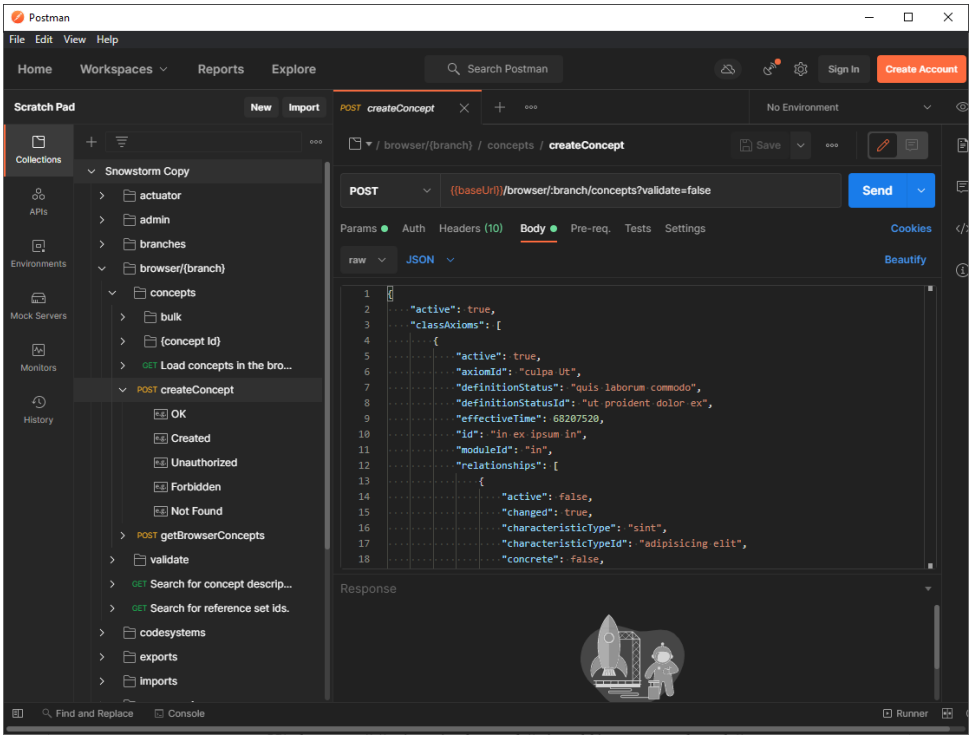
První z nabízených možností je nástroj oficiálně poskytovaný společností IHTSDO, SNOMED International Authoring Platform. [31] K jeho použití oficiální cestou je zapotřebí disponovat mezinárodním SNOMED účtem, který je poskytován členům IHTSDO a partnerským národním organizacím. [35] Pro běžného uživatele je tedy nedostupný.

Alternativní možností je nasadit si vlastní instanci Authoring Platform, protože její kódy jsou volně dostupné na internetu. [14] Zde jsem však i jako pokročilý uživatel znalý problematiky narazil na vysokou obtížnost provedení takovéto instalace.



Obrázek 1.1: SNOMED International Authoring Platform [31]

Druhou možností je použití čistě jen terminologického serveru (např. Snowstorm [13]) obsahující SNOMED CT a využití poskytovaného API například skrze nástroj Postman. [23]



Obrázek 1.2: Snowstorm API v aplikaci Postman

Všechna tato řešení mají jeden společný problém a to vysokou složitost a náročnou obsluhu. Toto značně komplikuje i případně seznámení s touto terminologií a může odradit spoustu potencionálních uživatelů.

1.5 Cíl

Mým cílem je vytvoření webové aplikace, která bude sloužit k procházení a editaci terminologie SNOMED CT nasazené na terminologickém serveru v co možná nejjednoduší, uživatelsky přívětivé formě.

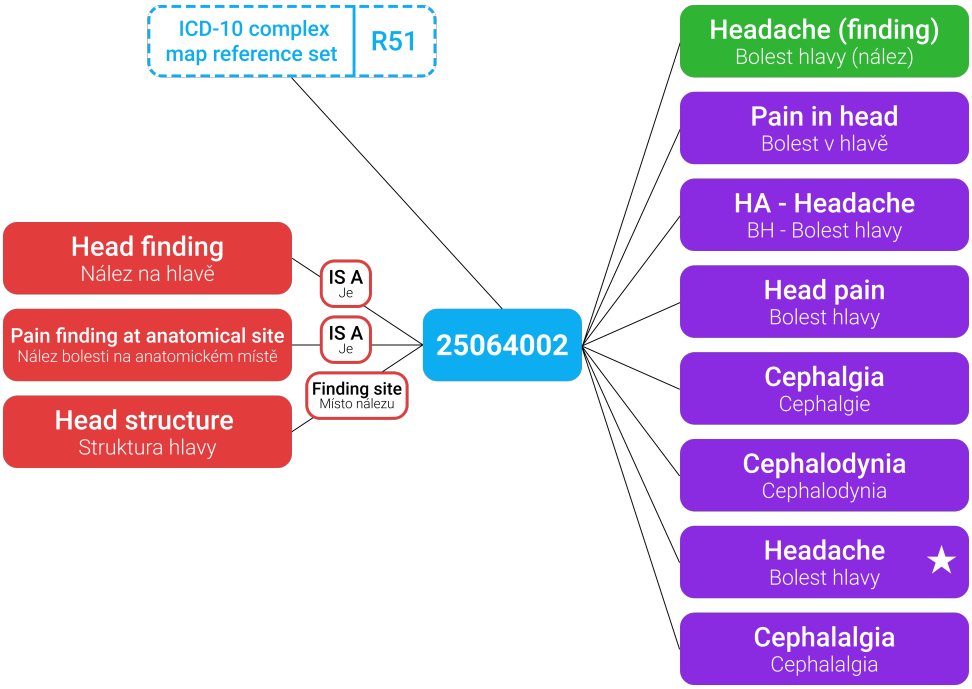
Na rozdíl od Authoring Platform bude aplikace poskytovat jednoduché a přehledné uživatelské rozhraní. Zároveň ji bude možné provozovat nezávisle na vlastním hardwaru a nebude vyžadovat přihlašování.

Kapitola 2

SNOMED CT

V následující kapitole detailně rozeberu terminologii SNOMED CT. Nastíním její historii, vývoj a logickou strukturu.

V celé této kapitole budu pro lepší představu používat jako ukázku již výše zmíněný koncept `25064002 |Headache (finding)|`, jehož vizualizaci najdete na obrázku 2.1. Tento koncept reprezentuje bolest hlavy. Prozatím jej stačí chápat intuitivně, jednotlivé části budou vysvětleny v následujících odstavcích. České názvy nejsou součástí terminologie SNOMED CT, jedná se o mé vlastní ilustrační překlady pro čtenáře bez znalosti anglického jazyka.



Obrázek 2.1: Příklad konceptu Headache (finding)

2.1 Použití

Primárním účelem SNOMED CT je kódování významů používaných ve zdravotnických informacích a podpora účinného klinického záznamu dat s cílem zlepšit péči o pacienta. SNOMED CT poskytuje základní obecnou terminologii pro elektronické zdravotní záznamy (EHR). Pokrytí SNOMED CT je komplexní, zahrnuje: klinické nálezy, příznaky, diagnózy, postupy, tělesné struktury, organismy a jiné etiologie, látky, léčiva, přístroje a vzorky.

SNOMED CT zajišťuje konzistentní výměnu informací a je základem interoperabilního elektronického zdravotního záznamu. Poskytuje komplexní prostředky k indexování, ukládání, načítání a agregaci klinických dat napříč nemocnicemi a ordinacemi. Pomáhá také při organizování obsahu systémů elektronických zdravotních záznamů tím, že snižuje variabilitu způsobu, jakým jsou data shromažďována, kódována a používána pro klinickou péči o pacienty a pro výzkum. [25] SNOMED CT lze použít k přímému záznamu klinických údajů o pacientech do jejich elektronických záznamů. Rovněž poskytuje uživateli řadu vazeb na cesty klinické péče, plány sdílené péče a další zdroje znalostí, aby usnadnil informované rozhodování a podporoval dlouhodobou péči o pacienta.

SNOMED CT je terminologie, kterou lze namapovat na jiné mezinárodními standardy a klasifikace. Mezi již existující řešení se řadí například mapování na terminologii ICD-10 [22], případně na standard LOINC [17].

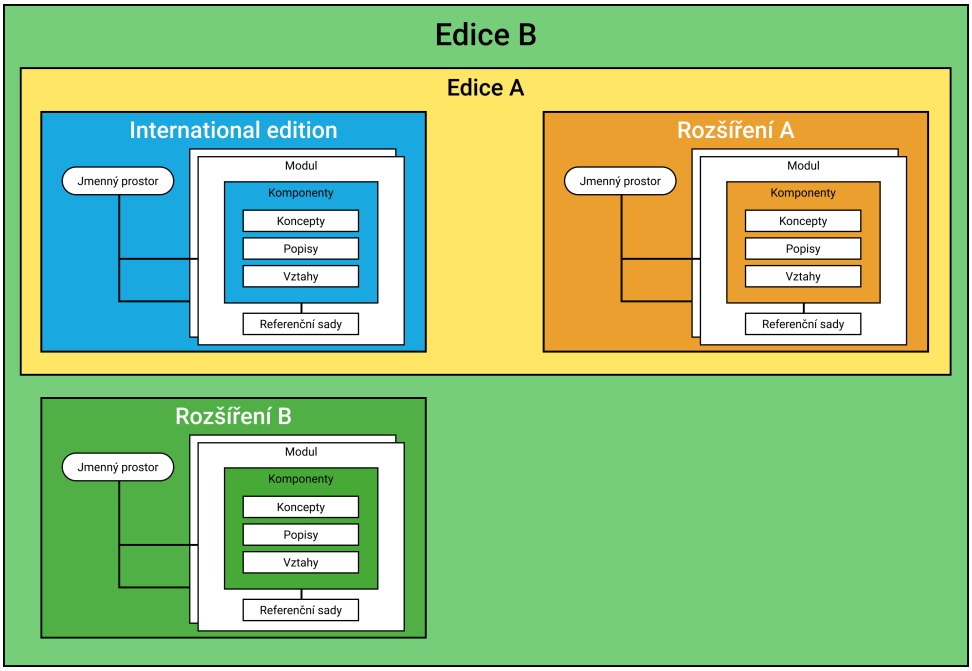
2.2 Logická struktura

Logická struktura SNOMED CT je velmi komplexní. V následujících odstavcích ji rozeberu od nejnižší úrovně po nejvyšší.

Na obrázku 2.2 si můžete prohlédnout logický model terminologie SNOMED CT.

Základními stavebními bloky terminologie SNOMED CT jsou komponenty, o nichž se více dočtete v podkapitole 2.3. Komponenty lze chápat jako jednotlivé stavební kostky, jejímž vzájemným spojením definujeme koncepty. Konceptům lze dále přidávat další význam pomocí referenčních sad. Jejich princip blíže představuje podkapitola 2.4.

Tyto koncepty a referenční sady dále logicky sdružujeme do modulů, jimž se věnuje podkapitola 2.6. Moduly slouží k jasnější a jednodušší organizaci komponent a referenčních sad, které pak rovněž mohou být závislé na komponentech a referenčních sadách z jiných modulů. Hovoříme poté o závislosti takovýchto modulů.



Obrázek 2.2: SNOMED CT logický model

S moduly jsou úzce spjaty jmenné prostory, které umožňují identifikovat autora komponent. Podrobněji se jimi zabývá podkapitola 2.5.

Dalším velmi důležitým pojmem jsou edice. Ty seskupují komponenty a referenční sady, které již lze samostatně provozovat a používat. Detaily se nachází v podkapitole 2.8.

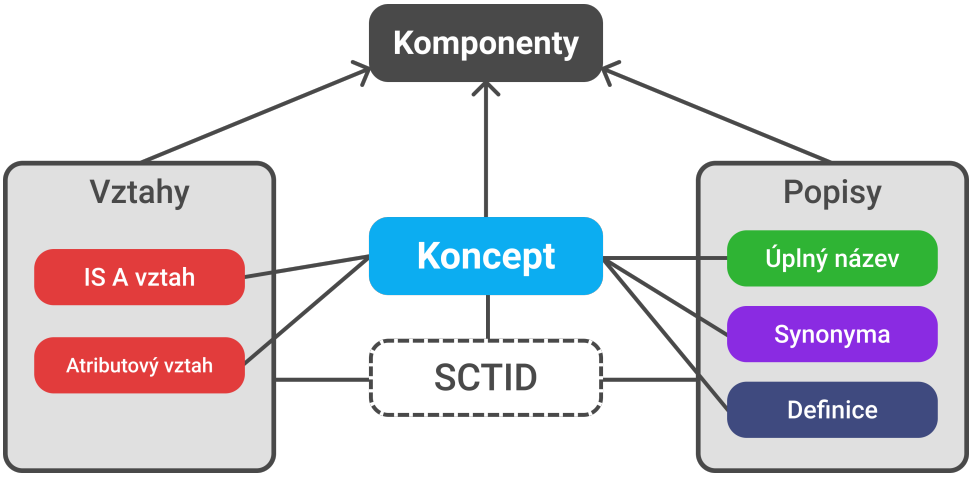
Nejdůležitější edicí je SNOMED CT International edition, která obsahuje všechny základní pojmy a je spravována společností IHTSDO. Námi zvolený ukázkový koncept `25064002 |Headache (finding)|` je součástní právě SNOMED CT International edition.

Další edice je možné vytvářet spojením edic a rozšíření (viz. podkapitola 2.7). Jednotlivá rozšíření a International edition se skládají právě z modulů.

2.3 Komponenty

Logický model SNOMED CT se skládá z komponent třech hlavních typů, kterými jsou koncept (concept), popis (description) a vztah (relationship).

Na obrázku 2.3 si můžete prohlédnout vzájemné propojení jednotlivých typů komponent.

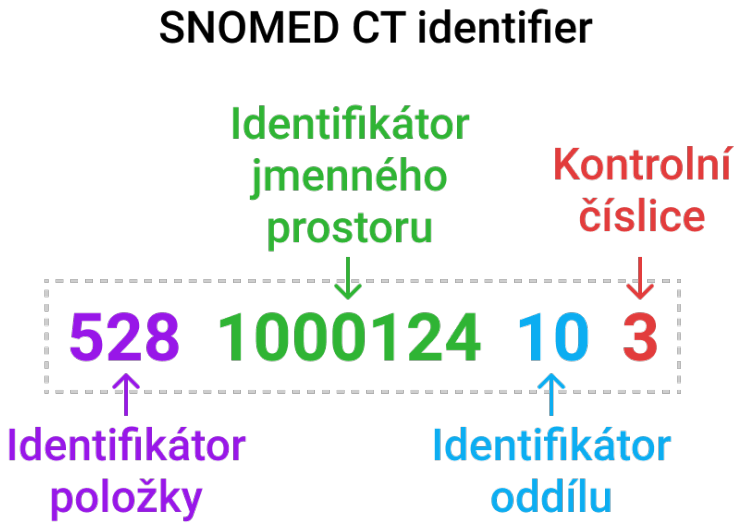


Obrázek 2.3: SNOMED CT Propojení komponent

Sdílené atributy

Komponenty mají své atributy, kterými jsou definovány. Některé atributy jsou sdíleny všemi typy komponent. [29]

Id. Každá komponenta disponuje identifikátorem, který je globálně unikátní napříč celou terminologií a všemi rozšířeními. Identifikátor označujeme jako `SNOMED CT identifier (SCTID)` a jeho struktura je striktně definovaná (viz. obrázek 2.4).



Obrázek 2.4: SNOMED CT identifier (SCTID)

effectiveTime reprezentuje datum ve formátu ISO 8601 (YYYMMDD), kdy byla publikována konkrétní verze komponenty. Pakliže je při tvorbě rozšíření přidána nová nebo modifikována existující komponenta, hodnota

Část	Pravidla
Identifikátor položky	Délka: 1-8 číslic Přiřazován autorem rozšíření Musí být unikátní v rámci jmenného prostoru a typu komponenty
Identifikátor jmenného prostoru	Délka: 7 číslic Přiřazován společností IHTSDO autorovi rozšíření Vynechán u komponent vytvořených společností IHTSDO Více informací viz. kapitola 2.5
Identifikátor oddílu	Délka: 2 číslice Hodnoty: 10 - koncept v rozšíření 11 - popis v rozšíření 12 - vztah v rozšíření
Kontrolní číslice	Délka: 1 číslice Vypočítán pomocí Verhoeffova algoritmu z ostatních číslic [28]

Tabulka 2.1: Popis částí SCTID

atributu bude odpovídat datu vydávané verze.

active je boolenská hodnota, která říká, zda-li je daná komponenta aktivní v daném efektivním čase. Hodnota 1 reprezentuje aktivní komponentu, 0 pak neaktivní.

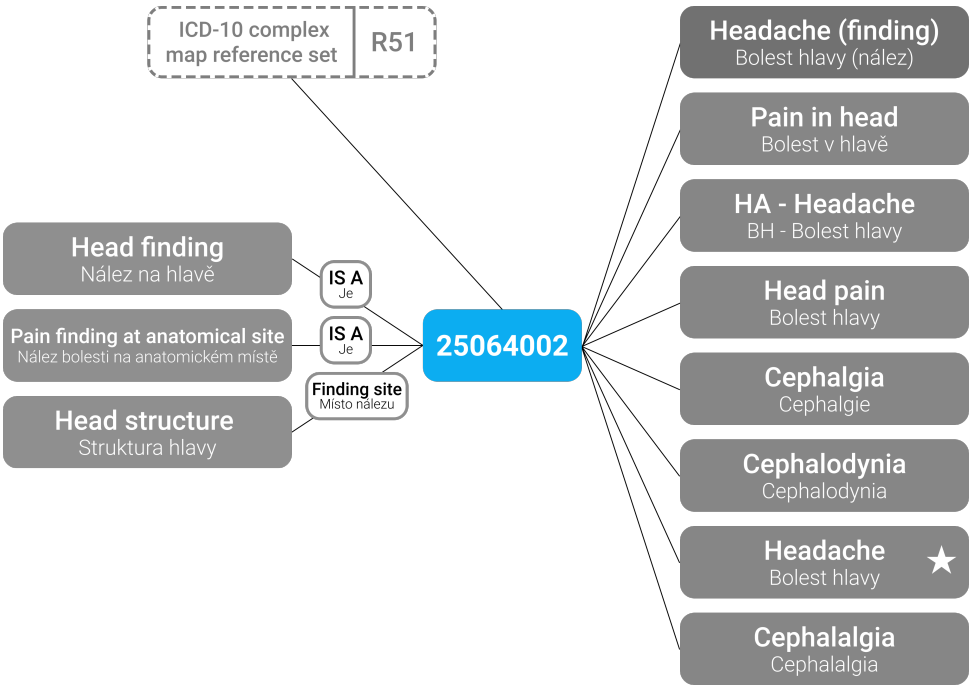
Záznamy, které již nejsou aktuální, jsou označeny jako neaktivní. Udržujeme je z důvodu záchování přehledu o historii změn. Nové komponenty nastavujeme vždy na hodnotu 1.

moduleId atribut specifikuje modul, ve kterém je daná komponenta spravována v daném efektivním čase. Hodnotou je SCTID konceptu modulu. Více informací o modulech se nachází v kapitole 2.6.

2.3.1 Koncept

Koncept (Concept) reprezentuje jeden lékařský pojem. Samotný koncept je pouze jakousi unikátní referencí v systému, čitelnost pro lidi mu dodávají až další komponenty. Jeho SCTID označujeme někdy jako `Concept code` případně `conceptId` a používáme jej jako referenci k danému pojmu. [27]

V případě námi zvoleného konceptu bolesti hlavy, se konceptem rozumí SCTID `25064002`. Povšimněte si, že vzhledem k tomu, že autorem konceptu je společnost IHTSDO, tak je v SCTID vynechán jmenný prostor. Z tohoto důvodu je i o něco kratší, než ukázkové SCTID na obrázku 2.4.



Obrázek 2.5: `Headache (finding)` - Koncept

Atributy

Koncept disponuje následujícími definujícími parametry. [29]

definitionStatusId rozlišuje, zda-li je koncept primitivní nebo plně definovaný.

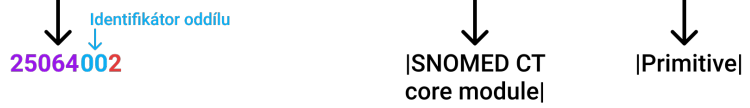
Koncept je plně definovaný, když ho jeho charakteristiky jednoznačně odlišují od podobných konceptů. [27] Příkladem je například koncept `Acute disease` (Akutní onemocnění), který je plně definovaný jeho dvěma vztahy. Prvním je vztah `IS-A disease` (JE onemocnění) a druhým `CLINICAL COURSE sudden onset AND/OR short duration` (KLINICKÝ PRŮBĚH náhlý nástup a / nebo krátké trvání). Řekneme-li, že je tento koncept plně definovaný, znamená to, že každý koncept s těmito dvěma vztahy je podtyp koceptu `Acute disease` nebo on sám.

