Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Prostředky sémantického webu v uživatelském rozhraní pro správu elektrofyziologických experimentů

Plzeň 2020 Jan Palcút

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 9. srpna 2020

Jan Palcút

Poděkování

Tímto bych chtěl poděkovat vedoucímu diplomové práce panu Ing. Romanovi Moučkovi, Ph.D. za cenné rady, připomínky a odborné vedení této práce.

Abstract

The work aims to verify the applicability of technologies and languages of the semantic web in creating an application for the management of electrophysiological experiments. The application will allow us to manage multiple experiments of the same characteristic type at once, display their metadata, and search in them. The theoretical part of the work describes the semantic web, selected data standards for electrophysiology, and the neuroinformatics laboratory of the University of West Bohemia in Pilsen. The analysis and design of the application describe requirements' specifications and select the data standard and technologies to implement the solution. Part of the proposal is also a description of the solution of writing metadata of experiments into the newly designed structure. Based on the design, the application for the selected data standard is implemented. The functionality of the application was verified on experiments of various types.

Abstrakt

Cílem této práce je ověření použitelnosti technologií a jazyků sémantického webu při tvorbě aplikace pro správu elektrofyziologických experimentů. Aplikace umožní spravovat více experimentů stejného charakteristického typu najednou, zobrazovat jejich metadata a vyhledávat v nich. Teoretická část práce popisuje sémantický web, vybrané datové standardy pro elektrofyziologii a neuroinformatickou laboratoř Západočeské univerzity v Plzni. V analýze a návrhu aplikace je popsána specifikace požadavků, proveden výběr datového standardu a technologií k provedení řešení. Součástí návrhu je taktéž popis řešení zapsání metadat experimentů do nově navržené struktury. Na základě návrhu dochází k implementaci aplikace pro vybraný datový standard. Ověření funkčnosti aplikace proběhlo na experimentech různého typu.

Obsah

1 Úvod

Cílem této práce je ověřit použitelnost technologií a jazyků sémantického webu při tvorbě aplikace pro správu elektrofyziologických experimentů. Vytvořená aplikace umožní spravovat více experimentů stejného typu najednou. Jednotlivé nahrané experimenty půjdou stáhnout, smazat nebo rozšířit je o další. Obsah metadat těchto nahraných experimentů půjde převést do nově navržené struktury. Zapsaná metadata v této struktuře bude umožněno pro každý převedený experiment zobrazit a také v nich půjde vyhledávat.

Druhá kapitola této práce se bude zabývat sémantickým webem. Budou zde popsány jeho principy a účel jeho využití. Jednotlivé technologie a jazyky sémantického webu budou popsány podrobněji. Na základě této kapitoly poté dojde při implementaci aplikace k použití vybraných technologií a jazyků z nich.

Třetí kapitola bude obsahovat popis existujících datových standardů pro elektrofyziologii. V úvodu této kapitoly bude popsán důvod vzniku těchto standardů a benefity jejich využití. Jednotlivé datové standardy budou poté popsány. Budou zde popsány jejich přístupy řešení k ukládání objemných dat a metadat experimentů. U těchto standardů budou také popsány jejich nástroje a podpora programovacích jazyků.

Následující kapitola bude popisovat neuroinformatickou laboratoř Fakulty aplikovaných věd, která je součástí Západočeské univerzity v Plzni. Bude zde popsáno vybavení této laboratoře, portál EEGBase, infrastruktura tohoto portálu nebo provedené experimenty touto laboratoří.

Pátá kapitola se bude zabývat analýzou a návrhem aplikace. Na začátku ní budou popsány specifikace požadavků aplikace. Následně poté bude proveden výběr datového standardu a hlavního frameworku pro použití. Součástí kapitoly bude i řešení převodu metadat experimentů, výběr vhodných knihoven, technologií a jazyků nebo popis rozvržení aplikace.

Šestá kapitola se bude zabývat implementací. Bude zde popsána adresářová struktura projektu a jednotlivé její soubory. V této kapitole budou také zobrazeny vybrané části kódů vytvořené aplikace. Následující část textu se bude věnovat ověření výsledků. Aplikace bude testována na více experimentech různého typu. Pro každý typ bude v aplikaci vytvořen experiment a na správě jeho nahraných experimentů budou testovány jednotlivé funkcionality aplikace. Součástí bude i testování správy většího množství nahraných experimentů najednou. Podrobnější způsoby testování budou v této kapitole.

2 Sémantický web

Webové stránky v dnešní době obsahují velké množství informací a dat vytvořených různými organizacemi, komunitami a jednotlivci. Uživatelé webových stránek mohou jednoduše přistoupit k informacím pomocí URI[1] adresy, vyhledáváním nebo použitím odkazů k nalezení souvisejících zdrojů. Uniform Resource Identifier adresa je textový řetězec, jehož účelem je jednoznačná identifikace daného dokumentu.

Současná situace celosvětové sítě pro prohlížení, ukládání a odkazování dokumentů má některé nedostatky. Je velice jednoduché se ztratit, nalézt irelevantní nebo nesouvisející informace. Důvodem může být například chyba při automatizovaném zpracování dokumentu strojem (robotem) nebo zaměnění významu slov při vyhledávání. Jedno slovo totiž může mít více významů nebo může v jiném jazyce představovat úplně něco jiného, a proto výsledek vyhledávání záleží především na kvalitně napsaném dotazu, který bude sestaven z vhodných klíčových slov.

Cílem sémantického webu je vyvinout podpůrné standardy a technologie navržené tak, aby pomohly strojům lépe porozumět informacím na webu. To umožní zlepšení vyhledávání, integrace služeb a snadnější výměnu dat. S využitím sémantického webu získáme nejen přesnější výsledky při vyhledávání informací, ale také víme, kdy můžeme integrovat informace z různých zdrojů a jaké informace porovnávat.

Sémantický web umožňuje přidat popisné informace k jakémukoliv zdroji. Například přidáním metadat o kategorii a tvůrci dokumentu můžeme následně vyhledávat dokumenty, které vytvořil specifikovaný tvůrce nebo spadají do zvolené kategorie. Toto umožňuje sémantický web, protože neposkytuje dokumenty pouze jako URI, ale obsahuje i informace přidané v podobě metadat [21]. Definované principy sémantického webu jsou [21]:

1. Vše může být identifikováno pomocí URI - Sémantický web umožňuje odkazovat na lidi, místa a další objekty ve fyzickém světě pomocí různých identifikátorů. Například město Helsinky lze identifikovat pomocí URI odkazující na informace o tomto městě.

2. Zdroje a odkazy mohou mít typy - Web jako takový se v dnešní době skládá ze zdrojů a odkazů. Dochází zde k absenci metadat, které popisují účel a vztahy k jiným dokumentům. Tento problém řeší sémantický

[1] Uniform Resource Identifier

web doplněním typu, který umožní popsat vztahy mezi jednotlivými odkazovanými dokumenty.

3. Částečné informace jsou tolerovány - Zdroje mohou obsahovat různé typy odkazů mezi sebou. Časem může dojít k přerušení těchto propojení mezi zdroji. Sémantický web tento výpadek toleruje, a přesto zobrazí alespoň dostupná data.

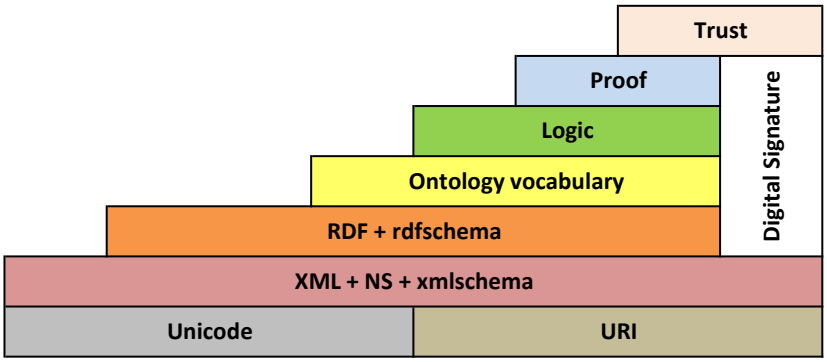
4. Absolutní pravda není nutná - Ani v případě sémantického webu nelze na všechny informace pohlížet jako na pravdivé. Jednotlivé aplikace jsou zodpovědné za zpracování informací. Pravdivost informace je vyhodnocena na základě kontextu aplikace.

5. Evoluce je podporována - Podobné koncepty jsou často definovány různými skupinami lidí nebo dokonce stejnou skupinou lidí v různých časech. Sémantický web umožňuje kombinovat data na webu z těchto konceptů s možností jejich úpravy.

6. Minimalistický design - Sémantický web zjednodušuje jednoduché věci a umožňuje složité věci. Cílem aktivity W3C není standardizovat víc, než je nezbytně nutné.

2.1 Vrstvy sémantického webu

Principy sémantického webu jsou implementovány ve vrstvách webových technologií a standardů. Vrstvy jsou zobrazeny na obrázku 2.1. Nejnižší vrstvy `Unicode` a `URI` zajišťují použití mezinárodní znakové sady a poskytují prostředky pro identifikaci objektů v sémantickém webu.



Obrázek 2.1: Vrstvy sémantického webu [21]

Vrstva XML[2] s definicemi jmenného prostoru a XML schématu slouží pro umožnění integrace s ostatními standardy, které jsou založené na XML. Následující RDF[3] vrstva obsahující RDF schéma umožňuje vytvářet tvrzení o objektech pomocí URI a definovat slovníky, na které se lze pomocí URI odkazovat. U této vrstvy můžeme definovat typy u zdrojů a odkazů. `Ontology vocabulary` vrstva podporuje vývoj slovníků a umožňuje definovat vztahy mezi rozdílnými koncepty. Následující vrstva `Logic` slouží k definování pravidel pro ověření správnosti [21].

Druhá nejvyšší vrstva je `Proof`. Tato vrstva zahrnuje dedukční proces, reprezentaci důkazů a ověření správnosti. To umožňuje aplikacím získat důkaz o rozhodnutí na základě vyhodnoceného závěru. Nejvyšší vrstva `Trust` poskytuje autentizaci identity a prokázání důvěryhodnosti dat a služeb. Tohoto je zajištěno prostřednictvím použití digitálního podpisu (představuje vrstvu `Digital Signature`), důvěryhodných agentů a hodnocením certifikačních agentur [7].

2.2 Resource Description Framework

RDF je standardní model pro výměnu dat a reprezentaci informací na webu. Poskytuje funkce, které usnadňují slučování dat, i když se základní schémata liší. Také podporuje vývoj schémat v průběhu času, aniž by bylo nutné kvůli tomu měnit veškeré nástroje a aplikace, které tato data využívají.

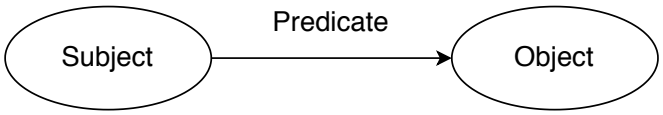
Tento framework rozšiřuje odkazování pomocí URI. To umožňuje pojmenovat vztah mezi věcmi a také mezi dvěma konci odkazu (obvykle se to označuje jako trojice). Pomocí tohoto jednoduchého modelu lze kombinovat, vizualizovat a sdílet strukturovaná a částečně strukturovaná data mezi různými aplikacemi.

Způsob odkazování u RDF tvoří orientovaný, označený graf, kde hrany představují vlastnosti mezi dvěma zdroji, které jsou reprezentovány pomocí uzlů. RDF graf tudíž představuje srozumitelné vysvětlení vztahů a vlastností mezi zdroji [36].

RDF graf je tvořen souborem trojic. Každá trojice se skládá ze subjektu (zdroje), predikátu (vlastnosti) a objektu (hodnoty vlastnosti). Reprezentace jedné trojice v RDF grafu je na obrázku 2.2. Zdroj může být prázdný uzel nebo být identifikován pomocí URI. Vlastnost je vždy identifikována pomocí URI. Objekt může mít identifikaci pomocí URI, textovou hodnotu nebo být prázdný uzel. U objektu lze pomocí URI definovat i datový typ, kterým může

[2]Extensible Markup Language
[3]Resource Description Framework

být například řetězec, číslo nebo datum [20]. Možné datové typy obsahuje odkaz [20].