Primitivní koncepty jsou takové, které podle charakteristik nelze odlišit od podobných konceptů. Příkladem jsou primitivní koncepty `Disease` (Onemocnění) a `Drug action` (Působení léku), které sdílí stejný vztah `IS-A clinical finding` (JE klinický nález).

Náš ukázkový koncept `25064002 |Headache (finding)|` je rovněž primitivním konceptem. Příbuzný koncept `735938006 |Acute headache (finding)|` je však již plně definovaný, protože poskytuje plnou charakteristiku.

Příklad

conceptId	effectiveTime	active	moduleId	definitionStatus
25064002	20100131	1	900000000000207008	900000000000074008



Obrázek 2.6: Příklad atributů konceptu `Headache (finding)`

2.3.2 Popis

Popis (Description) je sada textových reprezentací konceptu. SNOMED CT v současné době rozlišuje tři typy popisů. [27]

Úplný název (Fully specified name - FSN). Reprezentuje unikátní a jednoznačný význam konceptu. Většinou není pro svou složitost vhodný k zobrazení, ale je použit k jednoznačnému rozlišení pojmů.

Toto je zejména užitečné v případě podobně, či stejně znějících termínů. Příkladem může být koncept `35566002 |Hematoma (morphologic abnormality)|` a `385494008 |Hematoma (disorder)|`. Oba používají stejné synonymum `Hematoma`, ale jedná se o dva odlišné koncepty.

Každý koncept může mít pouze jeden úplný název v každém jazyce či dialektu. [27]

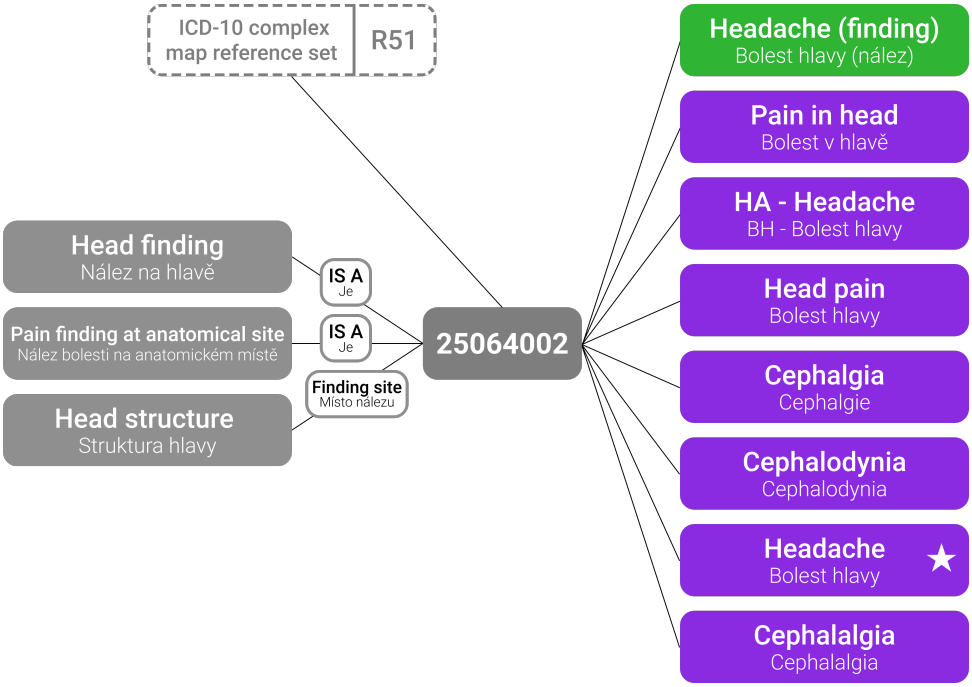
Sémantická značka (Semantic Tag) je část úplných názvů umístěná na konci v závorkách. Označují doménu, ke které koncept patří. Příkladem může být struktura těla (body structure), porucha (disorder) nebo vzorek (specimen). [33]

Synonymum (Synonym). Je jedna z možností, jak označovat koncept. Ten může mít více synonym. Synonyma nemusí být unikátní napříč terminologií.

Každý koncept má jedno ze synonym označené, jako preferovaný název (Preferred term - PT) v daném jazyce, či dialektu. Tento konkrétní název je potom obvykle použit pro zobrazení. Dále 0 až n synonym mohou být označeny jako akceptovatelné (acceptable) v daném jazyce či dialektu. [27]

Definice (Definition). Popisné, textové vysvětlení významu pojmu, které může překročit maximální povolenou délku pro úplný název. [34]

V našem modelovém konceptu na obrázku 2.7 se popisy nacházají v pravé části. Zeleně je vyznačen úplný název konceptu `25064002 |Headache (finding)|`. Sémantickou značkou je tedy `finding` neboli nález. Fialově jsou pak znázorněna všechna synonyma. Hvězdičkou označené synonymum je potom preferovaný název v britské i americké angličtině.



Obrázek 2.7: Headache (finding) - Popisy

Atributy

Popis disponuje následujícími definujícími parametry. [29]

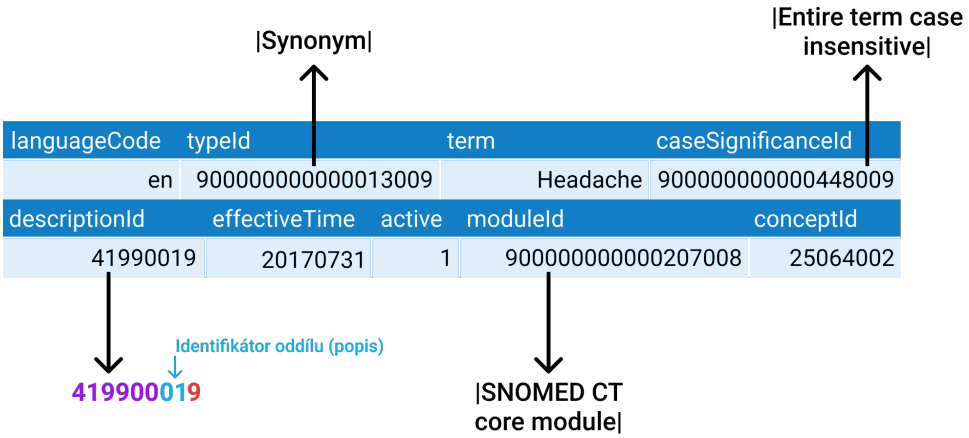
conceptId slouží k propojení popisu s konceptem. Jeho datovým typem je tedy SCTID. Id konceptu se musí nacházet ve stejném nebo závislém modulu jako popis.

languageCode specifikuje jazyk popisu dvou-písmennou zkratkou dle standardu ISO 639-1 (např. CS, EN). Tato zkratka reprezentuje pouze jazyk, nikoli použitý dialekt. Dialekt a přijatelnost popisu v daném dialektu je specifikován až pomocí jazykové referenční sady.

typeId upřesňuje typ konkrétního konceptu. Jeho datovým typem je SCTID konceptu, reprezentující atribut typu popisu.

Příklad

Na obrázku 2.8 si můžete prohlédnout příklad atributů popisu `Headache` konceptu `Headache (finding)`.



Obrázek 2.8: Příklad atributů popisu

2.3.3 Vztah

Vztah (Relationship) je definovaný vztah mezi dvěma koncepty. Každý koncept (vyjma kořenového konceptu) má minimálně jeden vztah IS-A (JE), který slouží k zařazení do hierarchie. [27]

Levá část obrázku 2.9 znázorňuje vztahy konceptu `25064002 |Headache (finding)|`. Koncept disponuje celkem třemi vztahy.

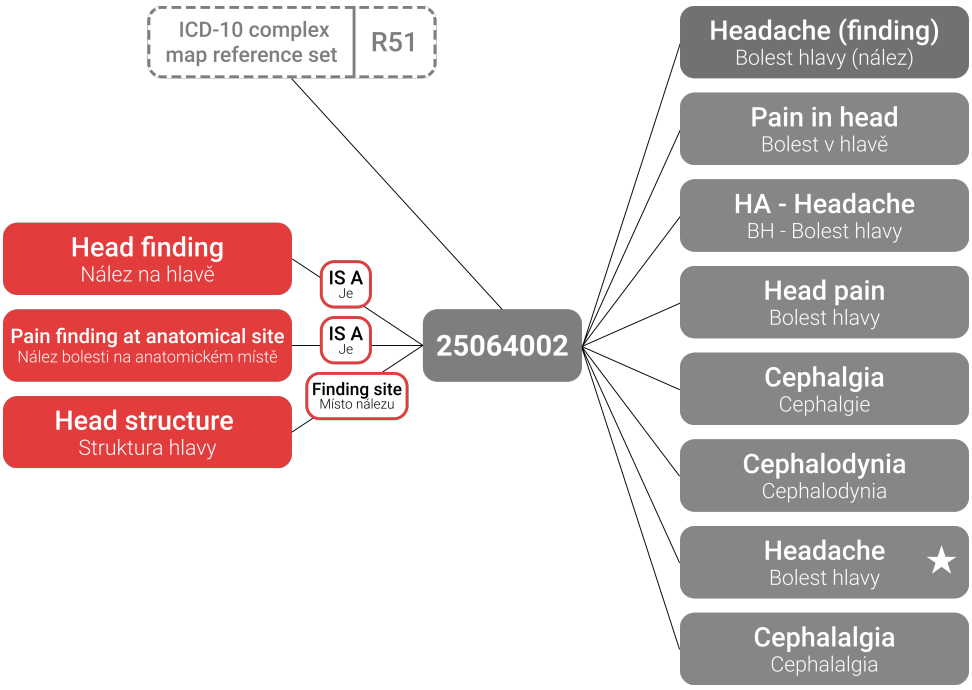
Dva z nich jsou typu `IS A` a to konkrétně `|IS A|Head finding|`, což by se dalo volně přeložit jako "je nález na hlavě" a `|IS A|Pain finding at anatomical site|`, což by v překladu znamenalo `"je nález bolesti na anatomickém místě"`.

Poslední vztah je typu `Finding site` ve spojení s konceptem `|Head structure|`. Volně přeloženo se jedná o vztah "místo nálezu - struktura hlavy".

Atributy

Vztah disponuje následujícími definujími parametry. [29]

sourceId je SCTID identifikující zdrojový koncept vztahu. Např. u vztahu typu IS-A ("Je") je zdrojem podřízený koncept.



Obrázek 2.9: Headache (finding) - Vztahy

destinationId je SCTID identifikující cílový koncept vztahu. Např. u vztahu typu IS-A ("Je") je cílem nadřazený koncept.

typeId je SCTID odkazující na koncept specifikující typ vztahu mezi dvěma koncepty. Tyto specifikující koncepty jsou podtypy konceptu |Concept model attribute|.

Typem vztahu je obvykle některý z konceptů SNOMED CT International edition, ale nic nebrání v tom definovat i nové.

characteristicTypeId specifikuje, zda-li byl vztah přímo zadán autorem, nebo zda-li byl odvozen popisovým logickým klasifikátorem.

modifierId je SCTID, které určuje typ logiky popisu. Rozlišujeme dva typy: existenční omezení (tj. „existuje nějaké“) a univerzální omezení (tj. „existuje pouze“).

Všechny vztahy v SNOMED CT International edition používají hodnotu `existenční omezení`. Tuto hodnotu je také doporučeno využít pro vztahy při vývoji rozšíření, opačná hodnota může výrazně negativně ovlivnit rychlost klasifikace. [29]

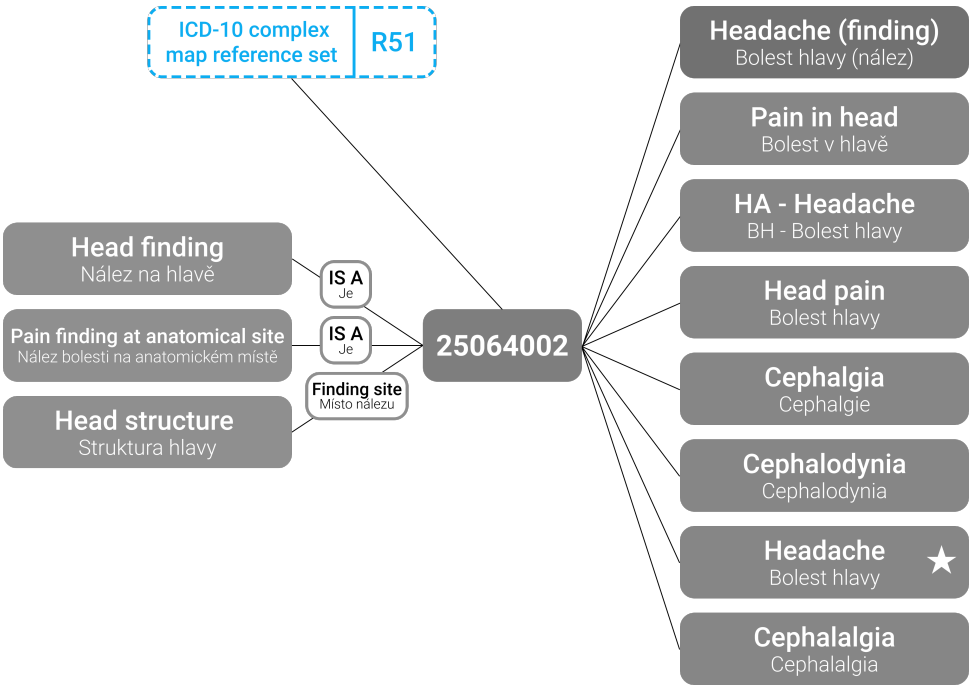
2.4 Referenční sady

Referenční sady (Reference sets - Refsets) jsou způsob, jak přiřadit sadě komponent další, nedefinující význam.

Referenční sada může být využita k vytvoření podskupiny komponent, které sdružuje nějaký kontext. Například se může jednat o termíny týkající se konkrétní lékařské specializace.

Dalším možností je přidružit komponentám pomocí referenčních sad další informace, které můžou sloužit například pro mapování pojmů terminologie SNOMED CT na nějaký terminologický systém (např. ICD-10 complex map reference set - mapa na terminologii ICD-10). [27, 32]

Na obrázku 2.10 si můžeme příklad využití referenční sady u modelového konceptu `25064002 |Headache (finding)|`. Konkrétně se jedná o hodnotu z referenční sady `ICD-10 complex map reference set`, která mapuje koncept bolesti hlavy ze SNOMED CT s identifikátorem `25064002` na koncept z terminologie ICD-10 s identifikátorem `R51`.



Obrázek 2.10: Headache (finding) - Referenční sady

2.5 Jmenné prostory

Všechny komponenty v rámci SNOMED CT disponují identifikátorem SCTID. Tento identifikátor musí být globálně unikátní napříč rozšířeními vyvíjenými po celém světe. K zajištění platnosti tohoto pravidla slouží jedna z částí identifikátoru SCTID - identifikátor jmenného prostoru. Jmenný prostor je vynechán v SCTID u komponent, jejichž autorem je společnost IHTSDO.

Před tvorbou rozšíření musí každý subjekt nejprve požádat IHTSDO o přidělení tohoto identifikátoru, který následně používá při tvorbě komponent. [29]

Ukázku jmenného prostoru v SCTID naleznete výše na obrázku 2.4.

2.6 Moduly

Moduly v rámci SNOMED CT slouží k organizování obsahu pro snadnější údržbu a publikaci. Každý koncept, popis, vztah a referenční sada musí patřit do nějakého modulu. Všechny komponenty v rámci modulu jsou publikovány najednou.

Modul je v rámci logického modelu koncept, který je pod-konceptem konceptu `900000000000443000 |Module (core metadata concept)|`. K přiřazení komponenty k modulu slouží atribut `moduleId`. [29]

2.6.1 Moduly SNOMED CT International edition

International edition se skládá ze dvou základních modulů spravovaných společností IHTSDO.

První z nich je |SNOMED CT core module|, který obsahuje základní klinické pojmy. Druhým je |SNOMED CT model component module|, ve kteřem se nachází komponenty metadat.

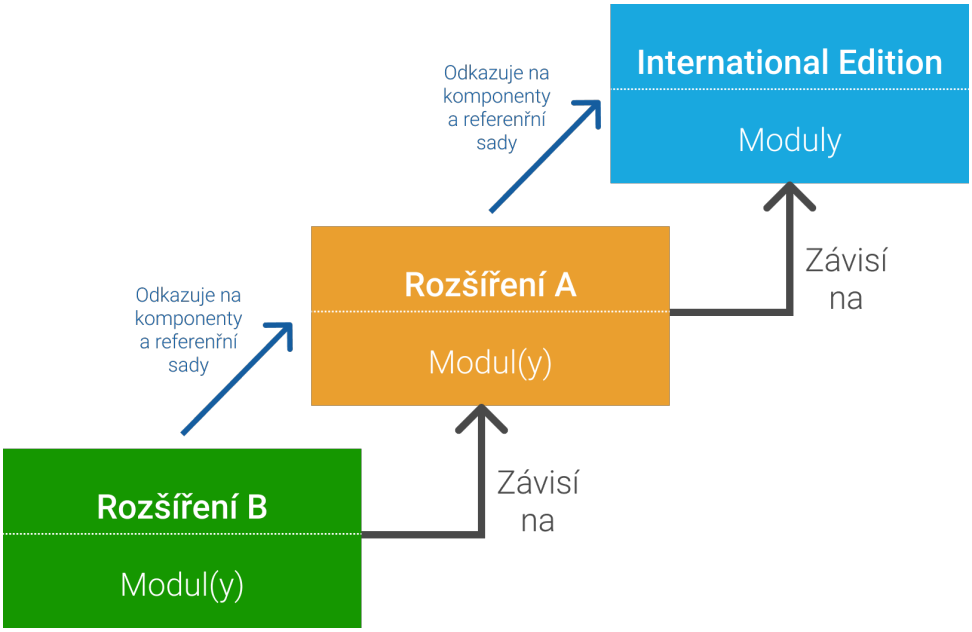
Společnost IHTSDO nadále spravuje několik dalších modulů. Příkladem může být modul |SNOMED CT to ICD-10 rule-based mapping module| umožňující mapování terminologie ICD-10 na terminologii SNOMED CT.

Mnoho dalších modulů je vyvíjeno společnostmi třetích stran. [30]

2.6.2 Závislost modulů

Obsahy jednotlivých modulů na sobě mohou být závislé. To znamená, že například námi vyvíjený modul může obsahovat koncept, jež je vazbou IS-A ("JE") spojen s nějakým konceptem z International edition. V takovém případě hovoříme o tom, že námi vyvíjený modul je závislý na SNOMED CT International edition.

Na obrázku 2.11 si můžete prohlédnout příklad závislosti modulů třech rozšíření.



Obrázek 2.11: Příklad závislosti modulů

2.7 Rozšíření

Rozšíření je sada komponent a referenčních sad, kterými lze rozšířit SNOMED CT International edition. Tato rozšíření jsou vytvářena, strukturována a distribuována podle pravidel stanovených společností IHTSDO.

Obsah těchto rozšíření není samostatnou terminologií, jedná se pouze o rozšíření existujícího obsahu. Rozšíření jsou proto vždy závislá minimálně na SNOMED CT International edition.

Z pohledu logického návrhu se rozšíření skládá z minimálně jednoho modulu. Veškerý komponenty a referenční sady rozšíření musí náležet těmto modulům. Pakliže se rozhodneme pro členění do více modulů, tyto moduly musí být hierarchicky seskupeny modulem reprezentujícím správce tohoto rozšíření.

V zásadě lze rozlišovat dva typy rozšíření.

- Jazykové rozšíření - Překládá terminologii do jiného jazyka (např. SNOMED CT Spanish Edition - španělský překlad)
- Pojmové rozšíření - Přidává pojmy týkající se jiné než klinické medicíny (např. SNOMED CT Veterinary extension - veterinářké pojmy)

2.8 Edice

Edice je kompletní sada komponent a referenčních sad, které jsou relevantní vzhledem k jejímu zaměření.

Každá edice se skládá z obsahu SNOMED CT International edition a nula nebo více rozšíření, potažmo jejich modulů. Díky tomuto může být edice použita jako samostatná teminologie.

Mnoho autorů rozšíření volí cestu vydání jejich obsahu jako edici, protože spolu s tím dodávají i obsah, na kterém je jejich rozšíření závislé. [29]

2.9 Implementace SNOMED CT

Pro úspěšnou implementaci SNOMED CT v rámci organizace je potřeba zajistit zejména splnění následujích podmínek:

- Dostupnost jazykové mutace
- Dostatečné pokrytí problematiky

V případě nesplnění některé z výše uvedených podmínek je jediným řešením vývoj rozšíření, případně vlastní edice SNOMED CT. [29]

2.9.1 Implementace v rámci České republiky

V případě České republiky je hlavní překážkou pro implementaci absence české edice SNOMED CT. Obecně vzato, IHTSDO spolupracuje s národními organizacemi na tvorbě jazykových mutací. Například Španělsko, či Švédsko disponuje vlastní edicí SNOMED CT, která obsahuje všechny pojmy International edition přeložené do příslušného jazyka, obohacené o národně specifické pojmy.

Ústav zdravotnických informací a statistiky ČR (ÚZIS) si je vědom nesystematického státního přístupu k využití terminologických a klasifikačních systémů pro oblast zdravotnictví. Z tohoto důvodu v lednu roku 2020 spustil projekt na vytvoření Národního centra pro medicínské nomenklatury a klasifikace (NCMNK). [44] To institucionálně spadá pod ÚZIS.

Jedním z cílů centra je právě analýza použití a potřebných kroků k implementaci SNOMED CT v České republice.

Centrum již zahájilo první kroky nutné k implementaci a v červenci 2020 zřídilo Národní release centrum SNOMED CT (NRC). [45] Základním úkolem NRC je registrovat uživatele licencí opravňujících používání SNOMED CT v České republice (vzhledem k tomu, že ČR je členem neziskové organizace SNOMED International, jsou tyto licence na území ČR poskytovány zdarma). Do konce roku 2021 plánuje NRC přeložit základní sadu konceptů a na pilotních projektech ověřit, zda a v jaké míře je tato terminologie využitelná v českém zdravotnickém prostředí.

Kapitola 3

SNOMED CT API

Vyvíjená webová aplikace je závislá na aplikačním rozhraní poskytované terminologickým serverem. V následující kapitole popíšu principy funkcionalit a strukturu REST API terminologického serveru Snowstorm (dále jen "API"). O terminologickém serveru Snowstorm se více dočtete v kapitole 5.2.1 Terminologický server.

3.1 Popis

API poskytuje kompletní sadu operací pro práci s terminologií. Terminologický server umožňuje vyhledávat, upravovat, mazat a vytvářet nové koncepty. Všechny tyto aktivity provozuje v rámci jím spravovaných vývojových větví. Zároveň posyktuje cesty k importování/exportování terminologie na/z serveru.

Poskytovány jsou i nástroje, které nejsou v rámci aplikace implementovány, jako práce s referenčními sadami a validační utilita.

3.2 Výčet použitých funcktionalit

V následujícím seznamu blíže specifikuji, které z nabízených funkcionalit API jsou využity pro chod aplikace.

- Koncepty
  - Tvorba nového konceptu
  - Načtení konceptu
  - Načtení více konceptů

- Aktualizace konceptu
- Smazání konceptu
- Načtení potomků konceptu

- Vývojové větve
  - Načtení všech větví
  - Načtení detailu větve
  - Načtení potomků větve
  - Tvorba nové větve
  - Posouzení větve
  - Kontrola integrity větve
  - Posouzení sloučení větví
  - Sloučení větví
- Statistika změn
  - Celková statistika
  - Změněné úplné názvy
  - Deaktivované koncepty
  - Deaktivovaná synonyma
  - Nové koncepty
  - Nová synonyma
  - Reaktivované koncepty
  - Reaktivovaná synonyma
- Popisy
  - Vyhledávání popisů

3.3 Principy funkcionalit

V odstavcích níže popíšu princip poskytovaných funkcionalit, které nejsou zřejmé z jejich názvu.

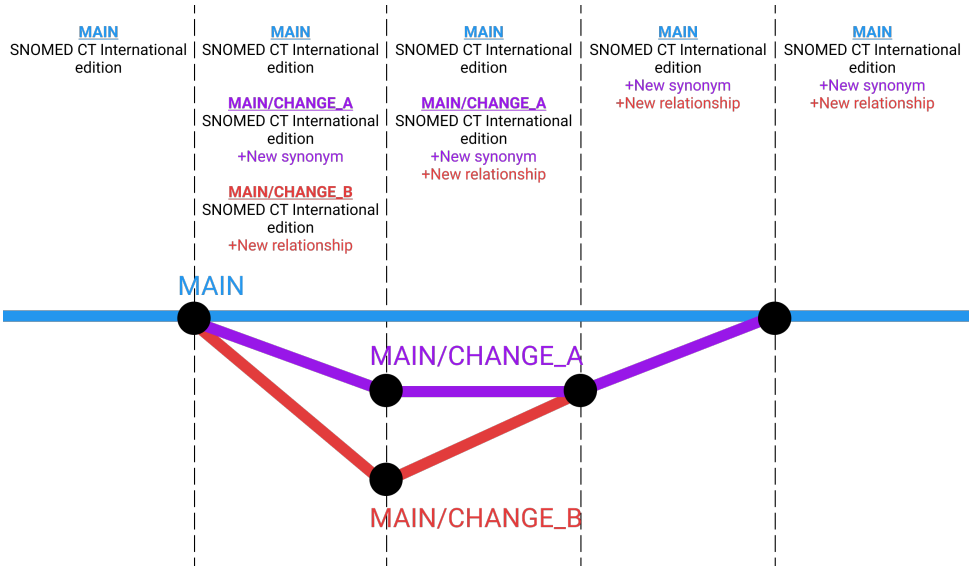
3.3.1 Vývojové větve

Vývojové větve slouží k organizaci změn a snadnější kolaboraci více uživatelů. Většina funkcionalit, které pracují s koncepty, vyžadují specifikovat vývojovou větev jako parametr.

Každá úprava (např. změna popisu, přidání vztahu ke konceptu) se pak efektem omezuje na svou větev. Obdobně pak při vyhledávání jsou změny reflektovány pouze v rámci konkrétní větve.

Výraz ”větve” používáme z důvodu, že je možné je dělit a poté znovu slučovat. Každá oddělená větev obsahuje všechny změny ze své rodičovské větve, slučováním dvou větví slučujeme i jejich změny.

Kořenová větev se nazývá `MAIN` a obsahuje naimportovanou terminologii.



Obrázek 3.1: Princip vývojových větví

3.3.2 Vyhledávání

Vyhledávání probíhá napříč všemi komponentami typu popis. Terminologický server poskytuje vyhledávání dvojího druhu.

Fulltextové. Tradičním způsobem je fulltextové vyhledávání, kdy jsou popisy porovnávány s hledaným textovým řetězcem. Navráceny jsou všechny částečně odpovídající řetězce. Scénář s využitím fulltextového vyhledávače najdete v podkapitole 7.2.2.

ECL. Expression Constraint Language (ECL) je formální jazyk pro definování množin omezení klinických výrazů představovaných buď prekoordinovanými nebo postkoordinovanými výrazy. [26] Příklad použití ECL vyhledávače najdete v podkapitole 7.2.3.

3.4 Příklad volání

Následující ukázka demonstruje příklad jednoho API volání. Jako příklad jsem zvolil načtení úplného názvu našeho ukázkového konceptu `25064002 |Headache (finding)|` pomocí API. Toto konkrétní volání jsem zvolil z důvodů malého rozměru navráceného objektu, ačkoli jej vyvíjená aplikace přímo nevyužívá.

Jedná se o standardní HTTP volání metodou `GET` na koncový bod:

`https://snowstorm.stepanek.app/MAIN/descriptions/755191011`

Odpověd v podobě `JSON` objektu, který reprezentuje komponentu typu popis naleznete níže (Ukázce kódu 1).

```
{
    "active": true,
    "moduleId": "900000000000207008",
    "released": true,
    "releasedEffectiveTime": 20170731,
    "descriptionId": "755191011",
    "term": "Headache (finding)",
    "conceptId": "25064002",
    "typeId": "900000000000003001",
    "acceptabilityMap": {
        "900000000000509007": "PREFERRED",
        "900000000000508004": "PREFERRED"
    },
    "type": "FSN",
    "lang": "en",
    "caseSignificance": "CASE_INSENSITIVE",
    "effectiveTime": "20170731"
}
```

Ukázka kódu 1: Komponenta typu popis - `Headache (finding)`

Kapitola 4

Návrh systému

4.1 Rozsah použití

Systém je navržen tak, aby umožňoval snadné procházení, průzkum a vývojové aktivity v rámci taxonomie SNOMED CT. Systém je tak vhodný nejen pro lidi, kteří se seznamují s technologií SNOMED CT, ale zároveň i pro terminology, kteří pracují například na tvorbě nové jazykové mutace. Díky jednoduchému a přehlednému uživatelskému rozhraní je vhodný pro uživatele všech úrovní.

4.2 Systémové požadavky

Tato sekce obsahuje přehled požadavků, které musí systém splňovat ze softwarového pohledu. Tyto požadavky vznikaly jak definicí zadavatele, tak postupným objevováním při vývoji.

4.2.1 Funkční požadavky

FR1 - Vyhledávač. Systém bude poskytovat vyhledávač konceptů typu popis dle:

- Textového řetězce
- ECL dotazu

Pro vyhledávání pomocí textového řetězce bude poskytovat filtrování dle:

- Sémantické značky
- Stavu - aktivní/neaktivní
- Jazyka
- Referenční sady

FR2 - Hierarchický prohlížeč. Systém bude poskytovat hirerarchický prohlížeč konceptů. Tato hierarchie bude začínat kořenovým konceptem. Uživatel bude mít možnost expandovat koncept a zobrazit tím jeho "děti", tedy pojmy speciálnější ve významu. Uživatel bude mít rovněž možnost zobrazit snadno detail vybraného konceptu.

FR3 - Prohlížeč konceptu. Systém bude umožňovat zobrazit si koncept včetně všech jeho vlastností, kterým jsou například synonyma, definice, vztahy, atp.

FR3 - Editace konceptů. Systém bude umožňovat úpravu existujících a tvorbu nových komponent. Změny budou reflektovány do uživatelem zvolené vývojové větve.

FR4 - Správa vývojových větví. Systém bude umožňovat volbu aktuální pracovní větve, tvorbu nových větví a jejich slučování.

FR5 - Přehled změn. Systém bude umožňovat zobrazit seznam změn provedených na aktuální větvi.

4.2.2 Nefunkční požadavky

NFR1 - Platformní nezávislost. Systém bude implementován tak, aby nebyl závislý na použitém operačním systému ani určité verzi prohlížeče.

NFR2 - Podpora desktopového rozlišení. Systém bude implementován tak, aby byl použitelný na všech typech desktopových rozlišení.

NFR3 - Škálování. Systém bude správně škálovat zobrazovaná data. SNOMED-CT obsahuje přes 350 000 pojmů, které jsou vzájemně provázány, je proto nutné načítat jen jejich nezbytně nutnou část.

NFR4 - Stav v URL. Aplikace bude udržovat stav pomocí URL. Při návštěvě takovéto adresy bude aplikace obnovena do stejného stavu.

4.3 Případy užití

Pro účely bakalářské práce jsem zvolil zachycení požadavků pomocí případů užití, které se nachází v příloze A a sledují následující ukazatele:

1. Identifikace - číselný identifikátor a název
2. Popis - krátký popis, co konkrétní případ užití reprezentuje
3. Aktéři - kdo daný případ užití vykonává
4. Předpoklady - podmínky, které musí být splněny pro zahájení případu užití
5. Hlavní scénář průchodu - přehled kroků v rámci případu užití
6. Alternativní scénář - akce reagující na nestandardní chování

Kapitola 5

Technická analýza

Následující kapitola se věnuje technickému aspektu vyvíjené aplikace. Popisuje návrh architektury a provádí rešerši zvolených technologií.

5.1 Návrh architektury

Práce je koncipována jako webová Single-page application (SPA) aplikace sloužící jako tenký klient k aplikačnímu REST rozhraní (REST API) poskytovaného terminologickým serverem. Implementována je tak, že respektuje všechny požadavky z kapitoly 4 Návrh systému. Formálně tuto aplikaci označuji termínem Snow UI.

SNOMED CT obsahuje enormě velké množství konceptů, potažmo popisových komponent. Z tohoto důvodu je každé hledání v rámci terminologie náročnou operací. Terminologické servery proto toto hledání akcelerují pomocí technologie Elasticsearch. Ta využívá chytrého indexování, a proto je vyhledávání proveditelné v relevantním čase i na průměrném hardwaru. [5]

Na obrázku 5.1 naleznete schématické znázornění architektury systému a komunikace mezi jednotlivými částmi.

Obrázek 5.1: Návrh architektury systému

5.1.1 Architektonické principy

V následující podkapitole nastíním použité architektonické principy použité v rámci webové aplikace.

Stránkování

Vzhledem k rozsáhlosti terminologie vzniká potřeba stránkování obsahu, například při vyhledávání. Díky tomuto se šetří bandwidth, udržujeme přehlednost a zbytečně nepřetěžujeme prohlížeč načítáním velkého množství obsahu.

State management

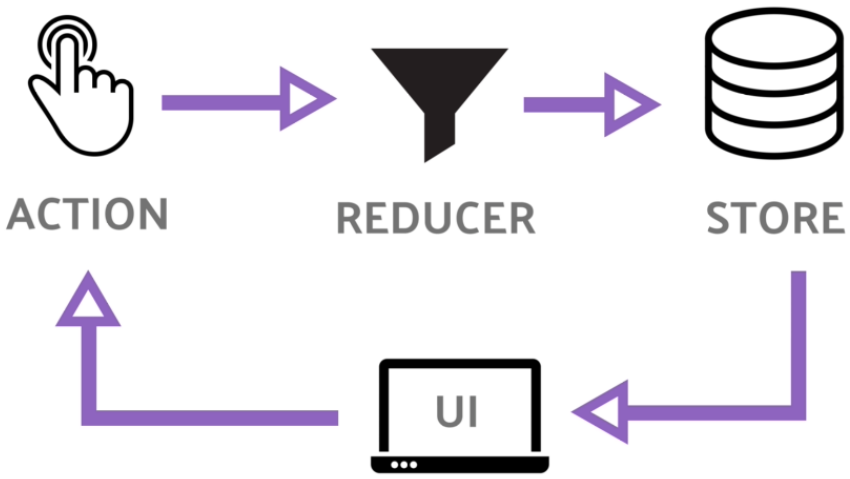
Nezávislost jednotlivých komponent přináší výhodu v podobě velmi dobrého oddělení zodpovědnosti, ale zároveň přináší problémy v případě nutnosti práce více komponent se stejnými daty. Řešením tohoto problému se zabývá state management.

V rámci bakalářské práce jsem implementoval dva přístupy.

URL stav. Nefunkční požadavek NFR4 - Stav v URL vyžaduje obnovení stavu aplikace z URL. Tohoto principu je využito tam, kde to dává smysl, tedy např. u vyhledávače. V aplikaci je využita kombinace URL query parametrů, které jsou obnoveny do stavu React komponent. V případě neexistence takového parametru je načtena výchozí hodnota. Toto garantuje, že v případě, kdy například jedna osoba pošle URL filtrovaného vyhledávání kolegovi, tak mu jsou zobrazeny totožné výsledky.

Redux pattern. V ostatních případech, které vyžadují state management (např. editor konceptu), je využit redux pattern. Ten funguje na principu držení stavu v úložisti (Store). Stav je následně přesně modifikován pomocí akcí (Action) vyvolaných prostřednictvím odesílatele (Dispatcher). Akce je odbavena na straně reduceru, kde jsou provedeny příslušné změny stavu.

V praxi potom v kořenové komponentě inicializujeme stav a odesílatele. Obě tyto proměné následně posíláme jako parametry do podřízených komponent. V případě potřeby modifikace stavu z nějaké z podřízených komponent vypravíme příslušnou akci pomocí odesílatele. Následně dojde automaticky k odbavení akce a úpravě stavu. Posledním krokem je automatické pře-renderování komponent, aby odpovídaly novému stavu.



Obrázek 5.2: Redux pattern [7]

Pre-rendering a cachování

Aplikace v maximálním možném měřítku využívá server-side pre-rendering. To znamená, že je stránka předpřipravena na straně serveru (v případě, že potřebuje data z API, provede tento dotaz právě server) a klientovi odeslána jako celek. [37] Toto má celkově pozitivní vliv na výkon aplikace.

V případě, že toto nelze implementovat (např. hierarchický prohlížeč) a API dotazy jsou prováděny z klienta, je tak učiněno pomocí knihovny SWR. [41] Ta nejenže značně zjednodušuje implementaci, ale zároveň veškeré dotazy automaticky cachuje. To přináší zvýšení výkonu při opakování stejných dotazů.

5.2 Volba technologií

Následující text se zabývá rozborem rozhodnutí v oblasti použitých technologiích (frameworků a knihoven), které jsou využity v rámci vyvíjené aplikace.

5.2.1 Terminologický server

Terminologický server slouží k nahrání a provozu SNOMED CT a poskytuje REST API, které jsem detailně popsal v kapitole 3 SNOMED CT API. Rozhodoval jsem se mezi dvěma konkurenčními řešeními.

Snowstorm. Terminologický server, jehož vývoj je zastřešen společností IHTSDO. Na pozadí jej využívá i nástroj SNOMED International Autho-

ring Platform. Je zaměřen na provoz a autorskou činnost pouze v rámci terminologie SNOMED CT. [13]

Snow Owl. Univerzální terminologický server s podporou SNOMED CT, vyvíjen maďarskou společností B2i Healthcare. Ta jej využívá pro provoz svého komerčního prohlížeče a editoru Snow Owl MQ. [1] Snow Owl je navržen flexibilně tak, aby na něm bylo možné provozovat i jiné terminologie a byl snadno rozšiřitelný. [2]

Srovnání

Metrika	Snowstorm	Snow Owl
Vývojář	IHTSDO	B2i Healthcare
Kvalita dokumentace	Nedostatečná	Dostatečná
Podpora vývojových větví	Ano	Ano
Podpora ECL	Ano - v1.3	Ano - v1.4
Stránkování	nextPage, offset	nextPage

Tabulka 5.1: Porovnání terminologických serverů

Shrnutí

Otestoval jsem obě dvě řešení. Testy prokázaly plnou funkčnost po importu SNOMED CT International Edition i srovnatelný výkon. Jeden z klíčových rozdílů nastává v poskytováném API.

Pro efektivní procházení taxonomie tak, jak jsem navrhl uživatelské rozhraní, je zapotřebí stránkování z důvodu obvykle velkého množství dat. Oba terminologické serververy umožňují stránkovat za pomocí `searchAfter` hashe. Toto řešení je výkonově optimalizovanější, ale neumožňuje stránkovat opačným směrem, což tento přístup činí nepoužitelným vzhledem k požadavkům.

Snowstorm však umožňuje i stránkování za pomocí odsazení (potažmo čísla stránky).

Zároveň je z mého pohledu lepší použít nástroj poskytovaný stejnou organizací, která vyvijí SNOMED CT a provozuje na něm vlastní administrační nástroj.

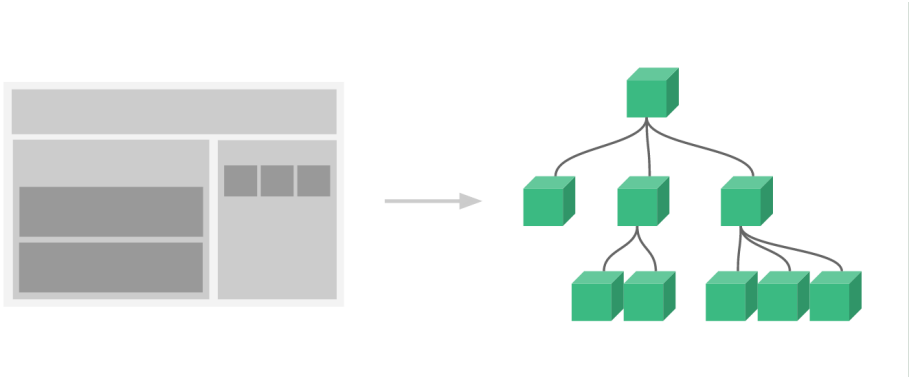
Z těchto důvodů padla volba na terminologický server Snowstorm.

5.2.2 React

Při tvorbě moderní webové aplikace je neodmyslitelným průmyslovým standardem využití javascriptové knihovny. Ta do značné míry abstrahuje práci s javacriptem jako takovým a zjednodušuje práci.

Každá z knihoven funguje na trochu jiném principu, ale v zásadě se protínají v seznamu nabízených zjednodušení a funkcionalit. Tento seznam lze definovat následovně.

Segmentace kódu na komponenty. Stránka je rozdělena na komponenty, které jsou povětšinou vzájemně nezávislé. Při jejich vývoji se klade velký důraz na jejich znovupoužitelnost. Výsledná stránka potom vznikne spojením a vrstvením těchto komponent.