Obrázek 2.2: RDF graf - trojice [20]

Framework poskytuje řadu syntaxí, které jsou k dispozici. Lze převést RDF graf nebo RDF datovou sadu do konkrétní syntaxe. Jednotlivý výběr syntaxe záleží na účelu použití nebo na dané aplikaci. K dispozici je například syntaxe RDF/XML, RDFa, Turtle, JSON-LD, a TriX. Zdroj [20] poskytuje odkazy na podrobnější popis těchto syntaxí.

```
<?xml version="1.0"?>

<RDF>
  <Description about="https://www.w3schools.com/rdf">
    <author>Jan Egil Refsnes</author>
    <homepage>https://www.w3schools.com</homepage>
  </Description>
</RDF>
```

Výpis 2.1: Ukázka RDF/XML syntaxe [43]

Výpis 2.1 představuje jednoduchou ukázku syntaxe RDF/XML. Tato data byla převzána ze zdroje [43] a obsahují dvě tvrzení. Prvním tvrzením je, že autorem stránky https://www.w3schools.com/rdf je Jan Egil Refsnes. Popis tohoto tvrzení trojicí je:

- Subjekt - https://www.w3schools.com/rdf
- Predikát - author
- Objekt - Jan Egil Refsne

Druhým tvrzením je, že hlavní stránka https://www.w3schools.com/rdf odkazu je https://www.w3schools.com. Vyjádření pomocí trojice je:

- Subjekt - https://www.w3schools.com/rdf

- Predikát - homepage
- Objekt - https://www.w3schools.com

2.3 Web Ontology Language a ontologie

OWL[4] je navržen pro použití v aplikacích, které potřebují zpracovávat obsah informací, nikoliv pouze prezentovat informace lidem. OWL umožňuje definovat význam termínů ve slovnících a také vyjádřit vztahy mezi těmito termíny. Slouží tedy k reprezentaci znalostí o objektech a jejich vztazích. Reprezentace těchto pojmů a vztahů mezi nimi se nazývá ontologie [23]. Vytvořená OWL ontologie tudíž může obsahovat popisy tříd, vlastností a jejich instancí [38].

Ontologii lze rozšířit o další znalosti o objektech a jejich vztazích pomocí importu. Při importu jedné ontologie do druhé se přenesou všechna její tvrzení. Zda-li importovaná ontologie importovala jinou, tak i její tvrzení budou přenesena. Lze importovat jakoukoliv dostupnou ontologii napříč celým internetem. Ovšem je potřeba si uvědomit, že zdroj nemusí být vždy dostupný a je potřeba to brát v potaz [38].

OWL poskytuje tři jazyky pro využití. Koncový uživatel si vybírá jazyk na základě jeho potřeby. Jednotlivé jazyky poskytují rozdílnou míru vyjádření, omezení a složitost zpracování. K dispozici jsou jazyky [23]:

- OWL Lite - Verze jazyka OWL, která má nejmenší vyjadřovací schopnost. Oproti ostatním jazykům má výhodu, že je jednodušší na zpracování. Tento jazyk například podporuje omezenou verzi kardinality, kde její hodnota může být pouze nula nebo jedna.
- OWL DL - Má maximální vyjadřovací sílu včetně zaručení výpočetní úplnosti a dokončení výpočtů v konečném čase. OWL DL zahrnuje všechny konstrukce jazyka OWL, ale pouze v omezené podobě. Příkladem omezení je třída, která může být podtřídou více tříd, ale nemůže být instancí jiné třídy.
- OWL Full - Podporuje všechny konstrukce OWL bez omezení. To nám zaručuje maximální vyjadřovací schopnost. Oproti ostatním jazykům má tento vysokou složitost zpracování bez zaručení výpočtu.

Aktuální verzí OWL je verze 2. Ta je zpětně kompatibilní s původní verzí. Novější verze oproti původní poskytuje například syntaxi (Manchester Syntax), rozšiřující datové typy, přidává asymetrické a vylučující se vlastnosti

[4]Web Ontology Language

a rozšiřuje možnosti anotace. Pro ukládání ontologií a jejich výměnu mezi aplikacemi a nástroji poskytuje OWL 2 řadu syntaxí. Tabulka 2.1 popisuje tyto syntaxe [31].

Název syntaxe	**Účel**
RDF/XML	Slouží pro výměnu dat (syntaxe je podporována všemi kompatibilními nástroji s OWL 2)
OWL/XML	Jednodušší pro zpracování XML nástroji
Functional Syntax	Poskytuje přehlednou strukturu ontologií
Manchester Syntax	Umožňuje čtení a zápis DL ontologií
Turtle	Pro čtení a zápis RDF trojic

Tabulka 2.1: Syntaxe OWL 2 [31]

2.4 RDF Schema

RDF Schema (RDFS) rozšiřuje RDF o možnost vytvářet vlastní slovníky a ontologie. Poskytuje mechanismy pro popis souvisejících zdrojů a vztahů mezi těmito zdroji. Slouží u zdrojů (tříd) k určení charakteristik, jako jsou domény a rozsahy vlastností.

Definice tříd a jejich vlastností u RDF Schema je podobná jako u objektově orientovaných programovacích jazyků (jako je například Java). Ovšem RDF Schema namísto definice třídy z hlediska vlastností, které mohou mít její instance, se liší tím, že popisuje vlastnosti z hlediska tříd zdrojů, na které se vztahují. Například bychom mohli definovat vlastnost `autor` tak, aby měla doménu `Dokument` (přiřadí vlastnost této třídě) a rozsah `Osoba` (hodnoty pro tuto vlastnost jsou instancí této třídy). Zatímco objektově orientovaný jazyk by mohl definovat třídu s název `Dokument` s atributem `autor`, který by byl typu `Osoba`.

Zdroje lze rozdělit do skupin, kde každá skupina reprezentuje příslušnou třídu. Členové třídy jsou označeny jako její instance. Třídy jsou sami o sobě označeny jako prostředky a mohou být identifikovány pomocí URI a popsány pomocí vlastností RDF. Vlastnost `rdf:type` se používá k označení, které říká, že prostředek je instancí dané třídy.

RDF rozlišuje mezi třídou a množinou jejích instancí. Každá třída má vlastní množinu instancí. Dvě třídy mohou mít stejnou sadu instancí, ale i přesto mohou být třídy různé. Třída může být členem vlastního rozšíření třídy a instancí sama sobě. Skupina prostředků, které jsou třídami RDFS, je sama o sobě instancí `rdfs:Class`. Pro vyjádření, že třída je podtřídou

jiné třídy se využívá vlastnost `rdfs:subClassOf`. K dispozici je větší množství vlastností. Lze například říci, že vlastnost je podmnožinou jiné pomocí `rdfs:subPropertyOf` [6]. Zdroj [6] podrobněji popisuje dostupné vlastnosti.

2.5 Dotazovací jazyk SPARQL

SPARQL je dotazovací jazyk, který se používá pro práci s daty, která jsou uložena ve formátu RDF. Dotazy tohoto jazyka obsahují sadu trojic vzoru, která je nazvána jako tzv. graf vzoru. Trojice vzoru se oproti RDF trojici liší tím, že každý subjekt, predikát nebo objekt může být proměnná.

Tento jazyk poskytuje základní čtyři typy dotazů. Jsou jimi SELECT, ASK, CONSTRUCT a DESCRIBE. První z nich je nejpoužívanější a slouží pro nalezení dat (množiny n-tic) dle zadaných podmínek. Funkčně odpovídá typu stejně pojmenovaného dotazu z jazyka SQL.

Dále bude dotazovací jazyk SPARQL představen pomocí příkladů, které popisují způsoby jeho využití a pomohou mu lépe porozumět. Výpis 2.2 obsahuje vzorová data, na kterých budou demonstrovány jednoduché příklady. Tato data jsou ve formátu Turtle. Obsahují prefix `foaf:`, který je použit pro jmenný prostor `<http://xmlns.com/foaf/0.1/>` a sadu trojic. U subjektů si lze všimnout značení "_:", které představuje prázdný uzel.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Johnny Outlaw" .
_:a  foaf:email   "outlaw@example.com" .
_:b  foaf:name   "Johnny Outlaw" .
_:b  foaf:email   "johnny@example.com" .
_:c  foaf:name   "Peter Goodguy" .
_:c  foaf:email   "goodguy@example.org" .
_:d  foaf:email   "carol@example.org" .
```

Výpis 2.2: Vzorová data ve formátu Turtle

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?email
WHERE
  { ?x foaf:name ?name .
    ?x foaf:email ?email }
```

Výpis 2.3: SPARQL dotaz typu SELECT

Výpis 2.3 slouží k nalezení všech jmen osob a jejich emailů. Tento dotaz se skládá ze dvou částí. První částí je klauzule SELECT, která identifikuje proměnné, které se mají objevit ve výsledcích. Druhou částí je klauzule WHERE, která poskytuje vzor grafu. Ten se skládá z dvou trojic vzoru, kde první má na pozici objekt proměnnou `?name` a druhá `?email`.

Tabulka 2.2 zobrazuje výsledky získané z dotazu, který je vidět na výpisu 2.3. Lze si všimnout, že výpis neobsahuje osobu, která má subjekt `_:d`. Je to z důvodu toho, že nemá predikát `foaf:name`.

name	**email**
"Johnny Outlaw"	"outlaw@example.com"
"Johnny Outlaw"	"johnny@example.com"
"Peter Goodguy"	"goodguy@example.org"

Tabulka 2.2: Výsledek dotazu pro výpis 2.3

V případě, že bychom chtěli vyřešit problém, že nám dotaz z výpisu 2.3 nezobrazil ve výsledku osobu `d`, lze tento dotaz modifikovat pomocí vnořené klauzule OPTIONAL. Výpis 2.4 demonstruje tuto modifikaci.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?email
WHERE
  { ?x foaf:email ?email .
    OPTIONAL { ?x foaf:name ?name }
  }
```

Výpis 2.4: SPARQL dotaz s klauzulí OPTIONAL

Tabulka 2.3 zobrazuje výsledky pro dotaz z výpisu 2.4. Byla zde použita data z výpisu 2.2. Jsou zde zobrazeny čtyři záznamy osob.

name	**email**
"Johnny Outlaw"	"outlaw@example.com"
"Johnny Outlaw"	"johnny@example.com"
"Peter Goodguy"	"goodguy@example.org"
	"carol@example.org"

Tabulka 2.3: Výsledek dotazu pro výpis 2.4

Další uvedený dotaz představuje výpis 2.5. Jeho využití je pro nalezení jedinečných jmen osob.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?name
WHERE { ?x foaf:name ?name }
```

Výpis 2.5: SPARQL dotaz s příkazem DISTINCT

Tabulka 2.4 obsahuje výsledky pro dotaz z výpisu 2.5. Pro tento dotaz byla použita data z výpisu 2.2. Ve výsledcích jsou zobrazeny dva záznamy s jedním sloupcem (name).

name
"Johnny Outlaw"
"Peter Goodguy"

Tabulka 2.4: Výsledek dotazu pro výpis 2.5

Výpis 2.6 slouží jako dotaz, který lze použít například pro stránkování. Příkaz ORDER BY v tomto případě seřadí osoby abecedně podle jejich názvu vzestupně. Pro seřazení sestupně se používá ORDER BY DESC. LIMIT zde omezí počet záznamů, které se mají zobrazit. OFFSET slouží pro počet přeskočení záznamů. Když bychom měli například 20 původních záznamů, tak se nám dle tohoto dotazu seřadí záznamy abecedně a poté se zobrazí ve výsledku záznam 11 až 15.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT   ?name
WHERE    { ?x foaf:name ?name }
ORDER BY ?name
LIMIT    5
OFFSET   10
```

Výpis 2.6: SPARQL dotaz s příkazem ORDER BY, LIMIT a OFFSET

V klauzuli WHERE je možné použít i příkaz FILTER, ve kterém lze například omezit výběr pomocí menšítka, většítka nebo rovnítka. Taktéž lze v tomto případě využít i regulárního výrazu. Výpis 2.7 nalezne všechny osoby, u kterých začíná jméno řetězcem "Johnny".

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT   ?name
WHERE    { ?x foaf:name ?name
           FILTER regex(?name, "^Johnny")
         }
```

Výpis 2.7: SPARQL dotaz s příkazem FILTER

Tabulka 2.5 obsahuje výsledky pro dotaz z výpisu 2.7. Pro případ použití vzorových dat byly získány dva záznamy s jedním sloupcem (name). Oba záznamy mají stejnou hodnotu ("Johnny Outlaw").

name
"Johnny Outlaw"
"Johnny Outlaw"

Tabulka 2.5: Výsledek dotazu pro výpis 2.7

Následující výpis 2.8 představuje dotaz typu ASK, který určí, jestli se zadaný vzor grafu v datech nalézá, nebo nikoliv. Výsledkem tohoto dotazu je tedy logická hodnota, která může být `true` (nalezen vzor grafu), nebo v opačném případě `false`.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