Obrázek 5.3: Vizualizace dělení stránky na komponenty [42]

Správa DOM. Na rozdíl od klasických javascriptových vanilla aplikací za nás knihovna obstarává práci s DOM a tím i vykreslování dat. Programátor poté pracuje jednoduše s datovými proměnými a knihovna zajišťuje jejich vykreslení. V případě změny příslušných dat sama zabezpečí překreslení. Odpadá nutnost provádění ručních manipulací.

S velkým množstvím výhod rovněž přichází i relativní nevýhoda v podobě nutné znalosti správné práce s takovou knihovnou. Tuto nevýhodu částečně mitiguje subjektivně velmi kvalitní dokumentace všech srovnávaných řešení.

Po důkladné rešerši, kterou naleznete v příloze B, jsem dospěl k rozhodnutí použít knihovnu **React**.

5.2.3 Next.js

React sám o sobě neposkytuje nástroje nutné pro tvorbu Single page aplikace (SPA), protože se stále jedná pouze o knihovnu. Pro tento účel je nutné použít

nějaký webový framework.

Z těchto důvodů jsem se rozhodl využít Next.js. [40] Při výběru jsem se nerozhodoval mezi žádnými dalšími alternativami, protože Next.js je velmi rozšířené a dobře zdokumené řešení, bez přímého konkurenta.

Tento framework nabízí několik funkcionalit, které jsou potřebné pro tvorbu SPA aplikací. Jsou jimi:

Routing. Směrování je proces, kdy naše aplikace reaguje překreslením na změnu URL adresy. V kontextu klasických aplikací hovoříme o přesměrování na jinou stránku. V našem případě však dojde pouze k překreslení DOM stránky na příslušný obsah.

Server-side rendering (SSR). Vyvíjená aplikace získáva data z terminologického serveru. Tradičně je toto volání provozováno z klientského prohlížeče na poskytovatele. Avšak v případě SSR toto volání provede webový server, z navrácených dat sestaví stránku a až tu pošle klientovi. Toto vede ke zvýšení výkonu aplikace a hladším přechodům.

Internacionalizace. Next.js poskytuje nástroje pro snadnou internacionalizaci obsahu. Dá se předpokládat, že vyvíjená aplikace bude potřebovat více jazykových mutací.

5.2.4 TypeScript

Aplikace používá technologii TypeScript, více informací naleznete v příloze C.

5.2.5 Tailwind

Při tvorbě moderních webových stránek se čím dál tím více opouští standardní přístup psaní kaskádových stylů ve prospěch tzv. CSS frameworků. V zásadě se jedná o sadu předpřipravených CSS tříd s jasně definovaným použitím.

Každý z frameworků poskytuje vlastní přístupy k řešení nejčastějších problémů. V zásadě by se přínosy daly shrnout následovně:

Vizualní konzistence. Knihovny si zakládají na znovupoužitelnosti jednotlivých částí, což přispívá k udržení vizuální konzistence napříč celou aplikací.

Responzivita. Většina knihoven praktikuje tzv. mobile-first přístup, kdy je mobilní verze vyvíjena jako první a postupně upravena pro vyšší rozlišení. Takto lze velice snadno dosáhnout plné responzivity webové aplikace.

Grid. Knihovny obvykle nabízí snadný způsob, jak responzivně udržovat grid aplikace, dle specifikace vývojáře.

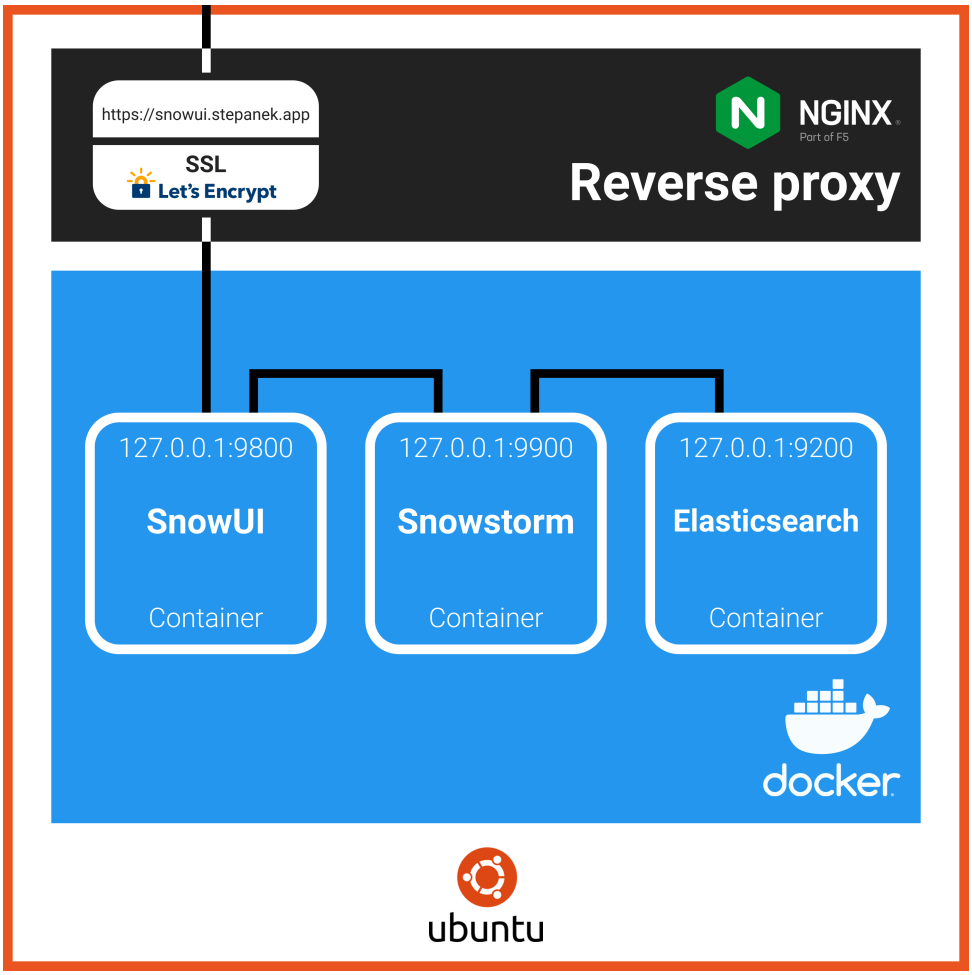
Po provedené rešerši, kterou nalezte v příloze D a pečlivém zvážení jsem se rozhodl využít Tailwind CSS jakožto použitý framwork.

5.3 Nasazení

Aplikace a terminologický jsou nasazeny na virtuálním serveru (VPS) pomocí virtualizačního nástroje Docker. Celkově se jedná o 3 nezávislé kontejnery:

- Vyvíjená webová aplikace (SnowUI)
- Terminologický server (Snowstorm)
- Vyhledávací engine Elasticsearch

O přístup z internetu a SSL zabezpečení se stará webový server Nginx [11], který je nastaven jako reverse proxy [6]. Certifikační autoritou je společnost Let's encrypt, která poskytuje certifikáty zdarma [10]. Testovací verze webové aplikace je dostupná na adrese `https://snowui.stepanek.app`.



Obrázek 5.4: Diagram nasazení

Kapitola 6

Testování

Následující kapitola popisuje metodiky testování využité v rámci bakalářské práce.

6.1 Rozsah testování

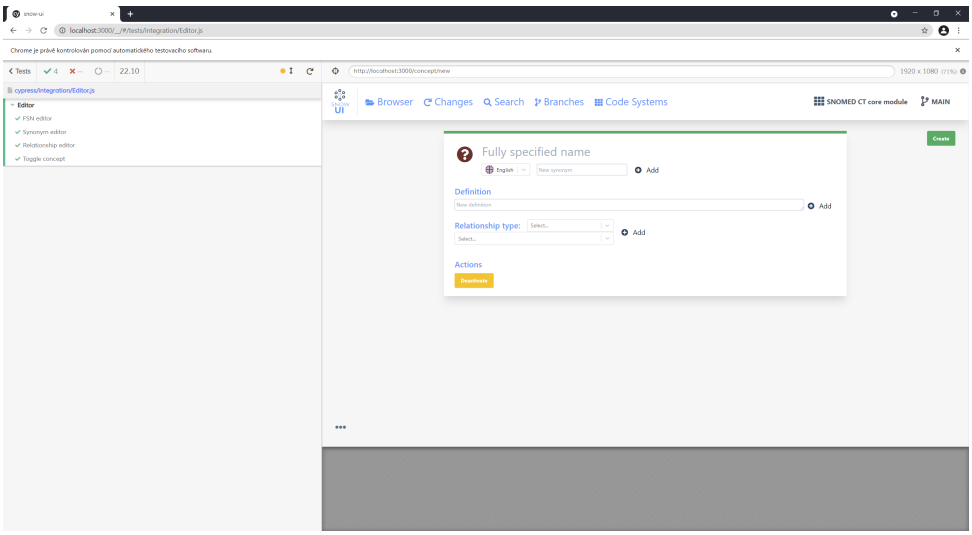
Vyvíjená aplikace je pouze tenký klient k poskytovanému API a neobsahuje příliš mnoho vlastní logiky. Toto značně omezuje rozsah testů, které jsem musel implementovat, protože terminologický server je testován jeho vývojářem.

Z tohoto důvodu jsem se rozhodl implementovat testy zaměřené na uživatelské rozhraní. Konkrétně jsem použil uživatelské a automatické integrační testy.

6.2 Integrační testování uživatelského rozhraní

Integrační testy z definice ověřují bezchybnou komunikaci a spolupráci mezi jednotlivými komponentami. [36] V našem případě testování vztahujeme na uživatelské rozhraní. To v praxi znamená, že automaticky prováděnými testy ověřujeme správnou funkcionalitu a spolupráci jednotlivých React komponent.

Pro tyto účely je využit testovací framework Cypress. [8] Tento framework poskytuje nástroje, které umožňují rychlou tvorbu spolehlivých testů uživatelského rozhraní. Sám přitom disponuje uživatelsky velmi přívětivým testovacím běhovým prostředím.



Obrázek 6.1: Cypress - Ukázka testovacího běhového prostředí

6.2.1 Otestované integrace

- Hierarchický prohlížeč
  - Inicializace
  - Expandování konceptů
  - Detail konceptu
- Detail konceptu
  - Inicializace
  - Přechod do editačního režimu
- Editor konceptu
  - Editor úplného názvu
  - Editor synonym
  - Editor vztahů
  - Změna aktivnosti konceptu
- Navigační menu
  - Cookies uživatelských nastavení
  - Výběr modulů
- Prohlížeč větví
  - Inicializace
  - Exapandování a změna větve
- Vyhledávač
- Přehled změn

6.2.2 Netestované integrace

- Průvodce sloučením větví

6.2.3 Příklad integračního testu

Na následujícím příkladu (Ukázka kódu 2) naleznete realizaci jednoho z integračních testů pomocí frameworku Cypress. Konkrétně se jedná o editor synonym v editoru konceptu.

Test přidá nové synonymum, přepne jeho stav (aktivní->neaktivní->aktivní), ověří, zda-li se správně zobrazí menu nastavení akceptability a následně synonymum odstraní.

6.3 Uživatelské testování

Druhým typem provedeného testování jsou uživatelské testy. Ty spočívají v průchodu aplikace reálným uživatelem dle předem definovaného scénáře a následném sběru zpětné vazby. Přinášejí výhodu v podobě nezaujatosti a jiného pohledu na problematiku.

Na uživatelském rozhraní proběhlo testování jednoho scénáře "Přidání českého synonyma", který naleznete v příloze E a jehož reálný průchod si můžete prohlédnout na obrázcích v kapitole 7.1.

6.3.1 Testeři

Tester 1

Profese: Vývojář
Zpětná vazba:

- Správce větví - Vývojová větev se po vytvoření automaticky nezvolí
- Editor - Akceptabilita není viditelná bez rozkliknutí nastavení
- Editor - Chybí tooltipy tlačítek
- Průvodce sloučením větví - Po sloučení neprovede automaticky přesměrování

Tester 2

Profese: Student informačních technologií
Zpětná vazba:

- Editor - Nepopsané tlačítko žárovky, co dělá?
- Editor - Discard nezahodí změny, je nutné znovu načíst stránku
- Editor - Při kliknutí na koncept vztahu dojde k přesměrování a zahození provedených změn

Tester 3

Profese: Student informačních technologií
Zpětná vazba:

- Editor - Nový vztah - Vstupy jsou nevzhledně nalepeny na sobě
- Editor - Tlačítko Deactivate má šedý rámeček
- Editor - Nelze změnit jazyk již vytvořeného synonyma

```
it("Synonym editor", () => {
    //Vytvoření synonyma
    cy
        .get('input[name=new-synonym]')//Nalezene textového pole
        .type("Test synonym")//Vyplní synonymum
    cy
        .get('#new-synonym button')//Nalezne tlačítko pro potvrzení
        .click();//Stiskne toto tlačítko a tím vytvoří nové synonymum
    cy
        .get('.synonyms input')//Nalezne všechna synonyma
        .should('have.value', 'Test synonym')//Obsahuje nové synonymum?

    //Deaktivace a opětovná aktivace synonyma
    cy
        .get('.synonyms button[name=toggle-synonym]')//Nalezne přepínač
        .as('toggle-button')//Vytvoří alias
        .click()//Přepne stav
    cy
        .get('@toggle-button')//Nalezne přepínač pomocí aliasu
        .children('i')
        .should('have.class', 'text-danger');//Je synonymum deaktivované?
    cy
        .get('@toggle-button')//Nalezne přepínač pomocí aliasu
        .click()//Přepne stav
    cy
        .get('@toggle-button')//Nalezne přepínač pomocí aliasu
        .children('i')
        .should('have.class', 'text-success');//Je synonymum aktivované?

    //Nastavení akceptability
    cy
        .get('.synonyms button[name=synonym-acceptability]')//Najde tlačíko
        .click()//Klinutím otevře nastavení
    cy
        .get('.synonyms')
        .contains('Acceptability');//Obsahuje správný nadpis?
    cy
        .get('.synonyms')
        .contains('GB English');//Obsahuje správně nastavení?

    //Odstranění synonyma
    cy
        .get('.synonyms button[name=delete-synonym]')//Nalezne tlačítko
        .click()//Kliknutím odstaní synonymum
    cy
        .get('.synonyms input')//Nalezne všechna synonyma
        .should('not.have.value', 'Test synonym')//Bylo odstraněno?

})
```

Ukázka kódu 2: Integrační test editoru synonym

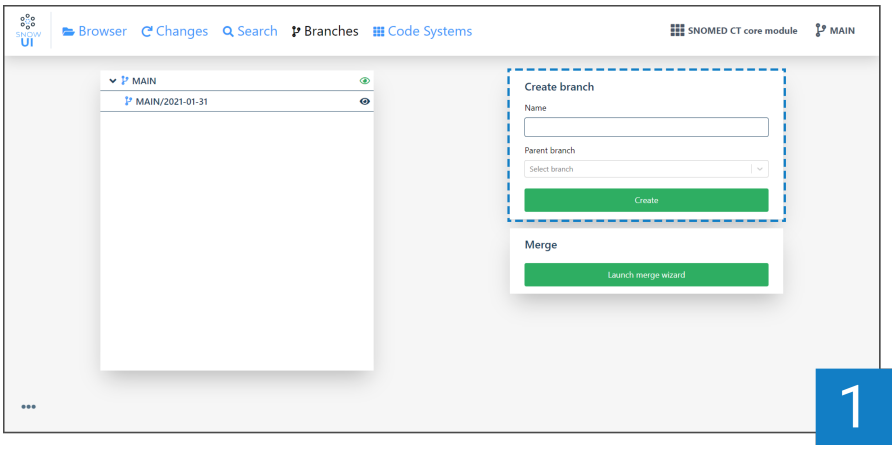
Kapitola 7

Uživatelské scénáře

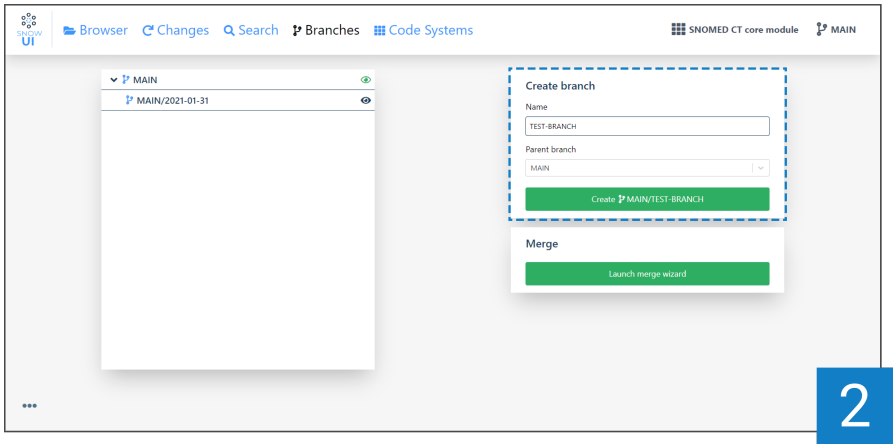
7.1 Přidání českého synonyma

Modelovým uživatelským scénářem je přidání českého synonyma existujícímu konceptu ve vývojové větvi a její následné sloučení do větve hlavní. Tato aktivita by byla nezbytná při překladu terminologii do českého jazyka. V následujícím scénáři popíšu tento proces na konceptu `25064002 |Headache (finding)|`.

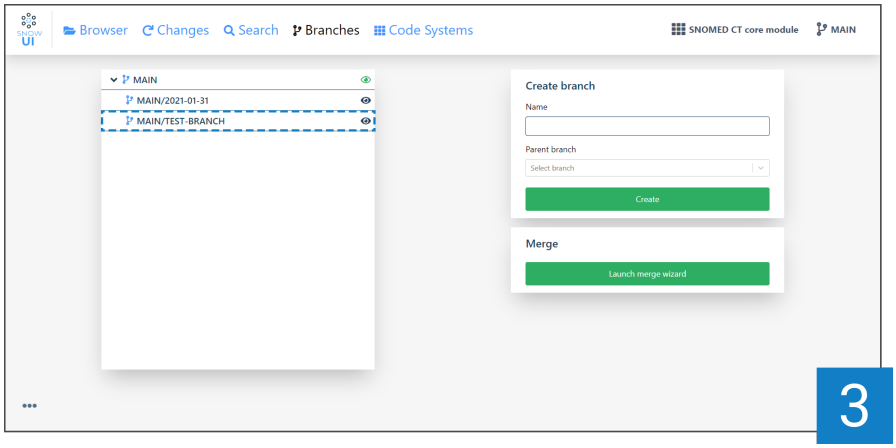
Ukázkový scénář:



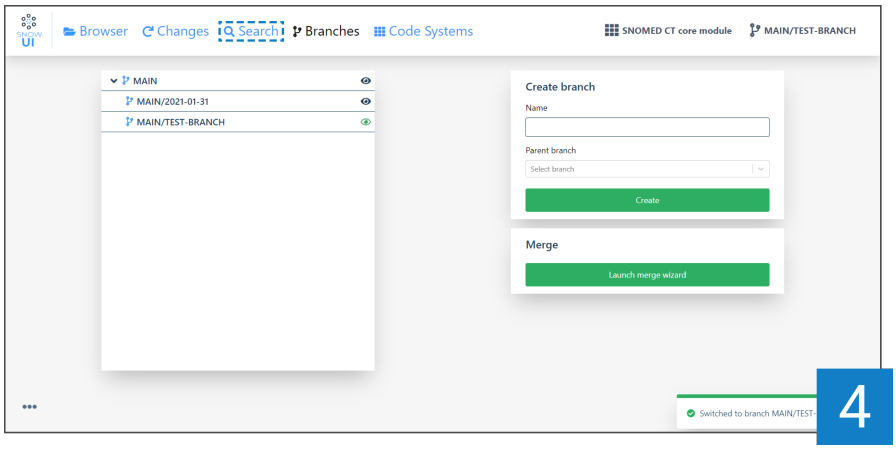
Obrázek 7.1: Uživatel otevře správce větví a přejde do formuláře pro tvorbu nové větve



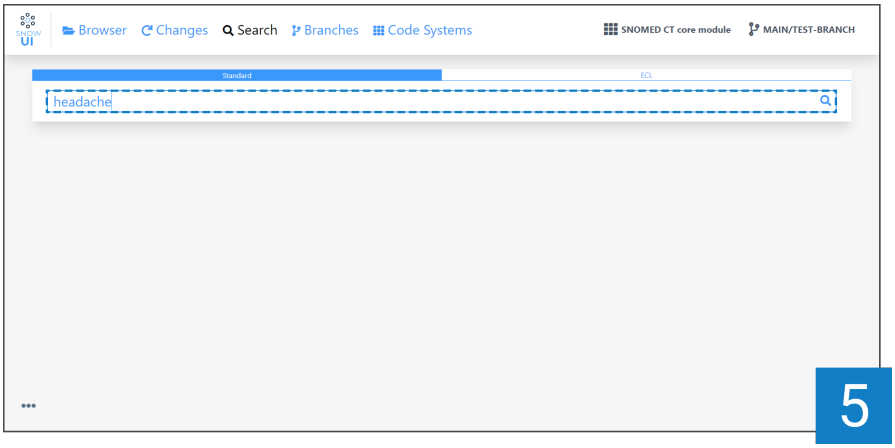
Obrázek 7.2: Uživatel vyplní vlastnosti nové větve, název `TEST-BRANCH` a rodičovskou větev `MAIN`



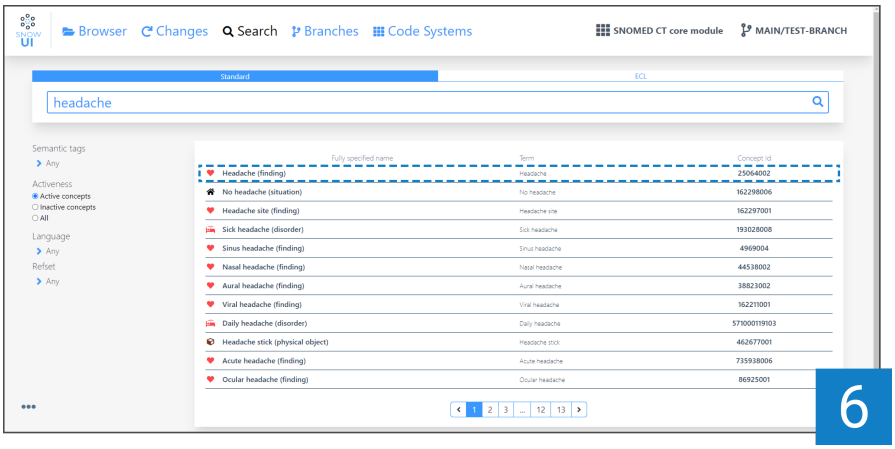
Obrázek 7.3: Systém vytvoží novou větev `MAIN/TEST-BRANCH`. Tu si uživatel zvolí jako pracovní větev.