ASK  { ?x foaf:name  "Alice" }
```

Výpis 2.8: SPARQL dotaz typu ASK

Zdroj [33] popisuje podrobněji dotazovací jazyk SPARQL. Je zde uvedeno větší množství příkladů a jejich uplatnění.

3 Elektrofyziologické datové standardy

Druhým bodem zadání je seznámit se s existujícími datovými standardy popisující elektrofyziologická data. Bude jim věnována celá tato kapitola. Jeden z těchto datových standardů bude později vybrán a využit při návrhu a implementaci aplikace.

Elektrofyziologie na rozdíl od jiných oborů (tj. genetika a buněčná biologie) neměla standardizovaný způsob ukládání dat a sdílení existujících dat mezi vědci. Absence společného formátu ztěžovala možnost porovnávání výzkumů napříč různými laboratořemi a přinášela problém replikace konkrétních experimentů. Celkově to zpomalovalo pokrok výzkumu v této oblasti. Z tohoto důvodu došlo k vzniku těchto datových standardů.

Cílem těchto standardů je standardizovat elektrofyziologická data v mezinárodním měřítku. Odstraňují geografické, institucní, technologické a politické bariéry, které brání toku dat v rámci vědecké komunity. Jejich záměrem je zvýšit kvalitu výzkumů a objevování v oblasti elektrofyziologie [3].

Tyto datové standardy poskytují podporu pro ukládání komplexních i objemných dat (včetně jejich metadat) a usnadňují vyhledávání v nich. Pro čtení a zápis dat vznikla různá řada aplikačních rozhraní, která podporují programovací jazyky jako Python, Matlab, Java a další (závisí na konkrétním standardu). Také vznikla spousta nástrojů, které usnadňují práci s experimenty. Jejich využití je například pro vizualizaci nebo analýzu dat.

3.1 NIX

Předtím, než-li dojde k popisu tohoto datového standardu, je potřeba prvně vysvětlit pojem HDF5, o kterém se bude v této sekci mluvit. HDF5 je datový model, knihovna a formát souborů pro ukládání a správu dat. Podporuje neomezené množství datových typů. Je navržen pro efektivní ukládání a načítání objemných a komplexních dat. Výhodou HDF5 je jednoduchá možnost jeho rozšířitelnosti a přenositelnost na různé platformy. Technologická sada HDF5 zahrnuje nástroje a aplikace pro správu, manipulaci, prohlížení a analýzu dat ve formátu HDF5 [19].

Projekt Neuroscience Information Exchange Format (NIX) byl v dřívější době označován pod názvem Pandora. Cílem tohoto projektu je vyvinout

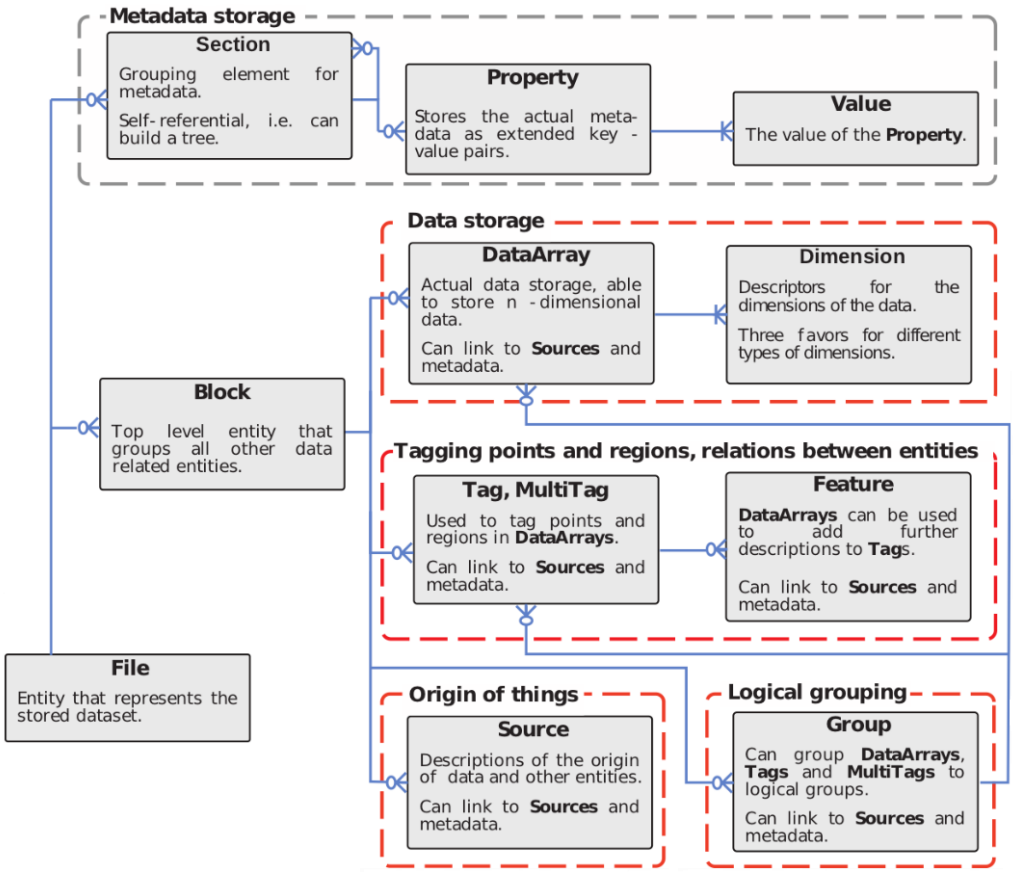
standardizované metody a modely pro ukládání elektrofyziologických a jiných neurovědních dat spolu s jejich metadaty v jednotném společném formátu založeném na HDF5. Aby toho bylo dosaženo, používá NIX vysoce obecné modely pro data i pro metadata a definuje standardní schémata pro HDF5 soubory, které mohou reprezentovat tyto modely [14].

Tento datový standard je podporován různými programovacími jazyky. Mezi ně patří především C++, Python, Matlab a Java. K dispozici jsou dvě API, které slouží pro čtení a zápis. První z nich je v jazyce C++ a druhé v jazyce Python. Součástí standardu je nástroj NixView, který slouží jako prohlížeč souborů NIX. Umožňuje zobrazit data, metadata i značky [18].

3.1.1 Datový model standardu

Principem návrhu datového modelu tohoto standardu bylo vytvořit minimalistický a obecný model, který bude mít dobré vlastnosti vyjádření a bude schopen reprezentovat data uložená v jiných běžně používaných formátech nebo modelech, jako je Neuroshare a NEO, bez ztráty informací. Jmenované formáty Neuroshare a NEO mají využití taktéž především pro elektrofyziologická data. Datový model NIX je schopen reprezentovat i například obrazová data nebo množinu obrázků. Toto je umožněno díky obecnosti návrhu tohoto modelu [15].

Obrázek 3.1 znázorňuje ER (entity-relationshop) diagram základních entit, které jsou definovány v datovém modelu NIX. Jsou zde vidět vazby mezi jednotlivými entity. U všech entit na obrázku je krátkým popiskem popsán jejich účel. Taktéž čárkované ohraničení popisuje jejich účel využití. Z vazeb mezi entitami lze vyčíst, zda-li cizí klíč může nabývat hodnoty NULL. Tato možnost je znázorněna prázdným kolečkem u vazeb mezi entitami.



Obrázek 3.1: ER diagram datového modelu NIX [28]

Datový model NIX se skládá ze šesti základních entit, kterými jsou [16]:

- Block - Je prvek seskupování na nejvyšší úrovni pro datové objekty, které spolu nějakým způsobem souvisí. Je vyžadováno, aby každý datový objekt byl spojen s jedním Block objektem. Block lze považovat za něco, co představuje související soubory informací nebo kombinované výsledky experimentu.

- DataArray - Tato entita je jádrem datového modelu. Jejím hlavním účelem je ukládat libovolná nezpracovaná data do n-dimenzionálního pole. Jednotlivé atributy slouží jako prostředky pro popis vlastností uložených dat. Obsahuje atributy pro datový typ, jednotky, popis jednotlivých os pro vykreslení dat a další.

- Tag - Entita Tag se používá k anotaci dat uložených v jednom nebo více objektech DataArray. Značka může být použita pro označení čehokoliv v množině dat. Lze pomocí ní označit například nějakou událost, ke

které došlo nebo reakci na určitý podnět. Z tohoto důvodu má značka výchozí pozici a rozsah.

- MultiTag - Jedná se o druhou možnost značkování pro anotaci dat. Ve srovnání s entitou Tag se tato používá k anotaci dat na více pozicích s možností různých rozsahů.

- Source - Tato entita popisuje původ objektu z DataArray nebo Tag. Tato data lze považovat za metadata. Hlavním důvodem zahrnutí této entity do datového modelu je zajištění kompatibility s jinými formáty, jako je NEO nebo Neuroshare, aniž by lidé byli nuceni používat odML[1].

- Group - Umožňuje vytváření podskupin pod objektem z entity Block. Skupina může být tvořena objekty z entit DataArray, Tag a MultiTag. Může být propojena se zdroji ze Source a mít připojená metadata.

Všechny entity v části modelu dat mají atribut id, name, definition, createdAt, updatedAt a type. Jedinou výjimkou je Dimension, který nemá name a type [16]. Význam těchto jednotlivých atributů je [16]:

- id - Unikátní řetězec, který se používá k identifikaci a odkazování na data a objekty metadat. Atribut id se skládá z předpony domény následované náhodným šedesátimístným hexadecimálním číslem. Mezi předponou a číslem je podtržítko. Tento způsob zajišťuje, aby nedocházelo ke kolizím ani ve velkých sbírkách dat z různých zdrojů. Jako předponu se doporučuje použít internetovou doménu.

- type - Definuje skutečnou povahu datového objektu. To umožňuje zavést specifičnost domény do obecného modelu.

- name - Uživatelem zadaná hodnota pro specifikaci objektu entity. Název nemusí být jedinečný. Ovšem doporučuje se užívat především jedinečné názvy, aby nedošlo k záměně.

- definition - Je volně přiřaditelná hodnota, která slouží k definování objektu entity.

- createdAt - Hodnotou tohoto atributu je datum vytvoření. Tato hodnota je zpracována automaticky a nelze ji ručně nastavit.

- updatedAt - Podobné jako u createdAt. Ovšem tady je hodnotou datum úpravy.

[1] open metadata Markup Language

Model NIX byl navržen tak, aby umožňoval přiřadit k datům metadata. Z tohoto účelu zahrnuje odML model pro metadata. V odML modelu jsou informace uloženy ve formě rozšířených dvojic klíč-hodnota. Tento vztah je v modelu znázorněn pomocí vazby mezi entitou Property a Value. Entita Property obsahuje platné informace pro všechny hodnoty vlastností v ní uložené a má atributy id, name, definition, unit, values a mapping [16, 17].

Součástí oblasti modelu pro anotování dat metadaty je i entita Section, která má atributy id, type, name, definition, repository, link, mapping, properties a sections. Lze si všimnout, že tato entita má vazbu sama se sebou. To nám umožňuje vytvářet stromovou strukturu metadat. Na metadata ze Section se mohou odkázat ostatní entity pomocí jejich atributu. Tento atribut mají entity Block, DataArray, Tag, MultiTag, Source a Group [16].

Model pro data a ani model pro metadata (odML) nejsou určeny pouze pro elektrofyziologii. Oba tyto modely mohou být spojené s předdefinovanými nebo vlastními terminologiemi, které uživateli umožňují přiřadit elementům modelů příslušnou doménu (sémantický kontext). Pro tento účel odML poskytuje seznam terminologií. Jejich poskytnutí usnadňuje a podporuje standardizace všech vědeckých metadat. Použití terminologií také zjednoduší práci, protože není potřeba poskytovat definice, jelikož už jsou jejich součástí [17, 29].

3.2 BIDS

Braing Imaging Data Structure (BIDS) byl původně navržen pro ukládání a zobrazování dat magnetické rezonance (MRI). V dnešní už tento standard podporuje ukládání dat pro různé modality záznamů a umožňuje sdílení jejich údajů mezi různými laboratořemi.

BIDS primárně řeší heterogenitu organizace údajů podle definovaných FAIR principů[2]. Těmi jsou vyhledatelnost, dostupnost, interoperabilita a znovupoužitelnost. Pojem interoperabilita lze chápat jako schopnost výměny a sdílení dat, i když jsou například odlišné softwarové platformy. Tento pojem zahrnuje i řešení datových struktur a rozhraní. BIDS řeší spolehlivost a znovupoužitelnost popisem dat metadaty. Pro ukládání metadat jsou vyhrazeny jednotlivé soubory v datové struktuře standardu. O těchto souborech bude více řečeno v sekci 3.2.1.

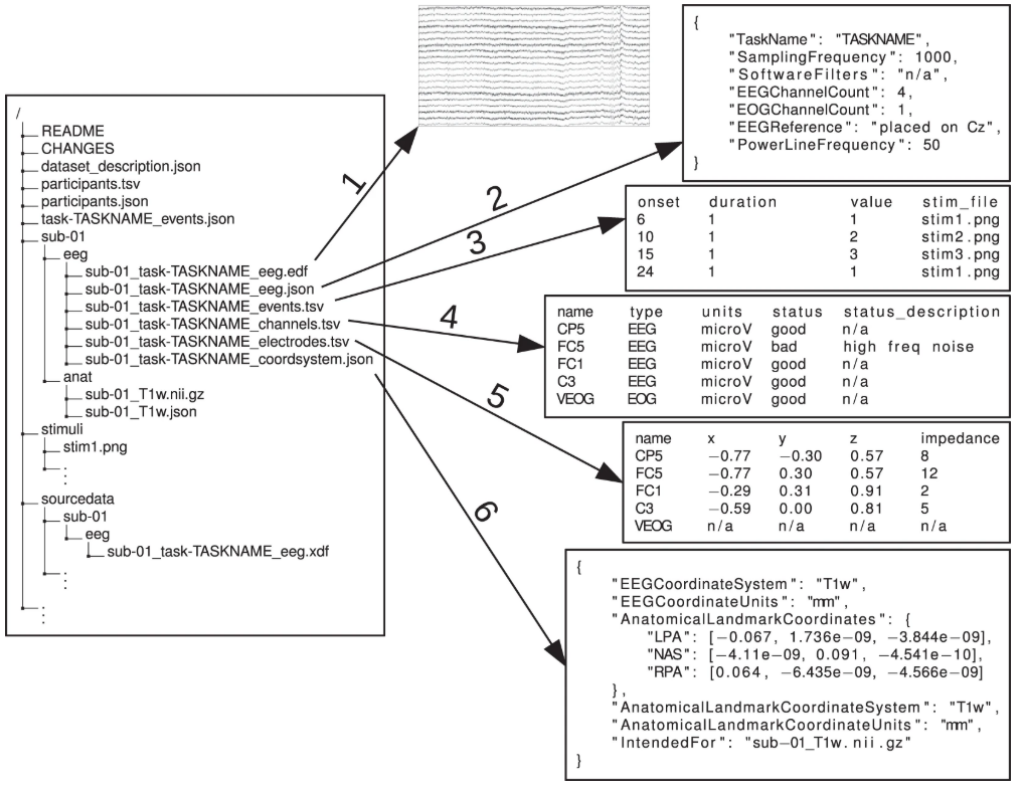
Princip interoperability je zde zajištěn pomocí existujících standardních datových formátů. Stanovená struktura dat, metadat a jmenné konvence podporují princip interoperability a znovupoužitelnost získaných datových

[2]https://www.go-fair.org/fair-principles/

souborů. BIDS, jako takový nezaručuje přístupnost. Ta je řešena pomocí úložišť, které jsou vytvořené na základě tohoto standardu. Příkladem uvedeného repozitáře je OpenNeuro [32].

3.2.1 Datová struktura standardu EEG-BIDS

Rozšíření BIDS pro EEG odpovídá obecné specifikaci (obrázek 3.2) tohoto standardu. Lze si všimnout, že jsou data uložena ve stromové struktuře. Ta pro každý subjekt obsahuje adresář, ve kterém jsou podadresáře pro každé měření experimentu a modalitu dat. Soubor `dataset_description.json` je v hlavním adresáři. Obsahuje obecné informace o použité datové sadě. Mezi volitelné složky, které lze přidat, patří `sourcedata`. Ten lze použít k dodání původních neformátovaných dat. Dále lze použít adresář `stimuli`, ve kterém můžou být uložené například obrázky nebo zvukové záznamy podnětů. Posledním volitelným adresářem je `code`. Ten obsahuje například prostředky pro předzpracování dat nebo pro opakování experimentu.



Obrázek 3.2: Adresářová struktura standardu BIDS [32]

V hlavním adresáři je soubor pro popis účastníků měření. Jeho název je `participants.tsv`. Dále jsou v tomto adresáři soubory `participants.json`

a `task-TASKNAME_events.json`, které obsahují nezbytně nutný popis k porozumění obsahu TSV souborů.

Pro EEG-BIDS každý adresář subjektu obsahuje podadresář `eeg`, ve kterém je uložen záznam EEG signálu a jeho metadata. Tento případ struktury adresářů, který je znázorněn i na obrázku 3.2, je jen pro jedno provedené měření experimentu subjektu. Kdyby jich bylo více, tak by byl adresář `eeg` vložen pod adresářem měření experimentu, který by byl pod subjektem. Následující popis bude uvažovat s tím, že každý subjekt má pouze jedno provedené měření. Popsané vzory budou obsahovat značení XX a YY, které slouží jako identifikátory pro označení. Adresář subjektu má vzor názvu `sub-XX`. Uvnitř složky `eeg` můžou být dva soubory, které mají název ve tvaru `sub-XX_task-YY_eeg.<extension>`. První z nich má koncovku edf a uvnitř něho je naměřený EEG záznam. Druhý soubor je ve formátu JSON a obsahuje metadata záznamu.

Adresář `eeg` dále obsahuje další čtyři soubory. První z nich má tvar názvu `sub-XX_task-YY_channels.tsv` a popisuje parametry sběru dat. Jsou zde vypsány jednotky elektrod a jejich status, který říká, jak dobře byla elektroda zapojena. Obrázek 3.2 obsahuje hodnoty statusu `good` nebo `bad`. Součástí tohoto souboru může být i popis nastavení filtru, který byl na data EEG záznamu aplikován. Následující dva soubory se zabývají zapojením elektrod. První z nich má název ve tvar `sub-XX_task-YY_electrodes.tsv` a specifikuje lokaci elektrod. U jednotlivých elektrod může být zobrazena i hodnota jejich odporu. Druhý má tvar `sub-XX_task-YY_coordsystem.json` a podrobněji popisuje rámec systému zapojení elektrod. Posledním souborem z těchto čtyř souborů je `sub-XX_task-YY_events.tsv`. Ten obsahuje všechny události, ke kterým v průběhu měření došlo. Součástí je doba nástupu události, typ pokusu, doba trvání, reakce na podnět a další. Jednotlivé záznamy v tomto souboru mohou odkazovat i na podněty, které byly použity v rámci měření (obrázek, zvuk a další) [32].

K dispozici je několik příkladů používající EEG-BIDS standard, které jsou k nalezení v repozitáři[3] GitHubu. Jsou také vydány tři úplné datové sady, kterými jsou [32]:

- The Matching Pennies dataset - Obsahuje data EEG záznamů. V adresářové struktuře je vždy právě jedno provedené měření na subjektu. Jsou to data, při kterých bylo prováděné zvedání pravé a levé ruky účastníků [5, 32].

- The Rishikesh dataset - Je příkladem více provedených měřeních na jednom subjektu. Taktéž obsahuje EEG data. Při nahrávání těchto

[3]https://github.com/bids-standard/bids-examples

dat účastníci meditovali a v náhodných intervalech jim byly pokládány otázky [32].

- The simultaneous resting-state EEG-fMRI - Obsahuje data záznamů EEG a fMRI (Funkční magnetická rezonance) klidového stavu [32].

3.2.2 Nástroje a podpora standardu

V rámci projektu BIDS lze datové sady, formátované podle standardu EEG-BIDS, ověřit pomocí aplikace bids-validator. Jedná se o JavaScriptovou aplikaci, kterou lze spustit v rámci příkazové řádky nebo využít webové aplikace[4]. Pomocí tohoto nástroje, pro ověření, lze zkontrolovat nově naformátované datové sady a plně využít silné stránky datové struktury. Dále lze zkontrolovat chybějící data nebo nedostatečně specifikovaná metadata.

Repozitář GitHub obsahuje stránku[5], která je určena jako startovací sada pro datový standard BIDS. Ta obsahuje kolekce komunitních průvodců, výukových programů, pomocných skriptů a zdrojů wiki, které pomáhají uživatelům začít s BIDS. Zdroje momentálně pokrývají dva programovací jazyky (Python a Matlab) a postupem času dochází k rozšíření množství návodů.

Součástí tohoto projektu je i spolupráce s vývojáři nejpoužívanějších nástrojů pro analýzu dat EEG. BIDS se touto spoluprací snaží usnadnit převod dat do dvou datových formátů, kterými jsou EDF a BrainVision Core Data Format. Obslužné programy pro převod dat z různých formátů prvotních dat EEG do dvou zmíněných datových formátů jsou k dispozici v Matlabu pomocí nástrojů FieldTrip a EEGLAB. V Pythonu lze tento převod provést pomocí MNE-Python balíčku. Kromě funkcionality převodu do jmenovaných datových formátů umožňují tyto nástroje i exportovat celé studie z těchto nástrojů do standardu BIDS. V EEGLAB je to pomocí souboru std_tobids.m a u FieldTrip to lze exportovat pomocí souboru data2bids.m. V Pythonu je tato možnost zajištěna pomocí už zmíněného balíčku MNE-Python. Tento balíček umí číst jakýkoliv datový soubor BIDS [32].

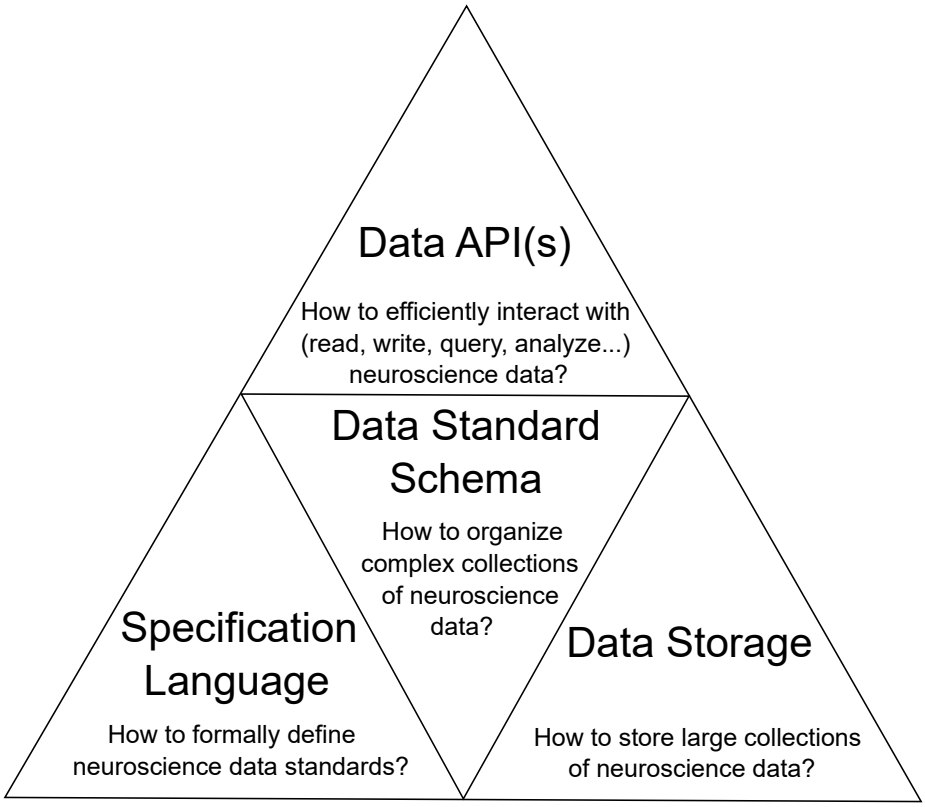
3.3 NWB:N

Tento standard poskytuje neurovědcům podporu pro sdílení, použití a archivaci dat. Taktéž podporuje vytváření analytických nástrojů pro tato data. Neurodata Without Borders: Neurophysiology (NWB:N) je navržen pro ukládání různých neurofyziologických dat, včetně dat z intracelulárních a extra-

[4]https://bids-standard.github.io/bids-validator/

[5]https://github.com/bids-standard/bids-starter-kit

celulárních elektrofyziologických experimentů, dat z optické fyziologie a stimulačních dat. Projekt zahrnuje nejen formát NWB, ale také širokou škálu softwaru pro standardizaci dat. Součástí jsou i rozhraní pro programování aplikací. Momentálně se tento standard nachází ve verzi dva, která byla vydána v únoru 2019 [40]. Obrázek 3.3 zobrazuje přehled hlavních komponent tohoto standardu, které řeší přístupnost dat, jejich organizaci, způsob uložení a další otázky týkající se standardizace neurovědních dat.



Obrázek 3.3: Hlavní komponenty standardu NWB:N [39]

3.3.1 Schéma datového standardu

Formát NWB je základní součástí tohoto standardu. Je navržen tak, aby ukládal data především z oblasti elektrofyziologie. Data jsou uložena ve struktuře, která je člověku srozumitelná a zároveň počítačově zpracovatelná. Formát je navržen tak, aby poskytoval podporu pro různé nástroje a byl jednoduchý na použití. NWB formát uspořádává data hierarchicky pomocí snadno použitelných primitiv. Těmi jsou:

- Group - Skupina může obsahovat libovolný počet dalších skupin a datových sad.

- Dataset - Slouží k popisu n-dimenzionálního pole a poskytuje primární prostředky pro ukládání dat.

- Attribute - Jedná se o kolekci méně rozsáhlých dat, která mohou být připojena ke skupině nebo k datové sadě. Obvykle se používá k ukládání metadat pro popis konkrétního objektu.

- Link - Toto primitivum umožňuje odkazovat na jinou skupinu nebo datovou sadu.

Formát NWB používá modulární návrh, ve kterém všechny hlavní sémantické komponenty formátu mají typ `neurodata_type`. Jedná se o princip podobný třídě v objektově orientovaném návrhu. To umožňuje opětovně znovupoužití a rozšíření typů pomocí začlenění a dědičnosti. Všechny datové sady a skupiny lze ve formátu jedinečně identifikovat pomocí jejich názvu nebo typu `neurodata_type`.

Dva důležité typy ve formátu NWB jsou `NWBContainer` a `TimeSeries`. První jmenovaný z nich definuje obecný kontejner pro ukládání shromažďovaných dat, a také lze pomocí něj formulovat různé funkcionality (i napříč kontejnery). `TimeSeries` je hlavní komponentou pro ukládání komplexních časových řad. Oba tyto jmenované typy jsou rozšířeny o další specializované typy. To je například z důvodu, aby bylo možné uspořádat kolekce zpracovaných dat pomocí kroků zpracování. Pro tento účel formát NWB definuje koncept ProcessingModule, kde je každý krok zpracování dat reprezentován odpovídajícím typem `NWBDataInterface` (jedná se o rozšíření `NWBContainer`). Data pak mohou být uspořádána pomocí modulů zpracování. Každý tento modul může mít jeden i více typů `NWBDataInterface` a má vlastní jméno (například segmentace obrazu), aby šlo pochopit význam modulu pro uložení dat [2].

Na vysoké úrovni tohoto formátu jsou data organizována do šesti hlavních skupin, kterými jsou [1, 2]:

- Acquisition - Jsou zde datové toky zaznamenané ze systému. Tato data mohou odkazovat na nezpracovaná data, která jsou uložena v externích souborech NWB. Po dokončení experimentu by data ve skupině měla sloužit pouze ke čtení.

- Intervals - Obsahuje intervaly experimentů. Lze pomocí nich označit jednotlivé fáze experimentů nebo provést dílčí označení v rámci záznamu. Taktéž tato skupina umožňuje označit i události, které mají stejné logické seskupení nebo intervaly, které by měly být odstraněny z analýzy dat.