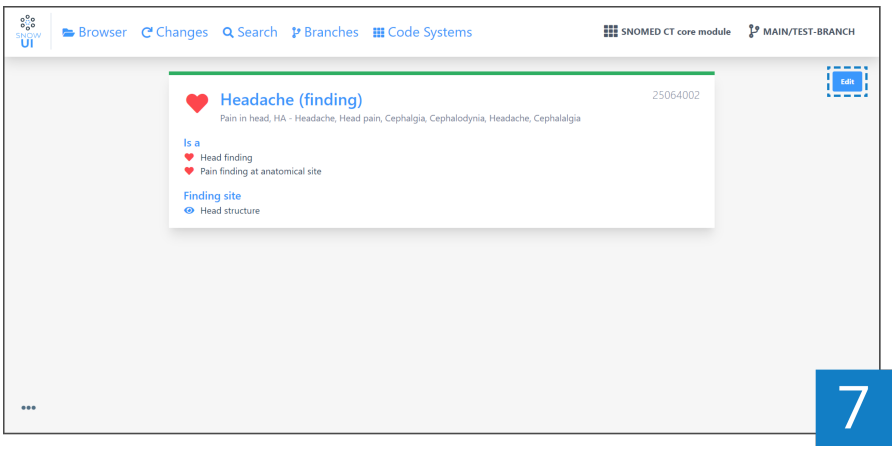
Obrázek 7.4: Systém uživatele notifkuje o změně větce. Uživatel následně přejde do fulltextového vyhledávače.



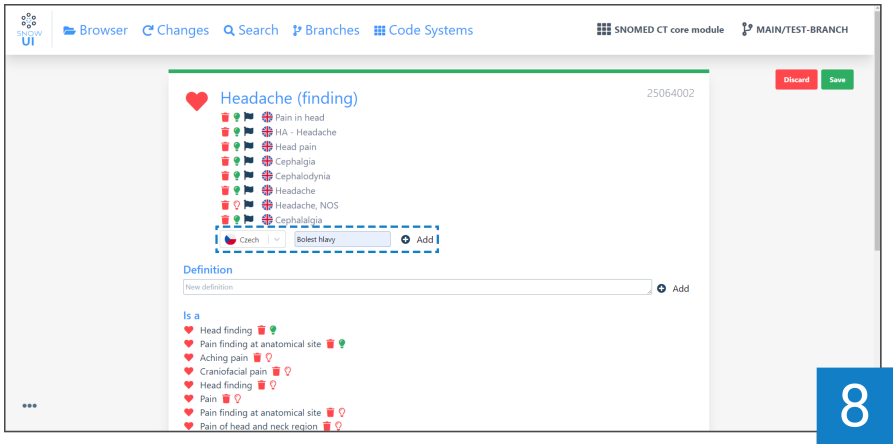
Obrázek 7.5: Uživatel pomocí fulltextového vyhledávače vyhledá řetězec ”headache”



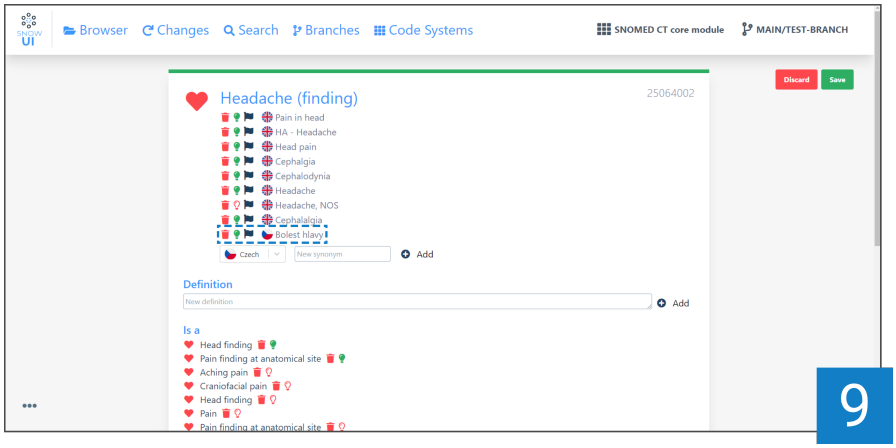
Obrázek 7.6: V nalezených výsledcích otevře koncept `25064002 |Headache (finding)|`



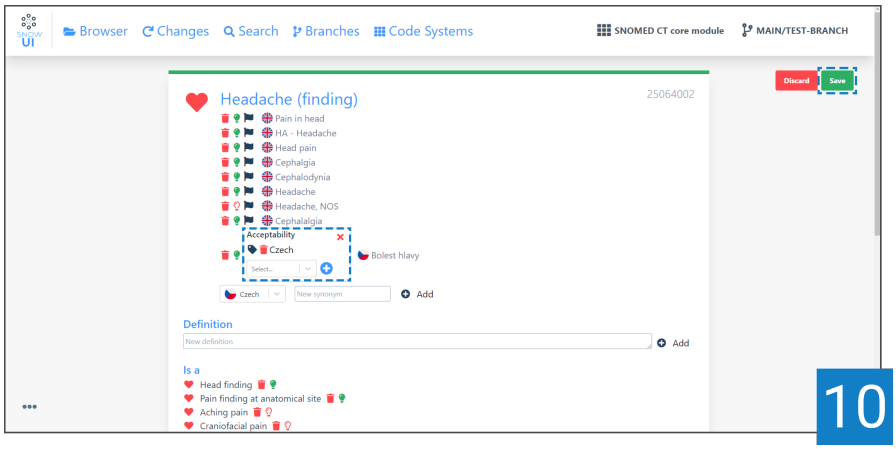
Obrázek 7.7: Uživatel přejde na detailu konceptu do editoru pomocí tlačítka `Edit`



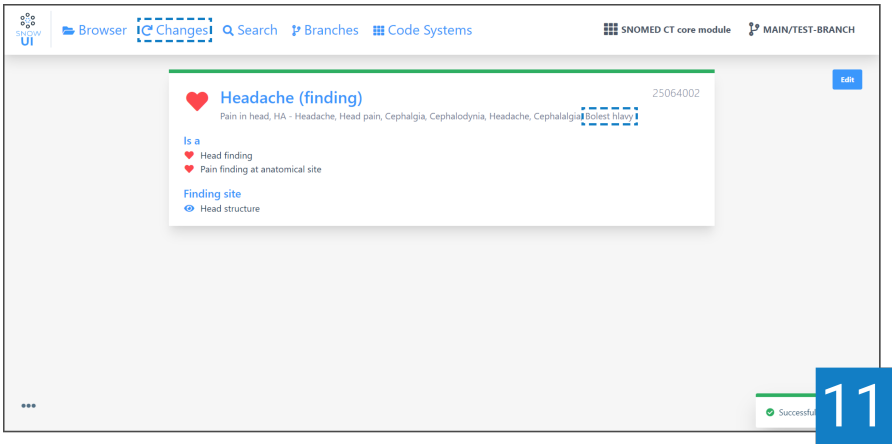
Obrázek 7.8: Uživatel nastaví jazyk synonyma na český jazyk, vyplní text ”Bolest hlavy” a klikne na tlačítko `Add`



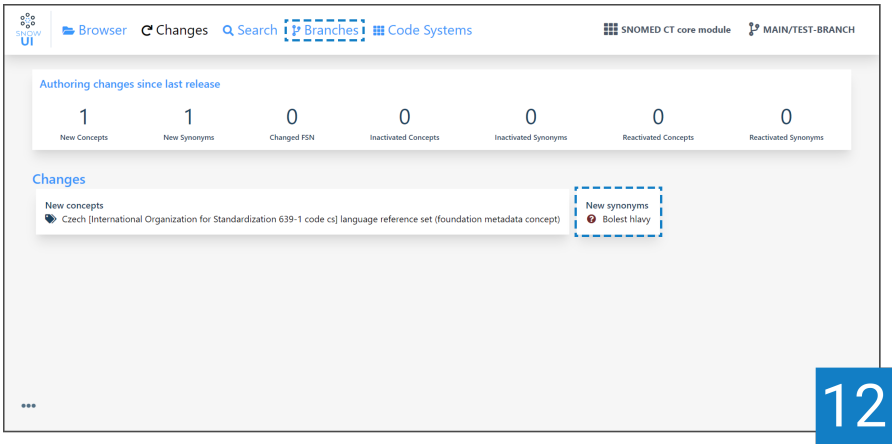
Obrázek 7.9: Uživatel klikne na ikonku vlaječky u českého synonyma, čímž otevře nastavení přijatelnosti



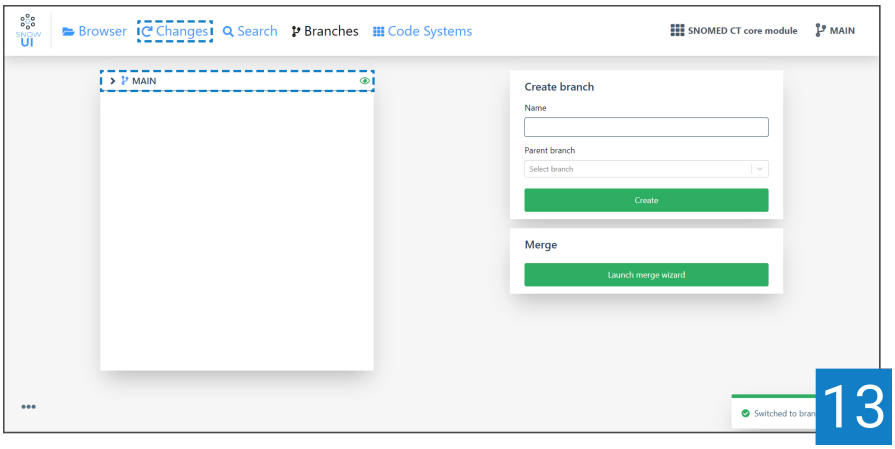
Obrázek 7.10: Uživatel nastaví synonymum jako preferovaný termín pro český jazyk a uloží změny v konceptu tlačítek `Save`



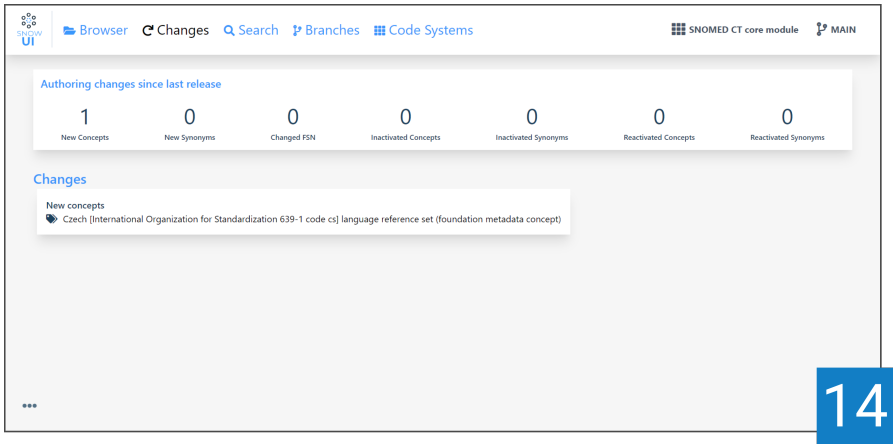
Obrázek 7.11: Systém zobrazí upravené synonymum v režimu pro čtení. Uživatel přejde na přehled změn.



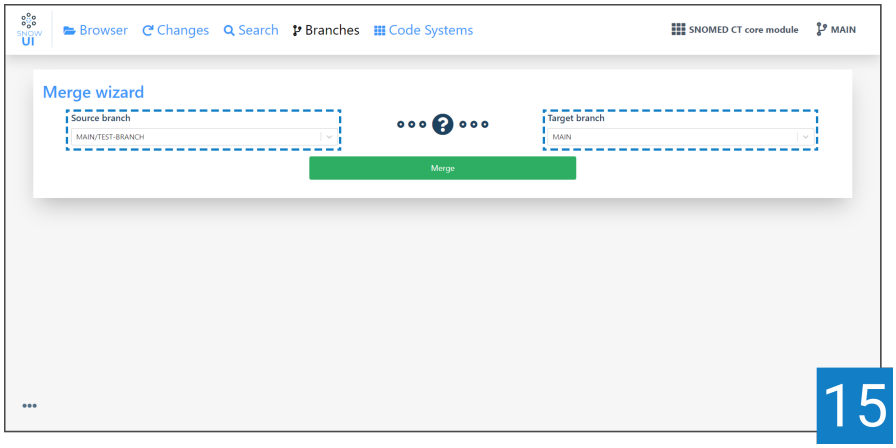
Obrázek 7.12: Zde vidí, že ve větvi `MAIN/TEST-BRANCH` skutečně přibylo synonymum ”Bolest hlavy”.



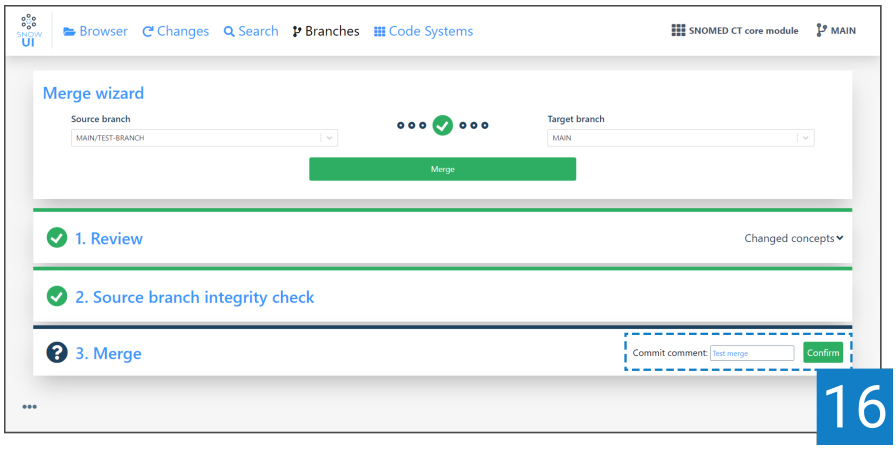
Obrázek 7.13: Uživatel přejde do správce větví a přepne se na pracovní větev `MAIN`.



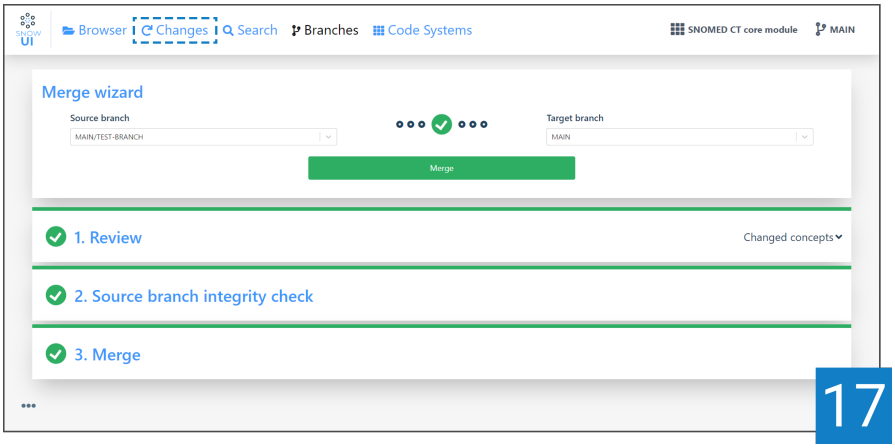
Obrázek 7.14: Na přehledu změn se přesvědčí, že ve větvi `MAIN` toto synonymum prozatím neexistuje.



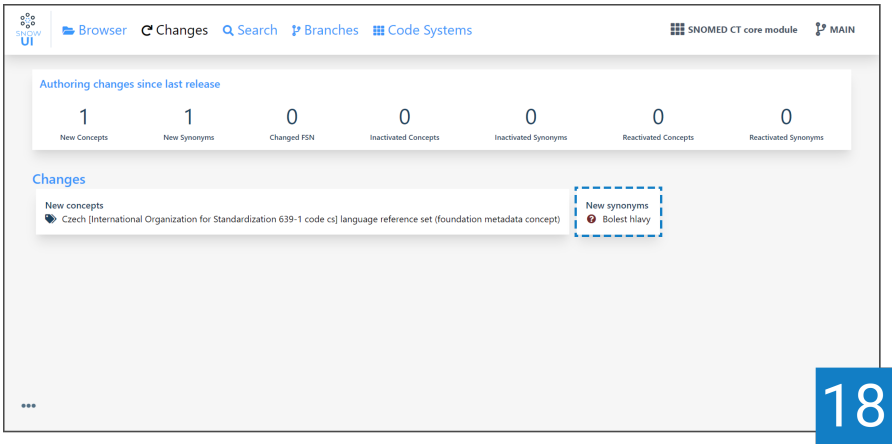
Obrázek 7.15: Uživatel přejde do průvodce sloučením větví a zahájí sloučení větví `MAIN/TEST-BRANCH` -> `MAIN`.



Obrázek 7.16: Systém provede kontrolu kompatibiltiy větví. Uživatel vyplní komentář ke sloučení a potvrdí.



Obrázek 7.17: Systém sloučí větve.



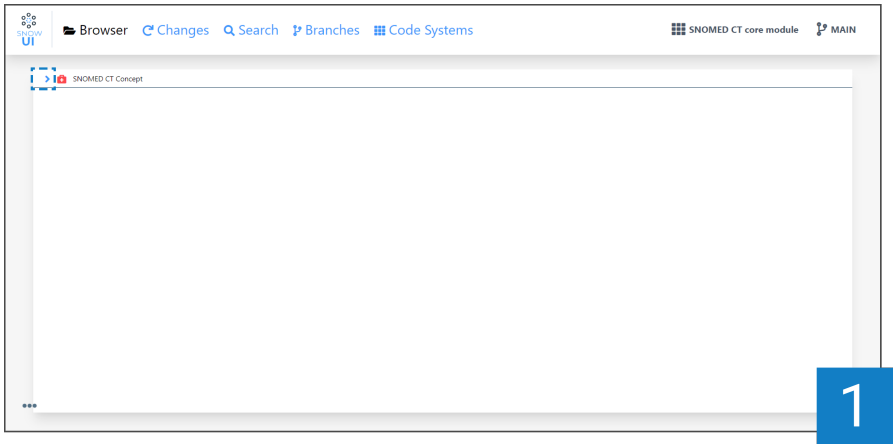
Obrázek 7.18: Uživatel se na přehledu změn přesvědčí, že nové synonymum existuje i ve větvi `MAIN`.