- Stimulus - Jsou zde data podnětů pro stimulaci. Tato skupina může obsahovat například zvuk, video nebo hodnotu napětí.
- General - Zde jsou experimentální metadata včetně zahrnutí protokolu, poznámek a popisu hardwarových zařízení.
- Processing - Jsou zde standardizované moduly zpracování, které často slouží jako součást průběžné analýzy dat před provedením vědecké analýzy. Tyto moduly mohou obsahovat data z rozsáhlejších nebo i menších analýz.
- Analysis - Obsahuje laboratorní a vlastní vědecké analýzy dat. Neexistuje žádný definovaný formát pro obsah této skupiny. Záleží to na konkrétní laboratoři nebo individuálním uživateli. Ovšem pro lepší manipulaci se doporučuje ukládat data ve standardních typech. Analytická data by měla být zdokumentována, aby mohla být sdílena s jinými laboratořemi.

Zdroj [1] podrobněji popisuje formát NWB. Jsou zde popsány jednotlivé datové typy a jejich rozšíření. Tato dokumentace je i doplněna obrázky pro lepší pochopení souvislostí.

3.3.2 Nástroje a podpora standardu

Tento datový standard poskytuje mnoho nástrojů pro zefektivnění a usnadnění práce s daty. Spousta z nich je vytvořena a podporována jinými komunitními skupinami a jsou v aktivním vývoji. Následně bude popsáno pár z těchto nástrojů. Prvním nástrojem je NWB Explorer. Jedná se o webovou aplikaci, která je vyvinuta společností MetaCell a umožňuje číst, vizualizovat a zkoumat obsah souborů NWB 2. Tato aplikace poskytuje rozhraní pro nástroj Jupyter Notebook, ve kterém lze provádět vlastní analýzy, zkoumání dat, vizualizace a další operace.

Druhým nástrojem je CaImAn, který se používá pro analýzu vápníků. Je v něm umožněna provádět manipulace se záznamem, korekce pohybu, extrakce zdroje, vizualizace výsledků a další. CaImAn také podporuje čtení a zápis dat v NWB 2.

Třetím nástrojem je Brainstorm, který se věnuje analýze mozkových záznamů. Jedná se o aplikaci, která je vyvinuta v programovacím jazyce Java a Matlab. Podporuje modality jako MEG (Magnetoencephalography), EEG, ECoG (Electrocorticography) a další. Aplikace poskytuje grafické prostředí pro předzpracování a analýzu dat. Jeho součástí je i vizualizace dat a interaktivní vizualizace anatomických topografií.

Čtvrtým nástrojem je NWBwidgets knihovna, která umožňuje vizualizovat NWB data v Jupyter Notebook. Tento nástroj umožňuje procházet hierarchickou strukturou souboru NWB a vizualizovat data každého prvku z tohoto souboru.

Posledním nástrojem, který bude zmíněn je EcogVIS. Jedná se o vizualizační nástroj časové řady pro signály ECoG, které jsou uložené v souborech NWB. Nástroj umožňuje snadno vizualizovat signály ECoG z vybraných kanálů, vytvářet anotace dat a označovat vybrané intervaly. S rozšířením pro zpracování a analýzu signálů se plánuje do budoucna [4].

Tento standard má k dispozici specifikační jazyk[6], který je založen na YAML (YAML Ain't Markup Language) nebo JSONu. Slouží k definování formální struktury pro popis organizace komplexních dat a používá se k rozšíření NWB formátu. Tento formát využívá pro skladování dat HDF5, protože splňuje několik klíčových požadavků pro jeho využití (i NIX ho používá). K dispozici je API pro Python (PyNWB) a Matlab (MatNWB) [39].

[6]https://schema-language.readthedocs.io/en/stable/

4 Neuroinformatická laboratoř

Součástí zadání je seznámit se s podobou dat uchovávaných neuroinformatickou laboratoří, která patří pod Fakultu aplikovaných věd Západočeské univerzity v Plzni. Členové této laboratoře se zaměřují na výzkum elektrické aktivity mozku použitím metod a technik elektroencefalografie. Při výzkumu vědci zkoumají především mozkovou odpověď, která je vyvolána událostmi a podněty (označováno jako ERP[1]).

Neuroinformatická laboratoř začala fungovat v roce 2005 a v dnešní době je vybavena řadou komerčních i vlastních hardwarových a softwarových zařízení. Kromě základní elektrofyziologické infrastruktury (zesilovač, software pro nahrávání a analýzu záznamů, software pro prezentaci podnětů a další) je součástí laboratorního vybavení i například zvukotěsná kabina a simulátor automobilu. Součástí automobilu je kabina, pedály a software pro simulaci prostředí řízení a samostatné jízdy [25].

4.1 EEGBase

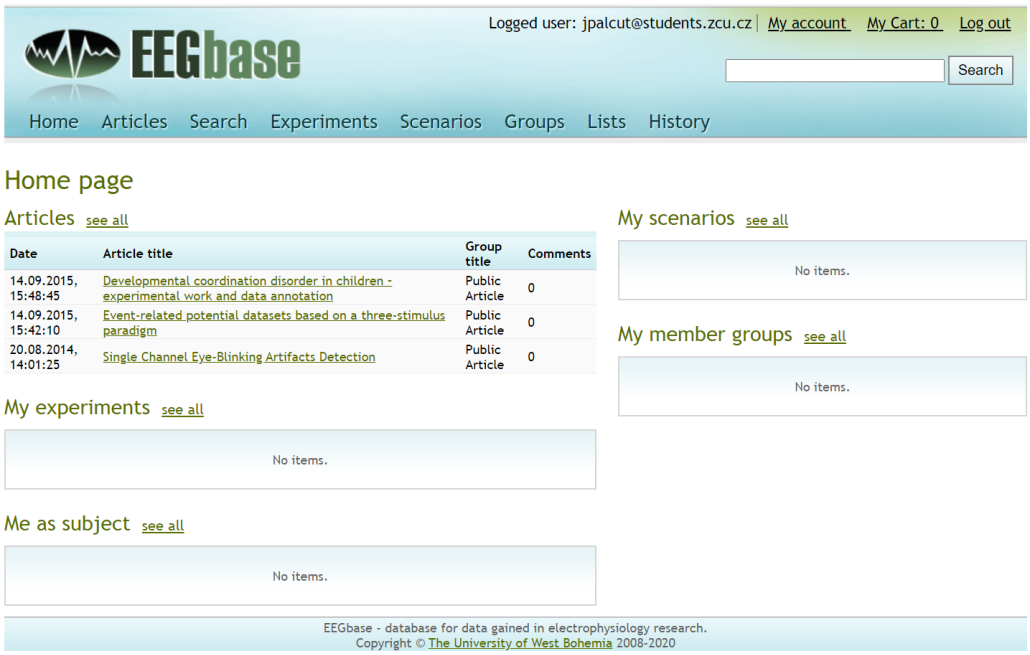
Neuroinformatická laboratoř vyvinula a spravuje EEGBase portál. Jedná se o webový systém, který umožňuje vědcům nahrávat, stahovat a spravovat EEG/ERP experimenty (data, metadata, experimentální scénáře a další). Funkce portálu také zahrnují sdílení znalostí, práce ve výzkumných skupinách, řízení vědeckých diskuzí, spouštění metod zpracování signálů a další funkcionality.

Uživatelé systému mají různé role a s tím související oprávnění. Registrovaní uživatelé jsou seskupeni do samostatně spravovaných skupin. Pro stahování a nahrávání experimentů musí být uživatel zaregistrován do systému a stát se členem alespoň jedné skupiny. Systém má definované různé role a pravomoci (čtenář, experimentátor, správce skupiny a supervizor).

Systém obsahuje jednoduchého průvodce, jehož účelem je provést přihlášeného uživatele procesem přidávání experimentu. Každý experiment obsahuje nezpracovaná data, včetně jejich doplnění o související metadata. Je definována sada metadat, kterou musí uživatelé prostřednictvím připravených formulářů vyplnit. Tato metadata jsou organizována do sémantických

[1]Event-related potential

skupin (experimentální protokol, experimentátoři a testované subjekty, použitý hardware, popis prvotních dat a další). Experimentátor může taktéž vytvořit experiment jako veřejný nebo soukromý. Veřejné experimenty si mohou stáhnout všichni registrovaní uživatelé (bez osobních údajů testovaných subjektů), zatímco k soukromým experimentům mají přístup jen experimentátoři skupiny. Systém taktéž obsahuje funkcionalitu hromadné správy více experimentů najednou [25]. Obrázek 4.1 představuje ukázku uživatelského rozhraní portálu po přihlášení uživatele. Funkční verze tohoto portálu je k dispozici na adrese http://eegdatabase.kiv.zcu.cz/.

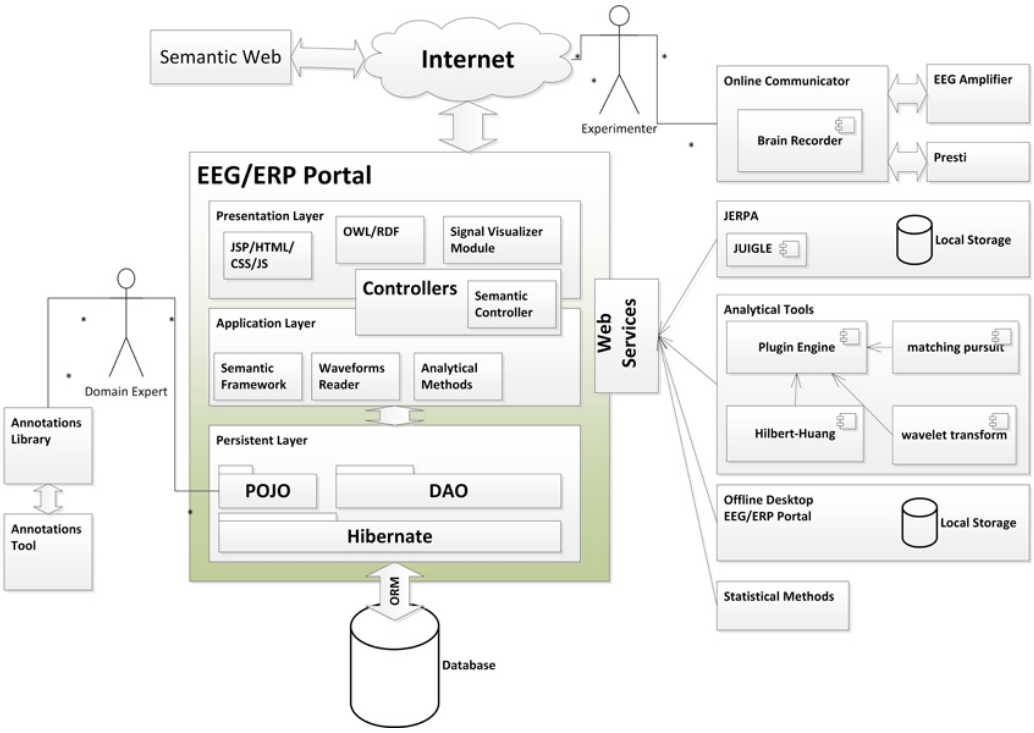


Obrázek 4.1: Uživatelské rozhraní EEGBase po přihlášení uživatele

4.2 Architektura

Celková architektura infrastruktury EEGBase portálu je zobrazena na obrázku 4.2. Základním cílem této infrastruktury je zvýšit efektivitu vědeckého výzkumu v oblasti EEG/ERP.

Centrálním bodem infrastruktury portálu EEGBase je služba poskytující rozhraní pro běžné uživatele a softwarové nástroje. Mezi hlavní vlastnosti této služby patří dlouhodobé a udržitelné ukládání dat a souvisejících metadat shromážděných z experimentů. Dále mezi další vlastnosti patří metody a pracovní postupy pro zpracování dat, které lze sdílet mezi skupinami experimentů. Taktéž je umožněno sdílet i běžná data a dokumenty.



Obrázek 4.2: Znázornění infrastruktury EEGBase portálu [25]

Kromě klasického webového rozhraní pro běžné uživatele bylo implementováno několik rozhraní pro komunikaci externích nástrojů. Mezi nástroje patří JERPA[2], offline a mobilní verze portálu EEGBase nebo prostředky pro vizualizaci signálů. Tyto nástroje komunikují s portálem pomocí webových služeb. Pro komunikaci se využívá SOAP (Simple Object Access Protocol) protokol a REST (Representational State Transfer) rozhraní. Další nástroje jsou implementovány jako knihovny integrované přímo v EEGBase. Nejdůležitějším z nich je Semantic Framework. Jeho cílem je poskytnout metadata experimentu v jazycích sémantického webu a jeho technologiích (RDF, OWL). Podstatnou část architektury tvoří několik hardwarových zařízení a softwarových nástrojů třetích stran. Ty jsou řízeny experimentátory pomocí webového prohlížeče nebo mobilní verze portálu.

Jádro portálu EEGBase tvoří framework Spring, který poskytuje komplexní programovací a konfigurační model podnikových aplikací založených na jazyce Java. Datová vrstva portálu v dřívější době používala databázový systém Oracle. V momentální době využívá NoSQL databázi s využitím fulltextového vyhledávání nástroje Elasticsearch. Pro mapování záznamů z databáze do objektů a zpět se používá framework Hibernate. Jeho výhodou je i značné usnadnění psaní dotazů. Prezentační vrstva je vytvořena framewor-

[2]desktopový systém pro zpracování a vizualizaci signálů

kem Apache Wicket, který usnadňuje implementaci opakovaně použitelných komponent psaných v jazyce Java a HTML (Hypertext Markup Language). Zajištění bezpečnosti aplikace a propojení se sociálními sítěmi zajišťují subkomponenty frameworku Spring [25].

4.3 Datové formáty, modely a ontologie

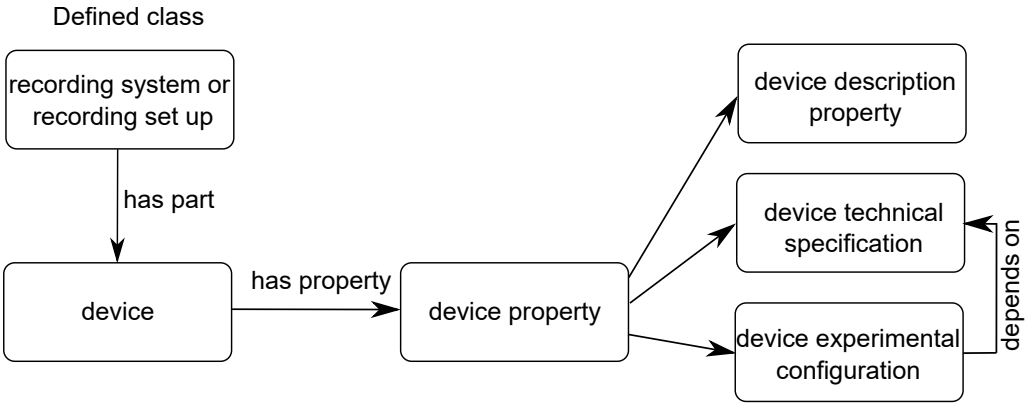
Datový model EEGBase portálu byl poprvé navržen v roce 2008 a od té doby se několikrát změnil. V současné době obsahuje základní ERA model více než 70 tabulek. Jeho důležitou vlastností je především flexibilita a možnost sdílení dat v rámci komunity. Hlavním cílem zdokonalování datového modelu a vývoje ontologie je pak zvýšit schopnost portálu sdílet data.

Použití ontologií je v dnešní době nejen doporučené, ale stává se i požadovaným prostředkem pro popis domén. Pro popis experimentů existují dvě známé ontologie, kterými jsou NEMO (Neural ElectroMagnetic Ontology) a OBI (Ontology for Biomedical Investigations). Ovšem tyto dvě zmíněné ontologie mají svoje nedostatky a nelze nimi plně popsat informace uložené na portálu EGGBase. Tento problém řeší vývoj a vytvoření ontologie OEN (Ontology for Experimental Neurophysiology). Skupina pracující na vývoji této ontologie byla vytvořena z členů následujících iniciativ:

- EEGBase protál (http://eegdatabase.kiv.zcu.cz/)
- G-Node (http://www.g-node.org/)
- NIF[3] (https://neuinfo.org/)
- INCF[4] (https://www.incf.org/)
- NeuroElectro (https://neuroelectro.org/)

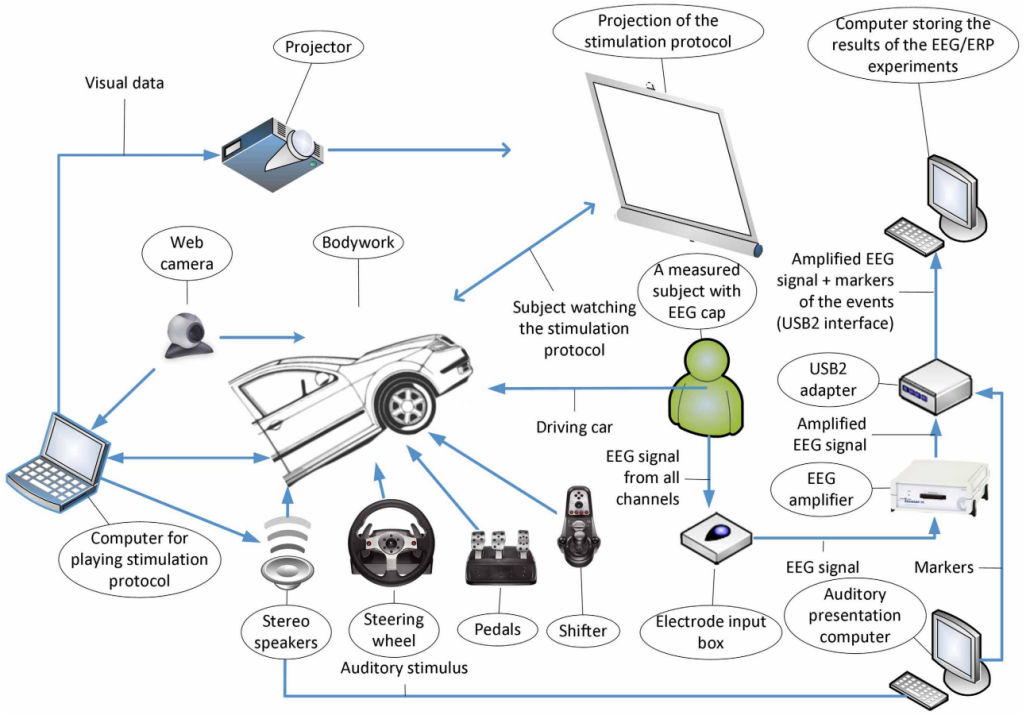
Vývoj ontologie OEN byl rozdělen do dvou částí. První větev se zabývá terminologií pro anotaci experimentálních metadat (jako jsou zařízení a metody). Druhá větev se naopak zabývá terminologiemi pro anotaci experimentálních dat (například akční potenciál). Znalostní model zařízení první větve je zobrazen na obrázku 4.3.

[3]Neuroscience Information Framework
[4]International Neuroinformatics Coordinating Facility



Obrázek 4.3: Znalostní model zařízení [25]

Pro definování terminologií a vytvoření první větve ontologie OEN bylo použito několik schémat. Zmíněným schématem je například experimentální nastavení pro zkoumání pozornosti řidiče nebo nastavení pro vizuální zkoumání experimentu mozkové kůry myši. První zmíněné schéma z nich je zobrazeno na obrázku 4.4. Terminologie ontologie OEN byly primárně zapsány ve formátu odML. Následně poté byl vytvořen soubor OWL za podpory společnosti Ontofox. Ontologie OEN je ke stažení k dispozici na adrese https://github.com/G-Node/OEN. Její vývoj byl zastaven [25].



Obrázek 4.4: Nastavení experimentu pozornosti řidiče [25]

Elektrofyziologická data jsou v neuroinformatické laboratoři zaznamenávána pomocí softwarového nástroje Brain Vision Recorder[5] a ukládána do tří souborů [41]. Těmito soubory jsou [41]:

- Datový soubor - Jedná se o binární soubor, který obsahuje získané hodnoty ze záznamového zařízení. Přípona tohoto souboru je eeg.
- Hlavičkový soubor - Textový soubor skládající se z různých sekcí, které obsahují páry klíč-hodnota. Soubor zahrnuje základní informace o měření (kódování, název datového souboru, název značkovacího souboru, počet kanálů, interval vzorkování v mikrosekundách, informace o kanálech a další). Přípona toho souboru je vhdr.
- Značkovací soubor - Struktura tohoto souboru je založena na stejném principu jako u hlavičkového souboru. Tento textový soubor obsahuje informace o podnětech (jeho číslo, typ, popis, pozice, velikost a další). Přípona tohoto souboru je vmrk.

4.4 Experimenty

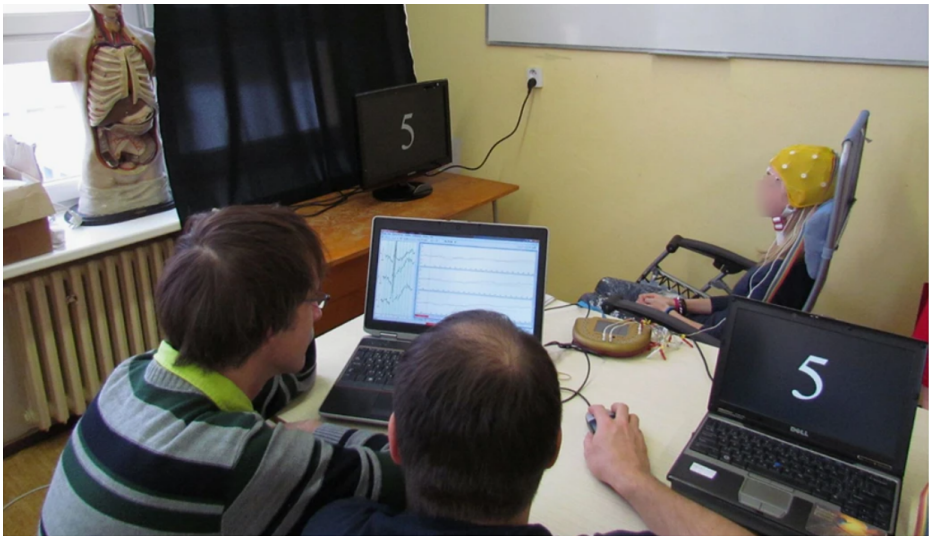
EEGBase obsahuje experimenty, které jsou veřejné i soukromé. Po přihlášení na portál si je lze zobrazit v sekci Experiments. Podrobnější informace o nich lze získat pak při jejich rozkliknutí. V této sekci bude popsáno pár vybraných z nich. Jsou jimi:

- Pozornost řidiče - Výzkumný tým spolupracuje s ČVUT v Praze na systému, který detekuje míru pozornosti řidiče v různých stresových dopravních situacích. Cílem je pomoci předcházet značnému množství nehod, které se na silnicích dějí. Pro tento účel se vyvíjí zařízení, které by mělo včas například detekovat zvýšenou únavu šoféra a poskytnout informace dalším systémům. Řidiče by to mělo varovat zvýšením hlasitosti rádia, varovným signálem nebo jiným způsobem. Výzkumný tým vyvíjí a používá metody pro stanovení míry pozornosti pomocí potenciálů souvisejících s událostmi. To jsou signály, které jsou vydávány, když mozek reaguje na vnější podnět. Výzkum je prováděn v laboratoři na Fakultě aplikovaných věd Západočeské univerzity v Plzni [27].
- Koordinační porucha u dětí - Porucha vývojové koordinace je jedním ze specifických negativních dopadů na učení. Je popsána jako narušený vývoj koordinace, který nelze vysvětlit celkovou mentální retar-

[5]https://www.brainproducts.com/index.php

dací nebo specifickou získanou nervovou poruchou. Tato porucha postihuje 6 % dětí školního věku a má významný dopad na školní výsledky i běžné každodenní činnosti. Součástí tohoto projektu byla spolupráce s Fakultní nemocnicí v Plzni a Centrem tělesné výchovy a sportu Západočeské univerzity. Při tomto experimentu byla získána data od 27 dětí ve věku od 6 až do 8 let. Pro stimulaci byl použit zvukový záznam. Děti během experimentu pozorovaly dětský program. Úkolem tohoto projektu bylo vyvinout a otestovat spolehlivou metodu pro stanovení motorického kvocientu pro předškolní děti [26].

- Hádej číslo - Jedná se o jednoduchý experiment, jehož se zúčastnilo 250 studentů základních a středních škol. Při tomto projektu byla shromážděna metadata o subjektech (pohlaví, věk, lateralita, počet hádáních, odhady experimentátorů a další doplňující informace). Taktéž byla uložena data ze tří EEG kanálů (Fz, Cz, Pz) a stimulační čísla. Účastníci tohoto projektu byli požádáni, aby si mysleli číslo mezi 1 až 9 a soustředili se na něj. Subjekt sedící na židli byl při experimentu vystaven vizuálním podnětům, které zahrnovaly náhodně se objevující čísla na monitoru v rozmezí 1 až 9. Experimentátoři se snažili uhodnout číslo, které si subjekt myslel. Odhad experimentátorů byl nakonec ověřen prozrazením myšleného čísla účastníkem [24]. Obrázek 4.5 znázorňuje ukázku při hádání myšleného čísla subjektu experimentátory.



Obrázek 4.5: Experiment hádání myšleného čísla subjektu [24]

5 Analýza a návrh

Tato kapitola se zabývá analýzou a návrhem výsledné webové aplikace. Budou zde použity získané znalosti z popisu prostředků sémantického webu a datových standardů pro popis elektrofyziologických dat. První část této kapitoly se bude zabývat požadavky na vytvoření webové aplikace. Budou zde popsány funkční i mimo funkční požadavky, které jsou kladeny na vytvořenou webovou aplikaci.

V druhé části bude proveden souhrn vlastností datových standardů pro popis elektrofyziologických dat. Na jejich základě následně dojde k výběru jednoho z nich. Vybraný standard bude poté použit při dalším návrhu řešení a následné implementaci aplikace.

Třetí část se bude zabývat výběrem webového frameworku v jazyce Python. Tento programovací jazyk se stal v oblasti neurověd jedním z nejpoužívanějších a je vhodné ho použít pro toto řešení. Taktéž existuje řada různých knihoven v tomto jazyce, které značně zefektivní vývoj výsledné webové aplikace.

Čtvrtá část této kapitoly se bude zabývat prozkoumáním obecných schémat pro strukturovaná data ze stránky https://schema.org/. V poslední části bude popsána architektura aplikace, konečně řešení převodu metadat experimentů a rozvržení webové aplikace. Taktéž zde dojde k výběru vhodných knihoven.

5.1 Specifikace požadavků

Pro jednodušší popis této sekce bude potřeba definovat pojem nahraný soubor. Pro tento účel bude jeden nahraný soubor představovat jeden experiment datového standardu. Na vytvořenou webovou aplikaci jsou kladeny různé požadavky. Tyto nároky lze rozdělit do dvou základních skupin. První skupinou jsou funkční požadavky. Těmi jsou:

- Vytvoření experimentu - Webová aplikace bude obsahovat nahrávání souborů. Půjde zde nahrát jeden nebo více experimentů pro vybraný datový standard (ten bude vybrán v sekci 5.2) najednou. To nám umožní spravovat hromadně metadata experimentů stejného charakteru. Při nahrávání souborů dojde k vytvoření experimentu.

- Správa nahraných souborů - U každého experimentu bude k dispozici tabulka, která bude obsahovat nahrané soubory. Tyto soubory půjde

stáhnout nebo smazat. Jejich metadata zde půjde jednotlivě nebo hromadně převést do nového formátu (ten bude popsán v sekci 5.5.1). Půjdou zde nahrát i další nové soubory (experimenty).

- Správa převedených souborů - Experiment bude obsahovat tabulku souborů, které budou obsahovat převedená metadata. Jednotlivé soubory zde půjde stáhnout nebo smazat.

- Vyhledávání v metadatech - K dispozici bude vyhledávání v metadatech experimentu. Půjde si zde zvolit experiment, ve kterém se provede hledání metadat.

- Zobrazení metadat - Pro každý experiment bude k dispozici stránka, kde půjde zobrazit metadata nahraných souborů.

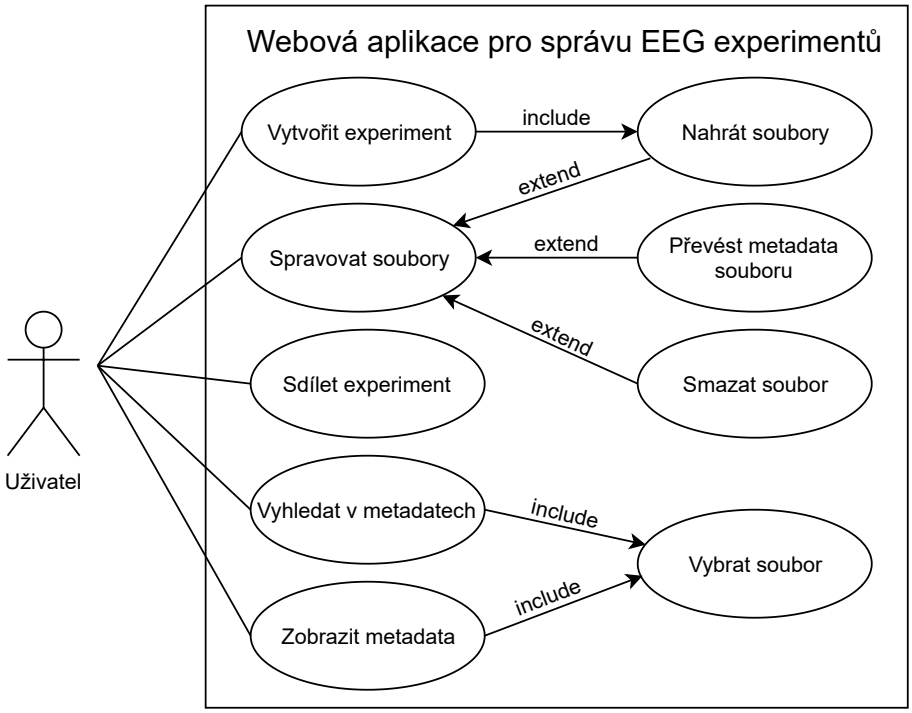
- Sdílení experimentu - Vytvořený experiment půjde sdílet mezi více uživateli pomocí URL odkazu.

Popis chování systému z hlediska uživatele znázorňuje obrázek 5.1. Kromě základních funkcionalit jsou kladeny na aplikaci i další požadavky. Jedná se o mimo funkční požadavky. Mezi ně patří:

- Jednoduchost - Webová aplikace by měla sloužit jako jednoduchý nástroj, který bude poskytovat základní nutné funkcionality. Z tohoto důvodu aplikace nebude například obsahovat registraci a ani přihlášení uživatelů.

- Přehlednost - Aplikace bude přehledná a půjde se v ní jednoduše orientovat i pro uživatele, kteří s ní budou pracovat poprvé.

- Bezpečnost - Vytvořená aplikace by měla být zabezpečena především proti vkládání různých skriptů, HTML značek a před dalšími bezpečnostními riziky. Přístup k experimentu bude zabezpečen pomocí vygenerovaného unikátního identifikátoru dostatečné délky.



Obrázek 5.1: Diagram případů užití webové aplikace

5.2 Souhrn datových standardů

Kapitola 3 obsahuje tři nepoužívanější datové standardy pro popis elektrofyziologických dat. Jsou jimi NIX, BIDS a NWB:N. Všechny tyto standardy mají stejný účel použití a základní podobné vlastnosti. Poskytují tudíž podporu pro ukládání komplexních i objemných dat včetně jejich metadat. To má zefektivnit možnost sdílení experimentů. Ovšem každý z těchto datových standardů k tomu přistupuje trošku jiným způsobem řešení.