7.2 Vyhledávání

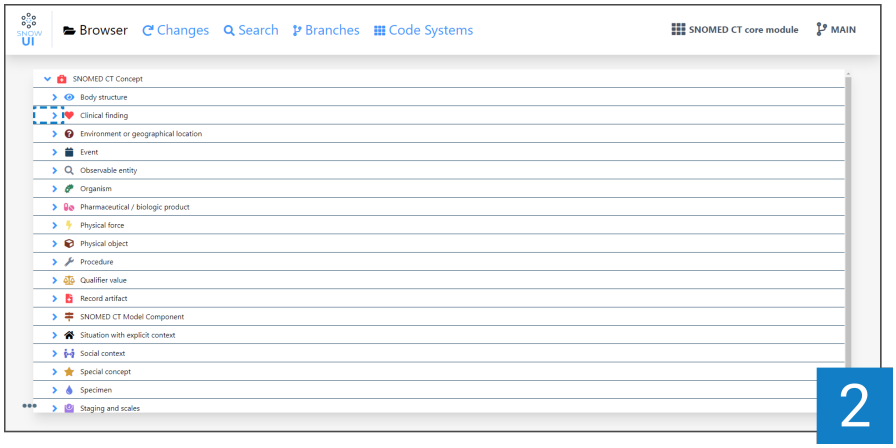
Typickou aktivitou je vyhledávávání určitého konceptu v rámci terminologie. Aplikace nabízí tři rozdílné přístupy.

7.2.1 Hierarchický prohlížeč

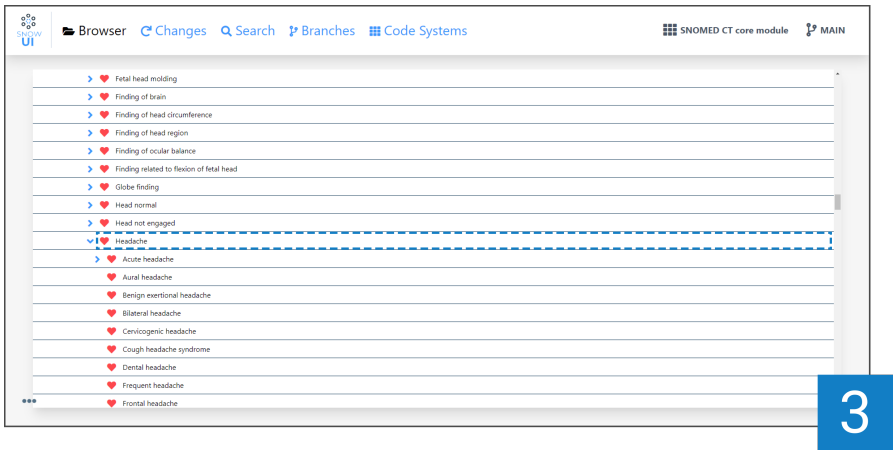
V tomto scénáři zkusíme najít koncept bolesti hlavy za pomoci hierarchického prohlížeče.



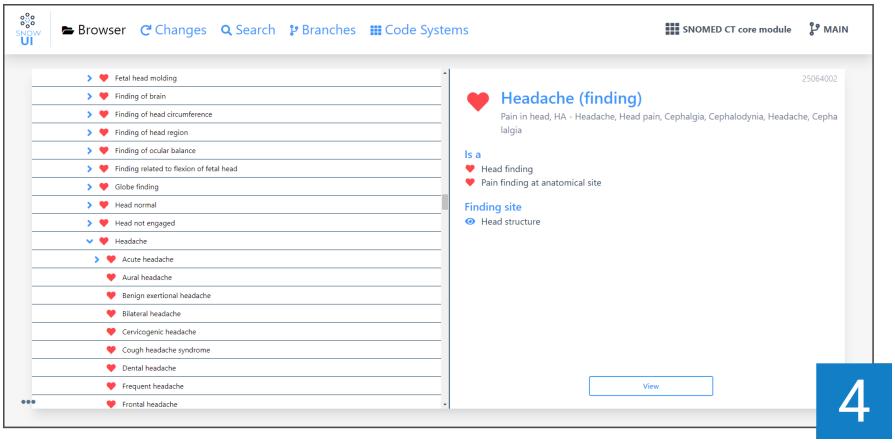
Obrázek 7.19: Uživatel otevře hierarchický prohlížeč a expanduje kořenový koncept



Obrázek 7.20: Uživatel expanduje koncepty v pořadí Clinical finding/Finding by site/Finding of body region/Finding of head and neck region/Head finding



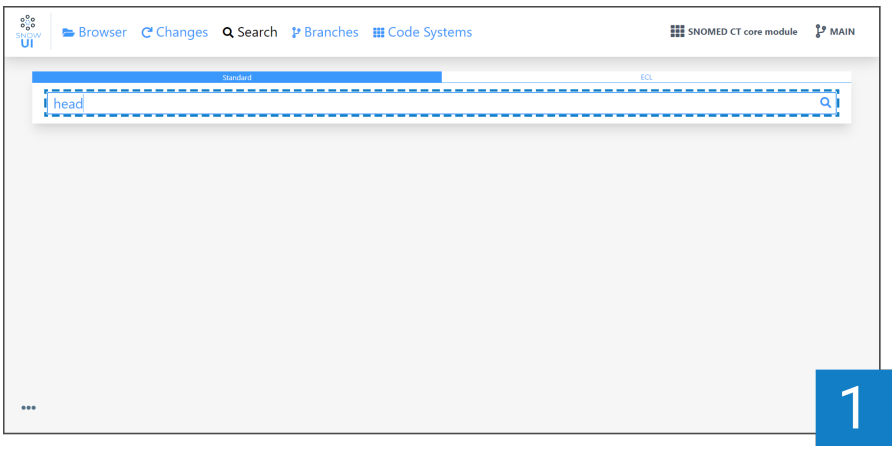
Obrázek 7.21: Kliknutím na `Headache` uživatel exapanduje detail konceptu



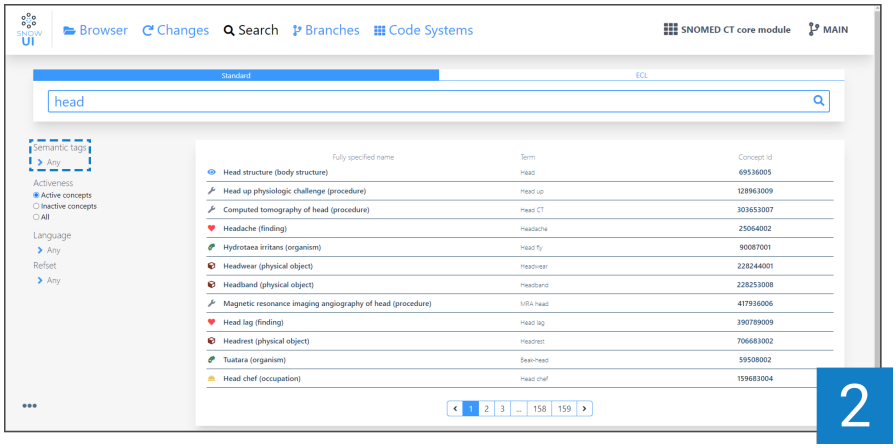
Obrázek 7.22: Uživatel si může prohlédnout detail konceptu a případně ho i otevřít

7.2.2 Fulltextový vyhledávač

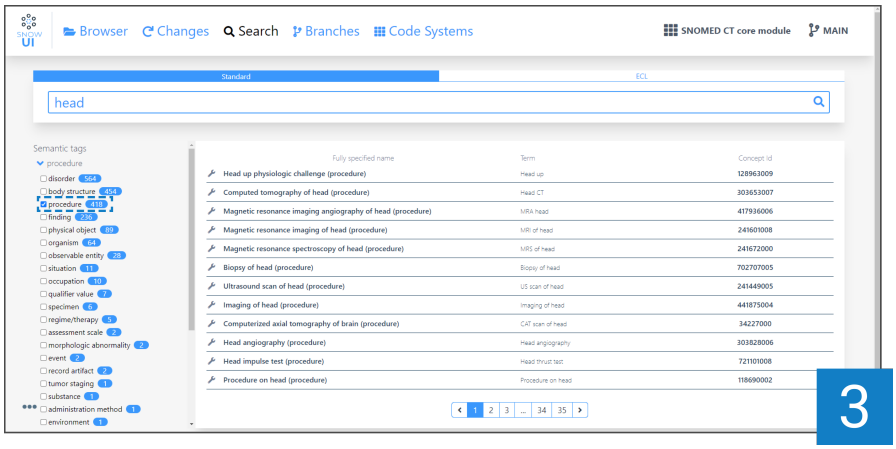
Tento scénář demonstruje funkci fulltextového vyhledávače. Výsledkem jsou všechny koncepty obsahující slovo ”head”, které jsou procedura.



Obrázek 7.23: Uživatel otevře fulltextový vyhledávač a zadá klíčové slovo `head`.



Obrázek 7.24: Pomocí postraního filtru sémantických značek vyfiltruje pouze koncepty typu procedura.

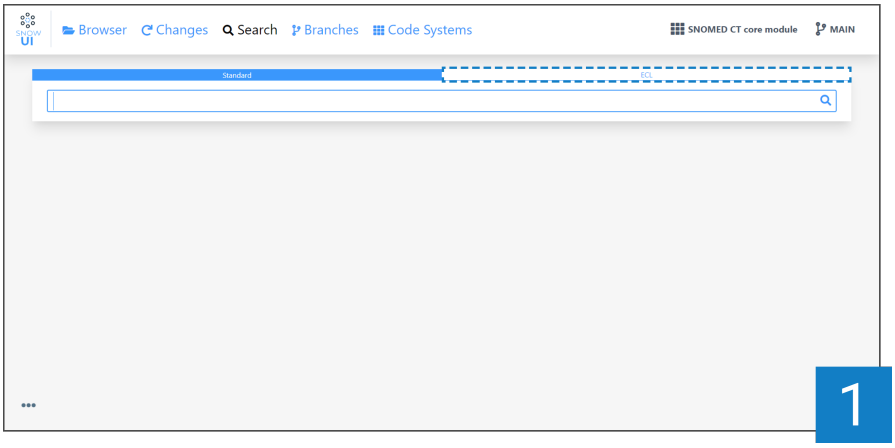


Obrázek 7.25: Výsledkem je tabulka konceptů, které odpovídají krtiériím.

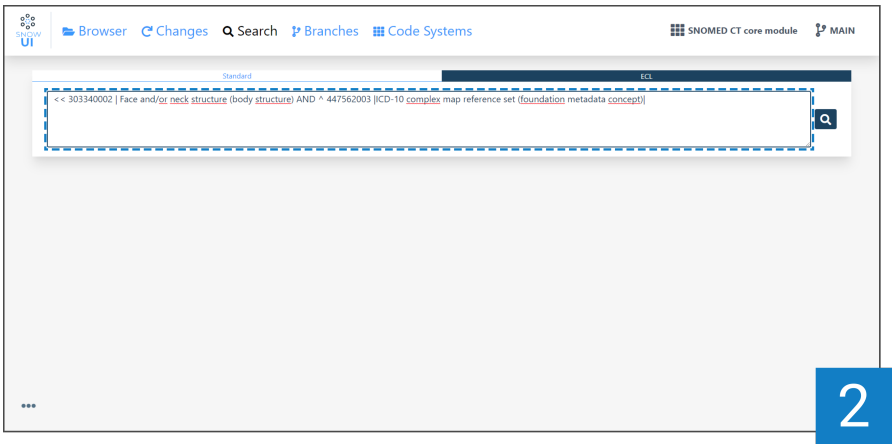
7.2.3 ECL Vyhledávač

Dalším modelovým scénářem je využití ECL vyhledávače. V následujícím scénáři se jedná konkrétně o dotaz `« 303340002 | Face and/or neck structure (body structure) AND ^447562003 |ICD-10 complex map reference set (foundation metadata concept)|`, který vrátí všechny koncepty označují části obličeje nebo krku a které jsou mapovány na ICD-10.

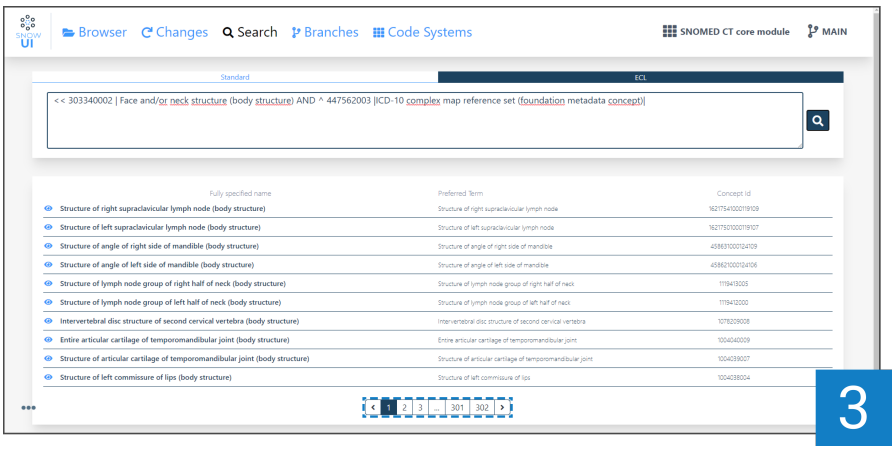
Ukázkový scénář:



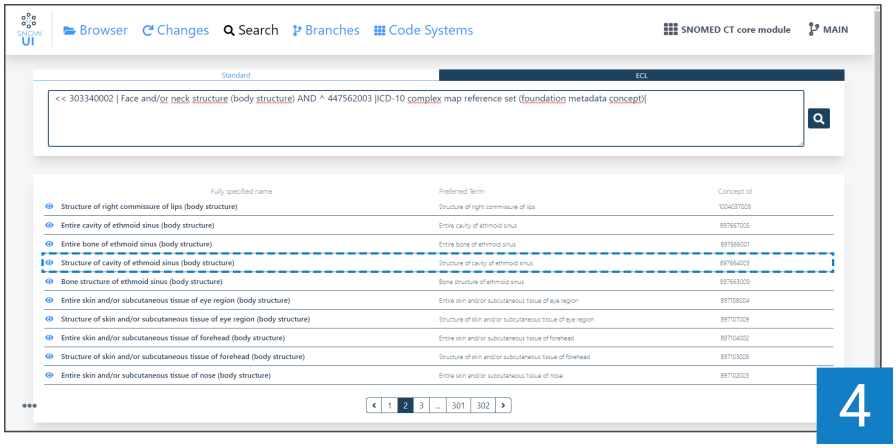
Obrázek 7.26: Uživatel otevře fulltextový vyhledávač



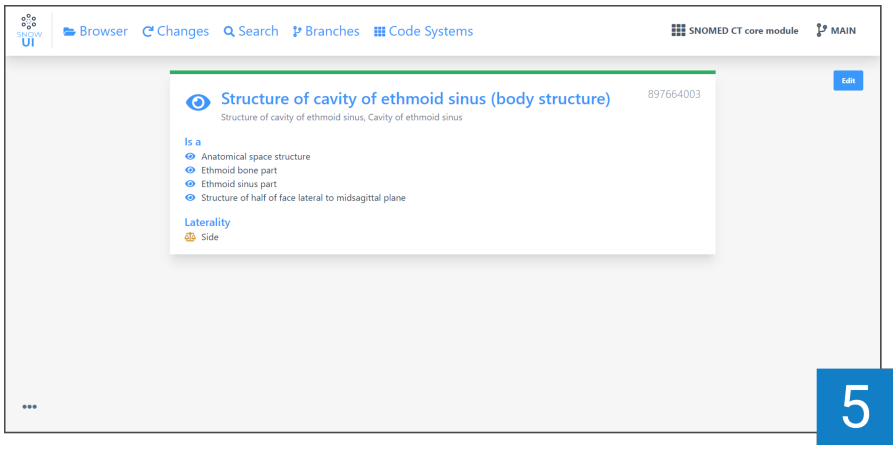
Obrázek 7.27: Uživatel vyplní ECL dotaz a vyhledá



Obrázek 7.28: Uživatel může využít stránkování k zobrazení více výsledků



Obrázek 7.29: Uživatel může otevřít detail jednotlivých vyhledaných konceptů



Obrázek 7.30: Detail konceptu

Kapitola 8

Závěr

8.1 Zhodnocení

Hlavní motivací této bakalářské práce bylo vytvoření přehledného a jednoduchého prohlížeče, potažmo editoru terminologie SNOMED CT.

Výsledná webová aplikace svou uživatelskou přívětivostí značně převyšuje všechny konkurenční nástroje. Aplikace nabízí interaktivní, výkonný a uživatelsky atraktivní prohlížeč. V terminologii lze snadno vyhledávat pojmy a uživateli je rovněž interaktivní formou zobrazit. Díky tomuto mohou do problematiky medicínské terminologie proniknout i technicky méně zdatní jedinci.

Implementovaný editor umožňuje základní autorské operace, jakými jsou tvorba nových a úprava stávajících konceptů. Změny provádí v rámci vývojových větví.

Vzhledem k velmi rozmanitému množství editačních operací, které lze nad konceptem provádět, nebylo v rámci rozsahu bakalářské práce možno tyto operace implementovat všechny (např. úprava case-significance u popisů). Zároveň průvodce sloučením větví nepodporuje některé stavy, ve kterých se větve můžou nacházet a tím pádem neumožní jejich sloučení. Rovněž některé operace jsou obtížněji proveditelné, například tvorba nového modulu, z důvodu chybně navrženého uživatelského rozhraní v důsledku nedostatečné znalosti SNOMED CT při počátečním návrhu.

Seznam použitých zkratek

API - Application Programming Interface
CAP - College of American Pathologist
CSS - Cascading Style Sheets
CTV3 - Clinical Term Vesrion 3
DOM - Document object model
ECL - Expression Constraint Language
EHR - Electronic health record
ICD - International Classification of Diseases
IHTSDO - International Health Terminology Standards Development Organisation
NLM - National Library of Medicine
REST - Representational State Transfer
SCTID - SNOMED CT identifier
SPA - Single-page application
ÚZIS - Ústav zdravotnických informací a statistiky ČR
VPS - Virtuální privátní server

Literatura

[1] b2ihealthcare. b2ihealthcare. http://b2i.sg/, 2020. [Online; accessed 19-December-2020].

[2] b2ihealthcare. b2ihealthcare / snow-owl. https://github.com/b2ihealthcare/snow-owl, 2020. [Online; accessed 19-December-2020].

[3] T. Benson. *Principles of health interoperability : SNOMED CT, HL7 and FHIR*. Springer, London, 2016.

[4] J. G. Betts, K. A. Young, J. A. Wise, E. Johnson, B. Poe, D. H. Kruse, O. Korol, J. E. Johnson, M. Womble, and P. DeSaix. *Anatomy and Physiology*. OpenStax, 2013.

[5] E. B.V. What is elasticsearch? https://www.elastic.co/what-is/elasticsearch, 2021. [Online; accessed 05-January-2021].

[6] Cloudflare. What is a reverse proxy? https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/, 2021. [Online; accessed 13-May-2021].

[7] W. Commons. File:ngrx-redux-pattern-diagram.png — wikimedia commons, the free media repository, 2020. [Online; accessed 4-May-2021].

[8] Cypress. Cypress. https://www.cypress.io/, 2021. [Online; accessed 03-May-2021].

[9] S. Daityari. Angular vs react vs vue: Which framework to choose in 2021, 2021. [Online; accessed 12-May-2021].

[10] L. Encrypt. Let's enrypt. https://letsencrypt.org/, 2021. [Online; accessed 13-May-2021].

[11] F5. Nginx. https://www.nginx.com/, 2021. [Online; accessed 13-May-2021].

[12] Google. Angular. https://angular.io/, 2020. [Online; accessed 26-December-2020].

[13] IHTSDO. Ihtsdo / snowstorm. https://github.com/IHTSDO/snowstorm, 2020. [Online; accessed 19-December-2020].

[14] IHTSDO. Ihtsdo / authoring-ui. https://github.com/IHTSDO/authoring-ui, 2021. [Online; accessed 10-Ma-2021].

[15] IHTSDO. Snomed ct browser. https://browser.ihtsdotools.org/?perspective=full&conceptId1=25064002&edition=MAIN/2021-01-31&release=&languages=en, 2021. [Online; accessed 04-January-2021].

[16] F. Inc. React. https://reactjs.org/, 2020. [Online; accessed 26-December-2020].

[17] LOINC. Loinc and snomed ct link. https://loinc.org/collaboration/snomed-international, 2021. [Online; accessed 10-May-2021].

[18] Microsoft. Typescript. https://www.typescriptlang.org/, 2021. [Online; accessed 04-May-2021].

[19] D. R. Murphy, A. N. Meyer, D. F. Sittig, D. W. Meeks, E. J. Thomas, and H. Singh. Application of electronic trigger tools to identify targets for improving diagnostic safety. *BMJ Quality & Safety*, 28(2):151–159, 2019.