NIX z těchto standardů poskytuje nejobecnější přístup k řešení ukládání dat a metadat experimentů. Obrázek 3.1 přestavuje ER diagram datového modelu tohoto standardu. Výhodou tohoto standardu je umožnění zapsat metadata pomocí odML terminologie. Pro tento účel obsahuje zmíněný model část nazvanou jako `Metadata storage`. Tento datový standard poskytuje podporu pro C++, Java, Python a Matlab. Pro ukládání dat využívá datového formátu HDF5. Důležitým nástrojem je NixView, který umožňuje prohlížet datové soubory tohoto standardu.

BIDS oproti dvěma ostatním datovým standardům představuje stromovou strukturu ukládání experimentů. Ta je zobrazena na obrázku 3.2. Nelze říci, že je na tomto méně obecném přístupu něco špatného. Struktura tohoto

standardu podporuje různou modalitu dat. Podrobnější popis této struktury obsahuje sekce 3.2.1. Tento standard momentálně podporuje programovací jazyk Python a Matlab. Na podpoře ostatních jazyků se stále pracuje. Oproti prvnímu popsanému standardu tento poskytuje větší množství nástrojů. Nástroje jsou k dipozici na adrese https://bids.neuroimaging.io/benefits.html.

Posledním zmíněným standardem je NWB:N. Tento standard uspořádá data hierarchicky pomocí snadno použitelných primitiv, kterými jsou Group, Dataset, Attribute a Link. Dále tento standard organizuje data do šesti hlavních skupin (Acquisition, Intervals, Stimulus, General, Processing a Analysis). Popis těchto primitiv a skupin je v sekci 3.3.1. Formát NWB používá modulární návrh, ve kterém všechny hlavní sémantické komponenty formátu mají typ neurodata_type. To umožňuje opětovně znovupoužití a rozšíření typů pomocí začlenění a dědičnosti. Typem nerodata_type lze ve formátu identifikovat všechny skupiny a datové sady. Lze si všimnout, že tento standard představuje oproti ostatním složitější přístup k ukládání dat a metadat experimentů. NWB:N stejně jako NIX využívá datového formátu HDF5. Podporovanými programovacími jazyky jsou Python a Matlab. Nástroje tohoto standardu jsou v sekci 3.3.2.

Tabulka 5.1 představuje kritéria výběru datového standardu. Prvním kritériem je umožnění jednoduchého nahrávání metadat experimentů na server. BIDS se zde ukázal jako jediný nevhodný z nich. Metadata experimentu jsou u tohoto standardu totiž uložena v různých souborech. Ostatní dva standardy obsahují metadata v jednom souboru. Druhým kritériem je jednoduchost datového standardu na pochopení. Standard NIX a BIDS představují struktury zápisu dat a metadat experimentů, které jsou jednoduché na pochopení. Oproti tomu standard NWB:N představuje složitější strukturu. Prozkoumání a pochopení tohoto standardu do detailu by mohlo zabrat značné množství času. Posledním kritériem je podpora jazyka Python. Tento jazyk podporují všechny zmíněné datové standardy. Na základě této tabulky došlo k výběru datového standardu NIX. Ten bude dále použit při návrhu a implementaci řešení.

Kritérium výběru	NIX	BIDS	NWB:N
Jednoduchost nahrávání metadat experimentů na server	ANO	NE	ANO
Jednoduchost datového standardu na pochopení	ANO	ANO	NE
Podpora jazyka Python	ANO	ANO	ANO

Tabulka 5.1: Kritéria výběru datového standardu

5.3 Výběr webového frameworku

Tato sekce se zabývá výběrem webového frameworku. V dnešní době existuje celá řada těchto frameworků a každý z nich poskytuje různé funkcionality. Výběr správného z nich dokáže značně zefektivnit práci při tvorbě webové aplikace. Zdroj [22] jich uvádí až třicet. Jejich základní rozdělení je na tři typy [22]:

- Full-Stack Framework - Tento typ poskytuje vývojářům největší podporu v podobě rozsáhlé možnosti funkcionalit. Zahrnuje vše od možnosti generování formulářů přes jejich validaci až po uložení dat. Tudíž tento typ dokáže značně urychlit vývoj, ale je nutné prostudovat jednotlivé způsoby zápisu a jeho použití více než u ostatních dvou typů.

- Micro Framework - Jedná se o odlehčený typ frameworku. Tento typ poskytuje pouze základní funkcionality. Nenajdeme zde žádné generování formulářů a ani jejich validaci. Oproti prvnímu typu umožňuje rychle přidat velké množství kódu a další požadavky bez nutnosti velkého studování frameworku.

- Asynchronous Framework - Jedná se o typ podobný druhému zmíněnému. Opět zde najdeme jen základní funkcionality. Tento typ umožňuje reagovat na změny na straně serveru a prezentovat tyto změny uživateli bez nutnosti interakce uživatele s rozhraním. Toto je umožněno použitím asyncio knihovny jazyka Python, kterou frameworky tohoto typu používají.

Na základě těchto tří popsaných typů frameworku se zdá být první varianta jako nejvhodnější pro řešení naší webové aplikace. Zmíněnými frameworky tohoto typu jsou například Django, CubicWeb, Giotto, Pyramid, Web2Py nebo Grok. Následující část textu bude popisovat tři vybrané z nich. Při jejich výběru byl dbán důraz především na velikost komunit jednotlivých frameworků, ale i na podporu technologií sémantického webu.

Prvním z nich je Django. Jedná se o nejpoužívanější framework z nich. Umožňuje vývojářům rychle a pružně pracovat na složitých kódech a aplikacích. Poskytuje rozsáhlý sortiment knihoven a funkcí, které zjednodušují mechanismus ověřování, správu obsahu, směrování URL, znovupoužití komponent, migraci schémat databáze a další funkcionality. Django podporuje databáze jako PostgresSQL, Oracle, MySQL a SQLite.

Druhým zmíněným frameworkem bude CubicWeb. Tento framework namísto definovaní pohledů a modelů využívá k vytváření webových aplikací

znovupoužitelné komponenty (tzv. kostky). Je umožněno sdružit více těchto komponent dohromady. Vytvořit instanci těchto komponent pak lze za pomoci konfiguračních souborů, webového serveru a případně databáze. Výhodou tohoto frameworku je podpora pro tvorbu sémantického webu. Součástí CubicWeb je dotazovací jazyk RQL (Relationship Query Language), který funguje podobně jako SPARQL. Taktéž podporuje v základu OWL a RDF.

Posledním zmíněným z nich bude Web2Py. Jedná se o framework, který podporuje škálovatelnost a má vlastní webové IDE. To poskytuje nástroje pro nasazení aplikace, editor kódu, testování a ladění aplikace. Web2Py poskytuje efektivní schopnost údržby aplikace a systém pro detekci chyb. Framework podporuje databáze Oracle, MySQL, PostgreSQL, MongoDB, DB2 a další. Web2Py byl navržen tak, aby zefektivnil vývoj aplikací a dokázal rychle vytvářet dynamický obsah webu [22].

Dále by šlo popsat některé další webové frameworky. Pro účel výběru nám ovšem postačí tyto tři. Jako nejvhodnější z nich se ukazuje Django a CubicWeb. Druhý zmíněný obsahuje prostředky a technologie sémantického webu, které už jsou uvnitř integrovány. To je určitě výhodou tohoto frameworku a ukazuje se jako vhodný pro použití. Oproti němu se prezentuje Django jako nejpoužívanější ze všech a má obrovskou komunitu, která za ním stojí. Taktéž poskytuje nejrozsáhlejší množství funkcionalit a podporuje rychlý vývoj webových aplikací. Funkcionality, které už CubicWeb obsahuje, lze pomocí externích knihoven integrovat i do frameworku Django. Využití externích knihoven nám umožní Django rozšířit jen o technologie sémantického webu, které nezbytně nutně potřebujeme. Výslednou aplikaci bude tudíž lepší vytvořit ve frameworku, který zná a používá více lidí. V případě, že by chtěl někdo na vytvořenou aplikaci navázat rozšířením funkcionalit, tak bude framework Django lepší volbou. Výsledná aplikace bude tudíž implementována v tomto frameworku.

5.4 Schema.org

Nežli dojde k návrhu řešení převodu metadat experimentů, tak bude prozkoumáno možné využití slovníku ze Schema.org. Zakladatelem Schema.org jsou společnosti Google, Microsoft, Yahoo a Yandex. Cílem této komunity je vytvářet, udržovat a propagovat schémata strukturovaných dat na internetu. Aktuálně Schema.org využívá více než 10 milionů webů k označení svých webových stránek a e-mailových zpráv [42].

Schémata jsou tvořené sadou typů (tříd), kde každý z nich obsahuje sadu jeho vlastností. Tyto typy jsou uspořádané v hierarchické struktuře. Každý

z těchto typů má jednu nebo více nadřazených tříd. Nejvyšším uzlem neboli kořenem této hierarchie je `Thing`. Jedná se o nejvíce generický typ této struktury. Jako jediný nemá nadřazenou třídu. Uváděný slovník momentálně obsahuje 829 tříd, 1351 vlastností a 339 výčtů hodnot [30]. Základními datovými typy jsou například Integer, Float, String, Boolean, DateTime, Date a Time.

U jednotlivých typů jsou uváděny příklady ve třech formátech zápisu. Těmito formáty jsou JSON-LD (JavaScript Object Notation for Linked Data), RDFa a Microdata. Ne vždy jsou u těchto příkladů vyplněny všechny tyto formáty. Tyto příklady slouží jako názorná ukázka použití těchto typů.

Výpis 5.1 představuje jednu z ukázek ve formátu JSON-LD ze stránky https://schema.org/Person. Tento výpis říká, že osoba jménem Joe Montana byla od roku 1979 do roku 1992 členem sportovního týmu San Francisco 49 ers. Dále nám říká, že osoba hrála americký fotbal na pozici quarterback. Z toho výpisu lze vyčíst, že zde byly použity tři typy, kterými jsou SportsTeam, Person a OrganizationRole. Použitými vlastnostmi těchto typů jsou name, member, startDate, endDate a roleName.

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "SportsTeam",
  "name": "San Francisco 49ers",
  "member": {
    "@type": "OrganizationRole",
    "member": {
      "@type": "Person",
      "name": "Joe Montana"
    },
    "startDate": "1979",
    "endDate": "1992",
    "roleName": "Quarterback"
  }
}
</script>
```