[20] National Library of Medicine. Health information technology and health data standards at nlm. https://www.nlm.nih.gov/healthit/index.html, 2020. [Online; accessed 21-December-2020].

[21] National Library of Medicine. Snomed ct. https://www.nlm.nih.gov/healthit/snomedct/snomed_overview.html, 2021. [Online; accessed 04-January-2021].

[22] National Library of Medicine. Snomed ct. https://www.nlm.nih.gov/research/umls/mapping_projects/snomedct_to_icd10cm.html, 2021. [Online; accessed 10-May-2021].

[23] Postman. Postman. https://www.postman.com/, 2020. [Online; accessed 30-April-2021].

[24] J. Potter. react vs vue vs @angular/core. https://www.npmtrends.com/react-vs-vue-vs-@angular/core, 2020. [Online; accessed 19-December-2020].

[25] P. Ruch, J. Gobeill, C. Lovis, and A. Geissbühler. Automatic medical encoding with SNOMED categories. *BMC Medical Informatics and Decision Making*, 8(S1):S6, Oct. 2008.

[26] SNOMED International. Snomed ct expression constraint language. https://confluence.ihtsdotools.org/display/SLPG/SNOMED+CT+Expression+Constraint+Language, 2020. [Online; accessed 27-December-2020].

[27] SNOMED International. Snomed ct logical model. https://confluence.ihtsdotools.org/display/DOCSTART/5.+SNOMED+CT+Logical+Model, 2020. [Online; accessed 27-December-2020].

[28] SNOMED International. Check-digit computation. https://confluence.ihtsdotools.org/display/DOCRELFMT/6.4.2+Check-digit+Computation, 2021. [Online; accessed 10-May-2021].

[29] SNOMED International. Extensions practical guide. https://confluence.ihtsdotools.org/display/DOCEXTPG/Extensions+Practical+Guide, 2021. [Online; accessed 15-April-2021].

[30] SNOMED International. Modules. https://confluence.ihtsdotools.org/display/DOCEXTPG/4.2+Modules, 2021. [Online; accessed 28-April-2021].

[31] SNOMED International. Snomed ct authoring platform. https://confluence.ihtsdotools.org/display/SIAPUG/SNOMED+International+Authoring+Platform+User+Guide, 2021. [Online; accessed 04-January-2021].

[32] SNOMED International. Snomed ct reference set. https://confluence.ihtsdotools.org/display/DOCGLOSS/reference+set, 2021. [Online; accessed 12-April-2021].

[33] SNOMED International. Snomed ct semantic tags. https://confluence.ihtsdotools.org/display/DOCEG/Semantic+Tags, 2021. [Online; accessed 08-January-2021].

[34] SNOMED International. Snomed ct textual definition. https://confluence.ihtsdotools.org/display/DOCGLOSS/textual+definition, 2021. [Online; accessed 15-April-2021].

[35] SNOMED International. Snomed international account. https://confluence.ihtsdotools.org/display/ILS/Confluence%2BUser%2BAccounts, 2021. [Online; accessed 10-May-2021].

[36] T. softwaru. Fáze a úrovně provádění testů. http://testovanisoftwaru.cz/tag/integracni-testovani/#integration, 2021. [Online; accessed 03-May-2021].

[37] B. Szczeciński. Server side rendering. https://medium.com/@baphemot/whats-server-side-rendering-and-do-i-need-it-cb42dc059b38, 2021. [Online; accessed 04-May-2021].

[38] Tailwindcss. Tailwindcss. https://tailwindcss.com/, 2020. [Online; accessed 27-December-2020].

[39] B. team. Bootstrap. https://getbootstrap.com/, 2020. [Online; accessed 27-December-2020].

[40] Vercel. Next.js. https://reactjs.org/, 2020. [Online; accessed 27-December-2020].

[41] Vercel. Swr. https://swr.vercel.app/, 2021. [Online; accessed 04-May-2021].

[42] Vue.js. Components basics. https://vuejs.org/v2/guide/components.html, 2021. [Online; accessed 05-January-2021].

[43] E. You. Vue.js. https://reactjs.org/, 2020. [Online; accessed 26-December-2020].

[44] Ústav zdravotnických informací a statistiky ČRl. Národní centrum pro medicínské nomenklatury a klasifikace (ncmnk). https://www.uzis.cz/index.php?pg=o-nas--projekty&prid=23, 2021. [Online; accessed 11-May-2021].

[45] Ústav zdravotnických informací a statistiky ČRl. Snomed ct. https://www.uzis.cz/index.php?pg=registry-sber-dat--klasifikace--snomed-ct, 2021. [Online; accessed 19-April-2021].

Příloha A

Případy užití

A.1 UC1: Zobrazit všechny koncepty v hierarchickém stromu

Use Case 1	**Zobrazit všechny koncepty v hierarchickém stromu**
Popis:	Umožňuje procházet koncepty v hierarchickém pořadí. Základem je kořenový koncept. Každý koncept lze expandovat a zobrazit jeho přímé potomky.
Primární aktér:	Koncový uživatel
Předpoklady:	

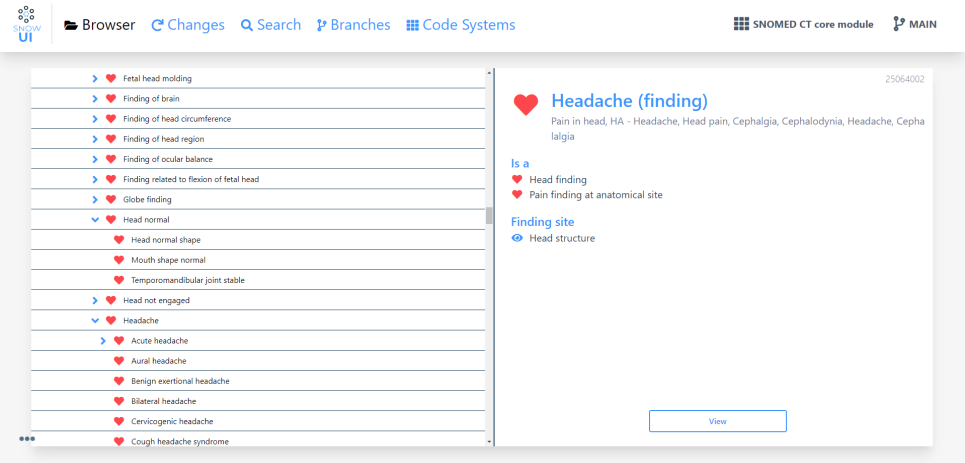
Hlavní scénář průchodu:

1. **Systém** zobrazí kořenový koncept s možností expanze jeho přímých potomků
2. IF **Uživatel** otevře detail konceptu THEN **Systém** zobrazí detail konceptu
3. IF **Uživatel** expanduje koncept THEN **Systém** zobrazí zobrazí přímé potomky konceptu

Alternativní scénáře:

3.a Koncový koncept:

1. Systém skryje akci pro expanzi konceptu
2. Uživatel je navrácen do kroku 2



Obrázek A.1: Obrazovka hierarchického prohlížeče

A.2 UC2: Zobrazit detail konceptu

Use Case 2	**Zobrazit detail konceptu**
Popis:	Umožňuje zobrazení detailu konkrétního konceptu v režimu jen pro čtení. Zobrazuje všechny příslušné vlastnosti jako synonyma, definice, *IS A* a další vztahy. Umožňuje přístup do editačního režimu.
Primární aktér:	Koncový uživatel
Předpoklady:	

Hlavní scénář průchodu:

1. **Systém** zobrazí koncept a příslušné vlastnosti
2. IF **Uživatel** si přeje editovat koncept THEN **Systém** přesměruje do editačního módu (UC3)

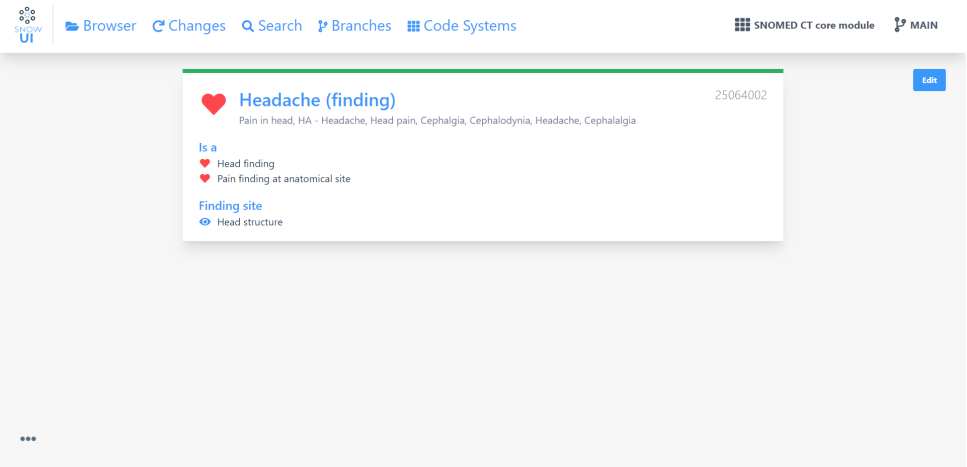
Alternativní scénáře:

1.a Koncept neexistuje:

1. Systém zobrazí chybovou hlášku o neexistenci konceptu
2. Uživatelký scénář končí

2.a Uživatel nemá oprávnění provádět změny:

1. Systém zobrazí chybovou hlášku o nedostatečném oprávnění
2. Uživatel je navrácen do kroku 1



Obrázek A.2: Obrazovka detailu konceptu

A.3 UC3: Editor konceptů

Use Case 3	**Editor konceptů**
Popis:	Umožňuje upravovat vlastnosti konceptu, jakými jsou synonyma, úplný název, vztahy a definice. Zajišťuje ukládání provedených změn.
Primární aktér:	Koncový uživatel
Předpoklady:	

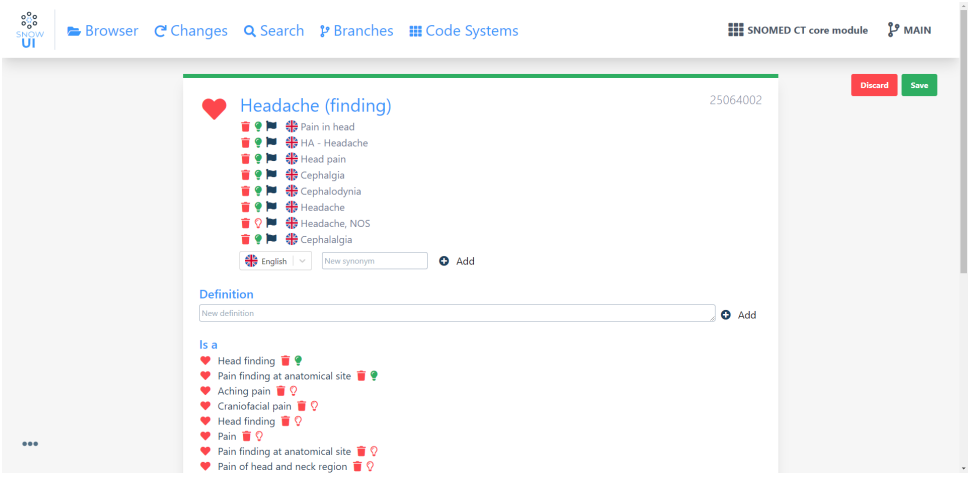
Hlavní scénář průchodu:

1. **Systém** zobrazí koncept a jeho vlastnosti v editovatelné podobě
2. **Uživatel** provede požadované změny do konceptu
3. **Uživatel** potvrdí uložení změn
4. **Systém** aktualizuje vybraný koncept

Alternativní scénáře:

3.a **Uživatel** zruší provedené změny

1. Systém zahodí změny a přepne koncept do režimu pro čtení



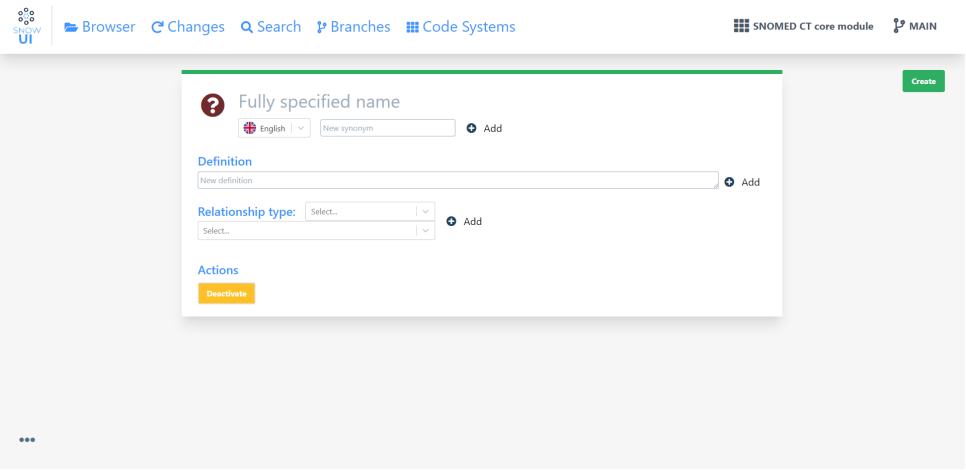
Obrázek A.3: Obrazovka pro editor konceptů

A.4 UC4: Nový koncept

Use Case 4	Nový koncept
Popis:	Umožňuje tvorbu nového konceptu a definování jeho vlastností.
Primární aktér:	Koncový uživatel
Předpoklady:	

Hlavní scénář průchodu:

1. **Systém** zobrazí formulář pro tvorbu nového konceptu
2. **Uživatel** vyplní požadované vlastnosti
3. **Uživatel** potvrdí vytvoření konceptu
4. **Systém** vytvoří nový koncept
5. **Systém** přesměruje uživatele na stránku detailu konceptu (UC2)



Obrázek A.4: Obrazovka tvorby nového konceptu

A.5 UC5: Fulltextový vyhledávač

Use Case 5	**Fulltextový vyhledávač**
Popis:	Umožňuje vyhledávat mezi koncepty pomocí fulltextového vyhledávání. Výsledky jsou filtrovatelné dle parametrů.
Primární aktér:	Koncový uživatel
Předpoklady:	

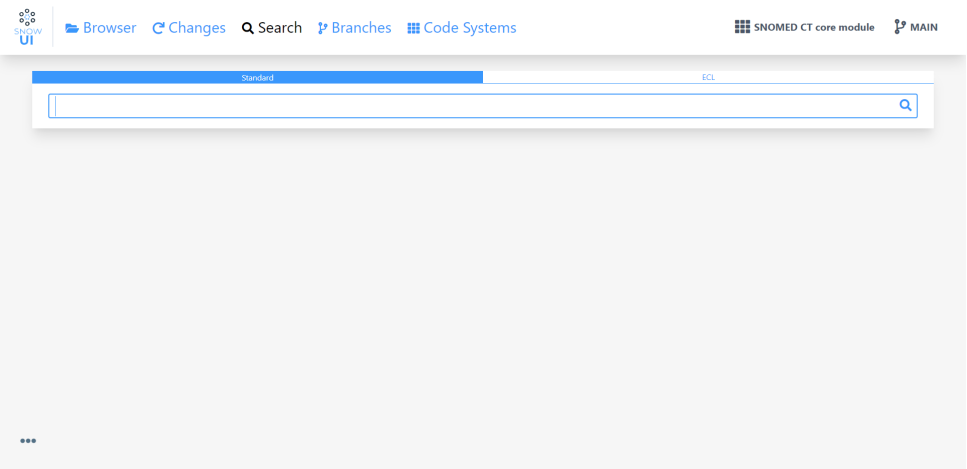
Hlavní scénář průchodu:

1. **Systém** zobrazí vstup pro vyhledávaný textový řetězec
2. **Uživatel** vyplní a potvrdí vyhledávaný řetězec
3. **Systém** zobrazí stránkovatelné výsledky a filtry
4. IF **Uživatel** použije filtr THEN **Systém** zobrazí filtrované koncepty
5. IF **Uživatel** vybere vyhledaný koncept THEN **Systém** ho přesměruje na detail konceptu (UC2)

Alternativní scénáře:

3.a Textovému řetězci neodpovídá žádná shoda:

1. Systém zobrazí hlášku o nulové shodě



Obrázek A.5: Obrazovka fulltextového vyhledávače

A.6 UC6: ECL vyhledávač

Use Case 6	**ECL vyhledávč**
Popis:	Umožňuje vyhledávat mezi koncepty pomocí dotazovacího jazyka ECL (data-centric programming language).
Primární aktér:	Koncový uživatel
Předpoklady:	

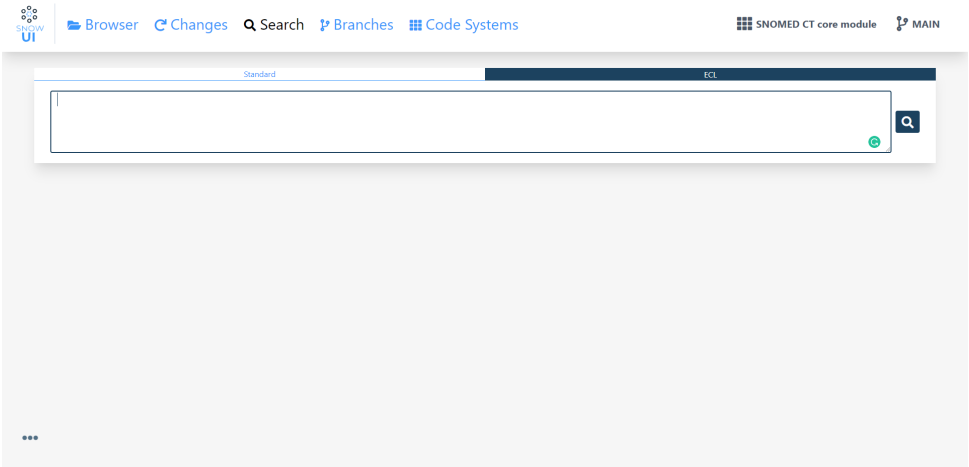
Hlavní scénář průchodu:

1. **Systém** zobrazí vstup pro vyhledávaný ECL dotaz
2. **Uživatel** zkonstruuje a potvrdí vyhledávaný dotaz
3. **Systém** zobrazí stránkovatelné výsledky
4. IF **Uživatel** vybere vyhledaný koncept THEN **Systém** ho přesměruje na detail konceptu (UC2)

Alternativní scénáře:

3.a ECL dotazy neodpovídá žádná shoda:

1. Systém zobrazí hlášku o nulové shodě



Obrázek A.6: Obrazovka ECL vyhledávače

A.7 UC7: Prohlížeč vývojových větví

Use Case 7	**Prohlížeč vývojových větví**
Popis:	Umožňuje zobrazení vývojových větví v hierarchickém uspořádání. Zároveň umožňuje volbu jiné pracovní větve.
Primární aktér:	Koncový uživatel
Předpoklady:	

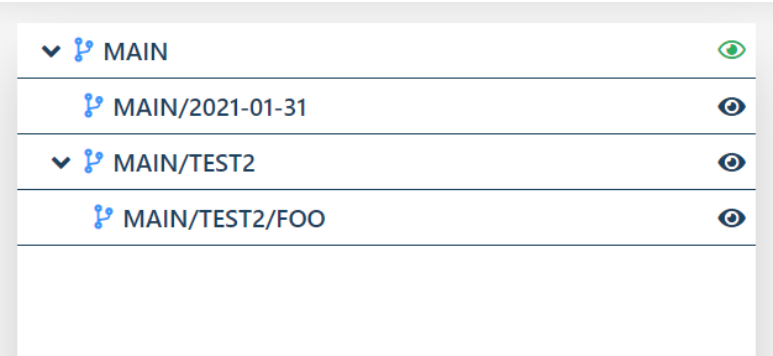
Hlavní scénář průchodu:

1. **Systém** zobrazí kořenovou větev s možností expanzí jejich přímých potomků
2. IF **Uživatel** expanduje větev THEN **Systém** zobrazí přímé potomky větve
3. IF **Uživatel** zvolí větev THEN **Systém** změní vybranou praocvní větev

Alternativní scénáře:

2.a Koncová větev:

1. Systém skryje akci pro exapanzi konceptu
2. Uživatel je navrácen do kroku 1



Obrázek A.7: Realizace hierarchického prohlížeče větví

A.8 UC8: Vytvoření nové vývojové větve

Use Case 8	**Vytvoření nové vývojové větve**
Popis:	Umožňuje vytvořit novovu vývojou větev vycházející z jiné.
Primární aktér:	Koncový uživatel
Předpoklady:	

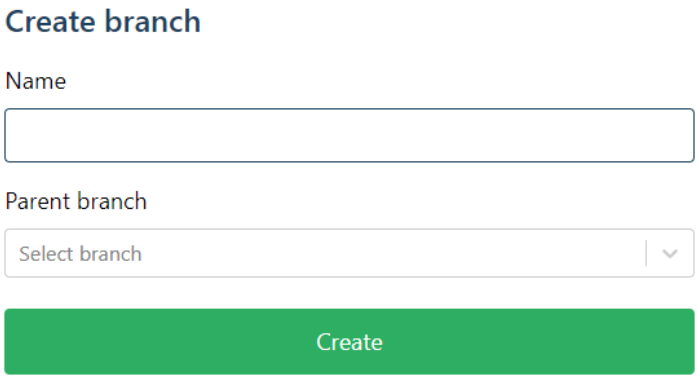
Hlavní scénář průchodu:

1. **Systém** zobrazí formulář pro tvorbu nové větve
2. **Uživatel** vyplní název nové větve a vybere rodiče
3. **Systém** vytvoří novou vývojovou větev

Alternativní scénáře:

3.a Chyba při tvorbě větve:

1. **Systém** zobrazí chybovou hlášku
2. Uživatel je navrácen do kroku 1



Obrázek A.8: Formulář pro tvorbu nové větve

A.9 UC9: Průvodce sloučením vývojových větví

Use Case 9	**Průvodce sloučením vývojových větví**
Popis:	Umožňuje snadno sloučit dvě vývojové větve. Zohledňuje všechny možné stavy, v kterých se větve mohou nacházet a provadí akce nutné k jejich sloučení.
Primární aktér:	Koncový uživatel
Předpoklady:	

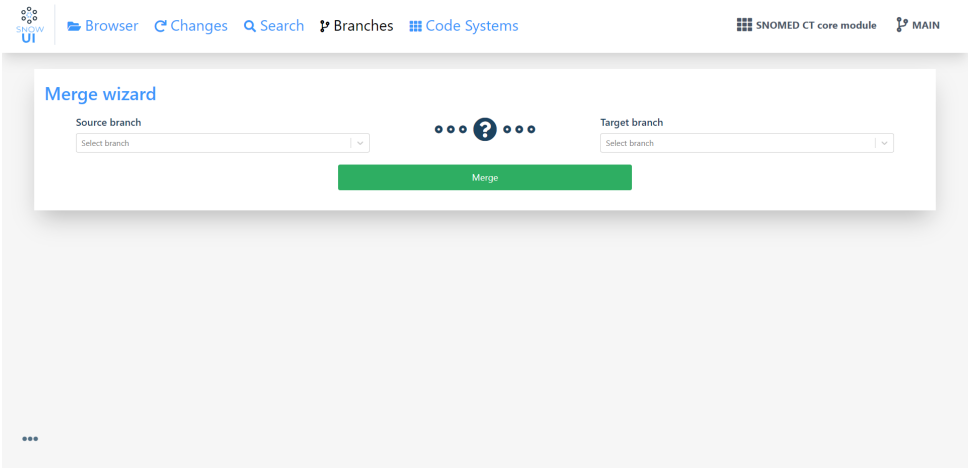
Hlavní scénář průchodu:

1. **Systém** zobrazí formulář pro výběr zdrojové a cílové větve
2. **Uživatel** provede volbu a potvrdí sloučení
3. **Systém** provede kroky nutné ke sloučení těchto větví

Alternativní scénáře:

3.a Chyba při slučování větví:

1. **Systém** zobrazí chybovou hlášku
2. Uživatel je navrácen do kroku 1



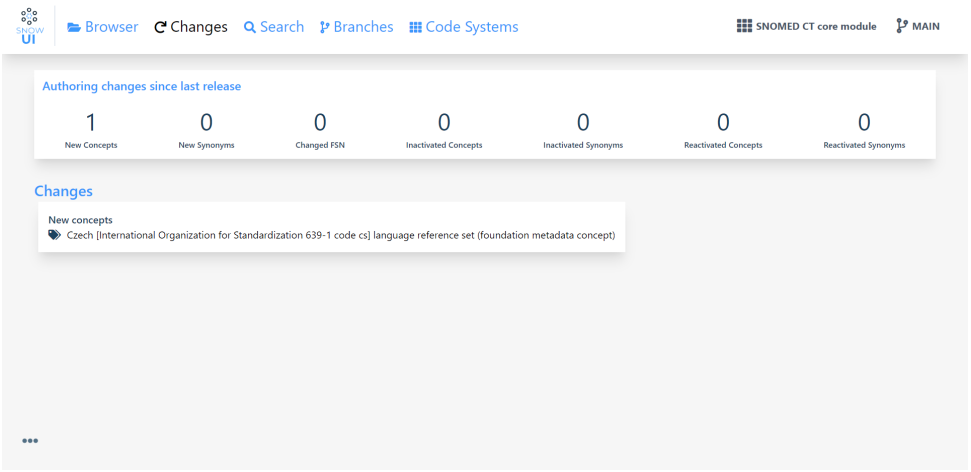
Obrázek A.9: Obrazovka slušování větví

A.10 UC10: Přehled změn

Use Case 10	**Přehled změn**
Popis:	Umožňuje zobrazit seznam provedených změn na aktuálně zvolené větvi. Konkrétně zobrazuje seznam nových, inaktivovaných a reaktivovaných konceptů a synonym.
Primární aktér:	Koncový uživatel
Předpoklady:	

Hlavní scénář průchodu:

1. **Systém** zobrazí seznam změn provedených na aktuální větvi
2. IF **Uživatel** zvolí jednu ze změn THEN **Systém** zobrazí detail změněného konceptu



Obrázek A.10: Obrazovka přehledu změn

Příloha B

Rešerše javascriptové knihovny

B.1 Kritéria

Při výběru použité javascriptové knihovny jsem zohlednil zejména následující kritéria. Seřazeny jsou abecedně, nikoli dle důležitosti. Slovní hodnocení kritérií používá středoškolskou stupnici.

- **Kvalita dokumentace** - Subjektivní kvalita ofciálně poskytované dokumentace.
- **Rozšířenost** - Rozšíření použití napříč světem.
- **Učící křivka** - Kvantifikace, jak rychle probíhá proces učení dané knihovny. [9]
- **Vývojář** - Kdo zaštiťuje vývoj.
- **Vývojářská základna** - Velikost a ochota komunita vývojářů. S větší komunitou je obecně vzato spojené snažší vyhledávání problémů.
- **Předchozí zkušenosti** - Předchozí zkušenost s využitím knihovny.

Pole javascriptových knihoven je v dnešní době doménou zejména tří hlavních hráčů.

B.2 Angular

Angular je nejstarší knihovnou z této trojice, první verze byla vydána již v roce 2010. Od počátku je vyvíjen pod taktovkou Googlu, který jej sám

hojně implementuje ve svých službách. [12] Postupem času se vyprofiloval jako řešení vhodné spíše pro velké enterprise aplikace vyvíjené velkými týmy. Jeho učící křivka je nejvíce pozvolná ze všech jmenovaných. [9]

B.3 React

React, jehož vývoj zaštiťuje Facebook, si právoplatně vydobyl pomyslnou první příčku [24] na poli javascriptových knihoven. První verze byla vydaná v roce 2013 [16]. Jeho použití je snadné v malých a zároveň výkoné i v těch největších aplikací. Sám Facebook ho používá pro svoje služby, stejně tak například jako Twitter, či Uber. Nabízí strmou učící křivku. [9]

B.4 Vue.js

Pomyslným nováčkem v poli je Vue, které se může pyšnit kompletně komunitně financovaným vývojem. Díky tomuto si ho oblíbily zejména čínské technologické společnosti jakou je například Alibaba. Vue je vyvíjeno od roku 2014, zakladatelem je ex-zaměstanec Googlu. [43] Knihovna si zakládá na flexibilitě, díky tomuto se velice snadno používá, ale zároveň je třeba dávat pozor na zásady kvalitního kódu.

B.5 Tabulka

Následující tabulka srovnává hodnotící kritéria všech uvážených řešení.

B.6 Shrnutí

Po důkladném zvážení jsem se rozhodoval mezi knihovnou React a Vue.js. Angular jsem vyloučil z důvodu absence předchozí zkušenosti a jeho obecnému zaměření spíše na rozsáhlejší projekty s velkými týmy.

Pro Vue.js hovořilo hlavně to, že jsem s jeho pomocí již dokončil dva projekty a byl jsem s ním velice spokojen. React na druhou stranu nabízel vyspělejší řešení s rozsáhlou uživatelskou základnou.

Na základě rešerše a diskuze s vedoucím práce jsme se shodli, že nejlepší bude použít knihovnu React. To především z toho důvodu, že se jedná o

Metrika	Angular	React	Vue.js
Kvalita dokumentace	Chvalitebná. Velmi obsáhlá, obtížné začátky.	Chvalitebná. Jednoduchá pro začátečníky, mnoho praktických ukázek.	Výborná. Přehledná a snadná.
Rozšířenost	Vyspělé řešení, spíše pro velké projekty.	Velmi rozšířené	Velký vzestup
Učící křivka	Pozvolná	Strmá	Velmi strmá
Hlavní vývojář	Google	Facebook	Komunita
Vývojářská základna	Chvalitebná	Výborná	Chvalitebná, rostoucí
Přechozí zkušenosti	Ne	Ne	Ano

Tabulka B.1: Tabulka shrnutí rešerše javascriptových knihoven

podstatně nejrozšířenější a případné navázání na mou práci je tak daleko snažší.

Příloha C

TypeScript

Hlavní nevýhodou používání Javascriptu je absence statické typové kontroly, kterou tento jazyk nativně nepodporuje. Tuto nevýhodu lze odstranit použitím TypeScriptu.

TypeScript je open-source programovací jazyk, který je nadstavou nad Javascript. Vývoj je zaštítěn společností Microsoft. [18]

TypeScript zajišťuje stastickou typovou kontrolu a přináší konstrukty známé z objektově orientovaného programování. TypeScript kód je automaticky kompilován do Javascriptu, proto i každý Javascript kód je validním TypeScript kódem.

Na ukázkách kódu 3 a 4 můžete porovnat syntax a podchycení chyb v obou jazycích.

```
function isAdult(person) {
  return person.age >= 18;
}

let testPersonA = { name: "John Doe", age: 21 };
let testPersonB = {
  name: { name: "John", surname: "Doe" },
  age: "21",
};

isAdult(testPersonA);
isAdult(testPersonB); // Při vykonání dojde k neočekávané chybě
isAdult("person"); // Při vykonání dojde k neočekávané chybě
```

Ukázka kódu 3: Příklad kódu v Javascriptu

```typescript
interface IPerson {
  name: string;
  age: number;
}

function isAdult(person: IPerson) {
  return person.age >= 18;
}

let testPersonA: IPerson = { name: "John Doe", age: 21 };
let testPersonB: IPerson = {
  name: { name: "John", surname: "Doe" },
  age: "21",
}; //Chyba statické kontroly: Nesprávný typ atributu "name" i "age"

isAdult(testPersonA);
isAdult("person");
/* Chyba statické kontroly:
 Do funkce je poslán parametr nesprávného typu => kód se nezkompiluje*/
```

Ukázka kódu 4: Příklad kódu v TypeScript

Příloha D

Rešerše CSS Frameworku

D.1 Rozdělení

Na poli moderních CSS frameworků lze pozorovat dva hlavní proudy.

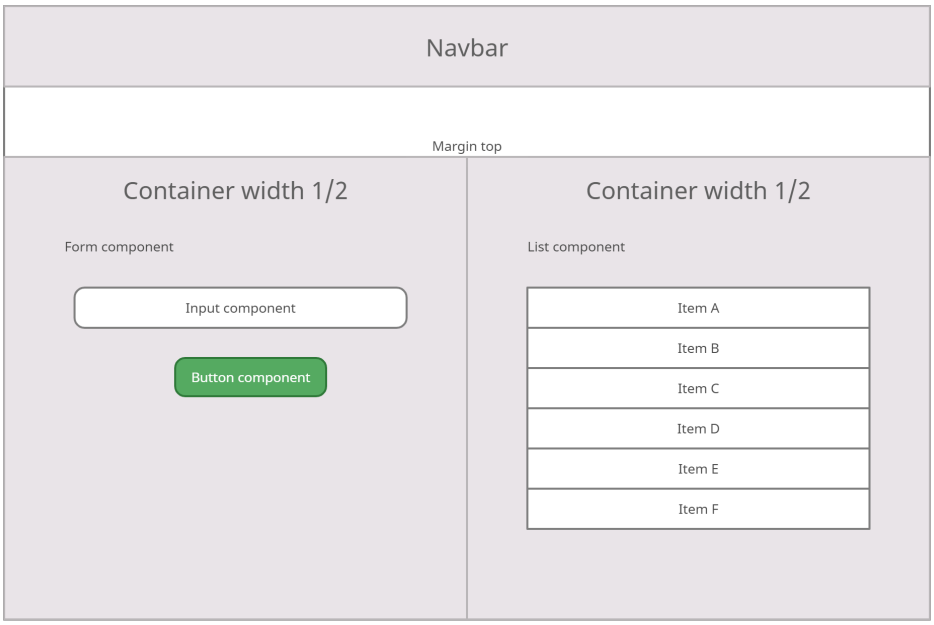
D.1.1 Component-based frameworky

Nabízí nástroje k tvorbě gridu, CSS utility (margin, padding, typografie, barvy aj.) a následně často používané komponenty (např. tlačítka, dialogová okna, formuláře). Tímto způsobem lze velice snadno a rychle tvořit komplexní a responzivní stránky. Rovněž je vývojář značně odstíněn od elementárního CSS jazyka a psaní selektorů.

+ Jednoduché
+ Rychlé
+ Rozšířené
– Obtížné přizpůsobení vzhledu
– Tvorba chybějících komponent

D.1.2 Utility-first frameworky

Abstrahují použití tradičního CSS za pomocí připravených HTML tříd. Jedna třída pak obvykle odpovídá jedné CSS vlastnosti. Díky tomu odpadá potřeba psaní selektorů a separátního CSS souboru.



Obrázek D.1: Vizualizace principu component-based CSS frameworku

\+ Vysoká flexibilita

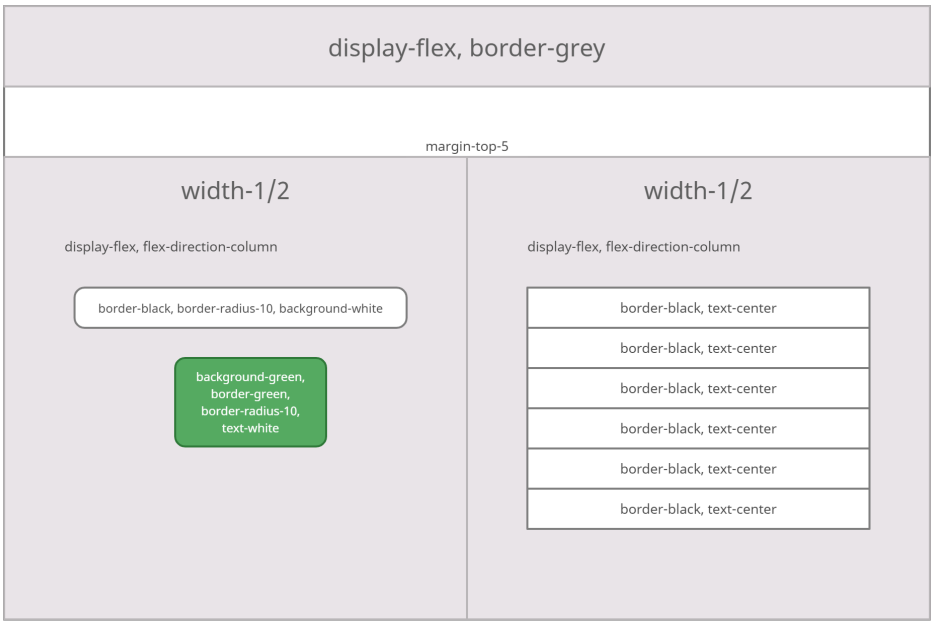
\+ Absence selektorů

– Nutný další mechanismus k tvorbě komponent

– Vyžaduje vyšší znalost CSS

D.2 Bootstrap

Bootstrap je nejrozšířenějším a nejznámějším CSS frameworkem. První verze byla vyvinuta společností Twitter v roce 2011 a v současné době se jde o komunitou řízený projekt již v 5. velké verzi. [39] Jedná se o component-based framework. Díky své strmé učící křivce a schopnosti snadno a rychle tvořit estetické a responzivní webové stránky jeho popularita nepřekvapí. Dlužno však říci, že zmíněná popularita měla za následek to, že mnoho stránek začalo vypadat velmi podobně. Bootstrap může být použit i pro tvorbu více přizpůsobených stránek, tím se však již vývojář vzdává oné jednoduchosti použití.



Obrázek D.2: Vizualizace principu utility-first CSS frameworku

D.3 Tailwind CSS

Pomyslným novým hráčem je framewok Tailwind CSS, jehož první verze byla vydána teprve v roce 2017. [38] Jedná se utility-first knihovnu, jejíž popularita pomalu roste. Next.js ji přímo referencuje jako doporučený přístup k tvorbě UI.

D.4 Shrnutí

Jednalo se o poměrně náročné rozhodnutí. S Bootstrapem mám mnohaleté zkušenosti a byl jsem obeznámen se všemi jeho úskalími. Nakonec jsem však dospěl k názoru, že vzhledem k tomu, že aplikace má komplikovanější UI s množstvím specifických komponent, nebude Bootstrap správnou volbou.

Zároveň jsem zaznamenal vzrůstající popularitu utility-first frameworků, kdy jsou hojně využívány například společností Gitlab a chtěl jsem tento přístup otestovat.

Z těchto důvodů jsem se rozhodl pro použití frameworku Tailwind CSS.

Příloha E

Uživatelský scénář

Uživatel vytvoří novou větev a provede v ní změnu přidáním českého synonyma k existujícímu konceptu. Následně provede sloučení jeho větve do větve hlavní, vizuální kontrolu a odstranění změny v hlavní větvi.

1. V menu klikněte na navigační tlačítko `Branches`
2. V sekci `Create branch` vyplníme tesovací data:
   - Název větve - např. MY-BRANCH
   - Nadřazená větev - zvolíme MAIN
3. Klikněte na tlačítko `Create`
4. V prohlížeči větví na levé straně pomocí tlačítka ve znaku šipky expandujte větev `MAIN`
5. Zvolte Vámi vytvořenou větev pomocí tlačítka s ikonou oka
6. Měla by se objevit notifikace o změně vybrané větce
7. V menu klikněte na navigační tlačítko `Search`
8. Do vyhledávacího řádku napište termín `headache` a stiskněte Enter
9. Ve výsledcích vyhledávání otevřeme kliknutím detail prvního konceptu `Headache (finding)`
10. Přejdeme do editačního režimu tlačítkem `Edit`
11. Do pole New synonym vyplníme hodnotu `Bolest hlavy`
12. Jazyk synonyma změníme z `English` na `Czech`
13. Vytvořte synonymum kliknutím na tlačítko `Add`

14. U nově vzniklého synonyma otevřeme nastavení `Acceptability` pomocí tlačítka s ikonou vlaječky a kliknutím na zelenou fajfku změníme synonymum na preferované pro český jazyk (obrázek visačky)
15. Potvrdíme změny pomocí tlačítka `Save` vpravo nahoře
16. V menu klikněte na navigační tlačítko `Changes`
17. Zkontrolujte, zda-li se v sekci `New synonyms` nachází `Bolest hlavy`
18. V menu klikněte na navigační tlačítko `Branches`
19. V prohlížeči větví na levé straně pomocí tlačítka s ikonou oka zvolte větev `MAIN`
20. V menu klikněte na navigační tlačítko `Changes`
21. Zkontrolujte, zda-li sekce `New synonyms` neexistuje/nenachází se v ní synonymum `Bolest hlavy`
22. V menu klikněte na navigační tlačítko `Branches`
23. Přejděte do průvodce sloučením větví tlačítkem `Launch merge wizard`
24. Nastavte:
    - Source branch: Vámi vytvořená větev
    - Target branch: MAIN
25. Stiskněte tlačítko `Merge`
26. Vyplňte `Commit comment` a stiskněte tlačítko `Confirm`
27. Dojde ke sloučení větví
28. V menu klikněte na navigační tlačítko `Changes`
29. Zkontrolujte, zda-li se v sekci `New synonyms` nachází `Bolest hlavy`
30. Otevřete detail konceptu `Headache (finding)` kliknutím na synonymum `Bolest hlavy`
31. Přejdeme do editačního režimu tlačítkem `Edit`
32. Odstraňte synonymum `Bolest hlavy` tlačítkem s ikonou koše
33. Potvrdíme změny pomocí tlačítka `Save` vpravo nahoře
34. V menu klikněte na navigační tlačítko `Changes`
35. Zkontrolujte, zda-li sekce `New synonyms` neexistuje/nenachází se v ní synonymum `Bolest hlavy`