Výpis 5.1: Ukázka zápisu dat pomocí JSON-LD

Po podrobnějším prozkoumání jednotlivých typů a jejich vlastností se ukázalo, že Schema.org neobsahuje pojmy pro zapsání všech metadat experimentů. Zdroj [37] uvádí, že původní slovník je možné rozšířit. Původně k tomu docházelo pomocí tzv. hostovaných rozšíření (subdomén schema.org).

Od tohoto postupu už se upustilo. Schema.org se snaží v dnešní době rychleji rozšiřovat pojmy slovníku. Proces jejich schvalování může být zdlouhavý. Dochází zde k různým diskuzím ohledně zavedení nového pojmu do slovníku ve veřejných fórech. Po jejich vyhodnocení dojde následně buď k zařazení pojmu do slovníku nebo jeho vynechání.

Použití Schema.org se zdá z důvodu absence potřebných pojmů jako nevyhovující. Uvažovalo se nad možností rozšíření tohoto slovníku o pojmy z oblasti elektrofyziologie. Nicméně tento proces by mohl být zdlouhavý a mohlo by to zabrat měsíce, možná i rok. Komunita by musela podporovat přidání těchto nových pojmů. Z tohoto důvodu by bylo nutné, aby se v komunitních fórech o rozšíření vyjádřilo pozitivně několik neurovědců. I přes toto vše tu není jistota, že navržené řešení by bylo přidáno do slovníku.

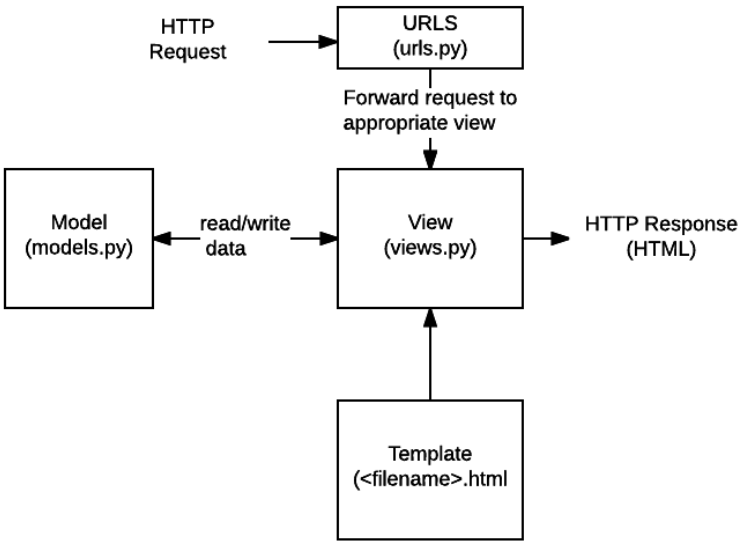
Navrhnout rozšíření slovníku by taktéž nemuselo být jednoduché. Už z obecného modelu NIX víme, že jsou metadata experimentu zapsána pomocí sekcí. Tyto sekce mají stromovou strukturu. Tudíž jedna sekce muže mít další podsekce. Jednotlivé sekce mohou obsahovat pole vlastností a ty mají pole hodnot. Při zápisu metadat do této struktury lze využít odML terminologií[1]. Ovšem při podrobnějším prozkoumání metadat u tří vybraných NIX souborů (tyto experimenty budou popsané v sekci 5.5.1) bylo zjištěno, že ne vždy je dodrženo použití těchto terminologií. Tomuto je dáno například různorodostí experimentů. Ne vždy jsou všechny pojmy v této sbírce termínologií definovány. Schema.org by šlo by rozšířit o pojmy z poslední verze této terminologie. Nicméně stále by tu byl problém, že ne vždy by šla všechna metadata zapsat pomocí tohoto rozšíření. Proto bude lepší zamítnout rozšíření slovníku.

5.5 Architektura

V této sekci bude popsána použitá architektura a dojde k navržení řešení převodu metadat experimentů. Taktéž zde bude proveden výběr důležitých knihoven a dojde k popisu rozvržení webové aplikace.

Frameworku Django používá architekturu MVT (Model-View-Template). Tato architektura je podobná MVC (Model-View-Controller). Jediným důležitým rozdílem mezi nimi je, že u MVT dochází k automatické interakci mezi Model a View. To ušetří množství napsaného kódu a zjednoduší tvorbu webové aplikace. Ještě je tu změna v názvech. View z MVT si lze představit jako Controller architektury MVC. Template zase naopak jako View [8].

[1] https://terminologies.g-node.org/v1.1/terminologies.xml



Obrázek 5.2: Architektura MVT [9]

Obrázek 5.2 obsahuje znázornění použití architektury MVT u frameworku Django. Tato architektura bude použita při implementaci aplikace. Ovšem dojde zde k jedné změně v této architektuře. Model je na obrázku 5.2 znázorněn jako soubor `models.py`. Tento soubor bude nahrazen souborem `experiments.py`. V souboru `experiments.py` budou vytvořeny funkce pro práci s experimenty. Půjde především o funkce týkají se vytváření adresářů experiment, nahrávání NIX souborů, mazaní těchto souborů a další. Vytvoření vlastních funkcí pro práci s experimenty, na místo definování tříd v `models.py`, urychlí vývoj aplikace. K tomuto kroku došlo na základě prozkoumání pár ukázek kódů týkající se právě vytváření adresářů a nahrávání souborů.

5.5.1 Převod metadat experimentů

Tato sekce se zabývá návrhem řešení převodu metadat experimentů. K dispozici je sedm typů experimentů. Stránka https://gin.g-node.org/jsedivy obsahuje jejich repozitáře. Každý z těchto repozitářů má popis jeho obsahu a několik experimentů. Pro prozkoumání obsahu vybraných NIX souborů byl použit program HDFView verze 3.1.1. Tento program umožňuje nahlédnout na data i metadata experimentu uvnitř těchto souborů. Tři nejvhodnější experimenty z nich byly vybrány pro návrh řešení. Jedná se o typově rozdílné experimenty. Šlo především o to vybrat takové experimenty, které mají rozdílná metadata. To nám lépe pomůže navrhnout řešení. Vybranými NIX soubory jsou (budou zde vypsány zkrácené názvy souborů):

- Experiment_276 - Dostupný v repozitáři[2]. Tento experiment obsahuje ze všech jmenovaných nejjednodušší strukturu dat a metadat. Byla zde zkoumána porucha vývojové koordinace.
- Experiment_341 - Tento experiment je dostupný v repozitáři[3]. Subjekt byl při tomto experimentu vystaven vizuální stimulaci třech typů. Šlo o cílové, necílové stimuly a stimuly pro rozptýlení subjektu.
- Experiment_91 - Vybrán z PROJECT_DAYS_P3_NUMBERS repozitáře. Jedná se o experiment při němž experimentátoři hádali myšlené číslo subjektu.

Je potřeba si uvědomit, že struktura metadat bude záležet na neurovědci, který ji zapíše. V těchto souborech se opakují sekce, které jsou zapsány pomocí odML terminologie. Těmito opakující sekcemi jsou například Electrodes, HardwareProperties, Experiment, Project, Experimenters, Recording, Subject a další. Nicméně i tyto sekce mohou obsahovat vlastnosti, které nedodržují terminologii dané verze. Taktéž tyto sekce mohou obsahovat další podsekce. Některé sekce nejsou ani zapsány pomocí odML terminologie. Z tohoto důvodu se zdá nejlepším řešením provést rekurzivní průchod stromové struktury těchto sekcí. To nám zajistí získání všech potřebných metadat experimentů. Metadata těchto sekcí budou zapsána do formátu JSON-LD.

Při převodu nesmíme zapomenout na metadata týkající se datových polí. Ta jsou uložena uvnitř bloků v NIX souboru. Součástí těchto bloků jsou i značky pro označení pozicí v těchto datových polí. Jednotlivá datové pole mohou být i shromážděna do skupin. U datového pole jsou pro nás důležitá metadata jako velikost dimenze, jednotky, označení pole a další. Datové pole, blok i značky mohou odkazovat na metadata sekcí. I metadata této datové části budou převedena a zapsána ve formátu JSON-LD.

Následující část textu bude obsahovat pár ukázek zápisu metadat do formátu JSON-LD. Při těchto příkladech budou použita metadata ze třech zmíněných experimentů. Na základě těchto ukázek pak budou převedena metadata při implementaci. Ještě předtím než představíme pár ukázek, tak bude potřeba definovat kontext. Ten umožní mapovat použité termíny na IRI (Internationalized Resource Identifier). IRI je rozšířením URI o další znaky ze znakové sady UCS (Universal Coded Character Set). Mapování umožní zjednodušit zápis těchto výrazů. Výpis 5.2 představuje zápis kontextu v JSON-LD, který bude použit u všech převodů experimentů.

[2]https://gin.g-node.org/jsedivy/Developmental_coordination_disorder_in_children_-_experimental_work_and_data_annotation

[3]https://gin.g-node.org/jsedivy/Event-related_potential_datasets_based_on_three-stimulus-paradigm

```
"@context": [
  {
    "@vocab": "http://example.com/eeg/"
  }]
```

Výpis 5.2: Zápis kontextu v JSON-LD

Výpis 5.3 představuje zápis metadat experimentátorů z NIX souboru ve formátu JSON-LD. Ve struktuře lze vidět jejich role, křestní jméno, příjmení a pohlaví.

```
"experimenters": {
      "experimenter": {
        "role": "Experimenter",
        "lastName": "Moucek",
        "firstName": "Roman",
        "gender": "Male"
      },
      "experimenter#1": {
        "role": "Experimenter",
        "lastName": "Mautner",
        "firstName": "Pavel",
        "gender": "Male"
      }
}
```

Výpis 5.3: Zápis metadat experimentátorů v JSON-LD

Metadata zápisu použitých elektrod při experimentu do formátu JSON-LD představuje výpis 5.5. Je zde vidět název elektrody, který říká o jakou elektrodu jde. Dále je součástí struktury popis, typ nebo model elektrody. V popisu může být například uvedena velikost elektrody nebo další údaje.

```
"subject": {
    "age": "26",
    "gender": "Female",
    "healthStatus": "myopia",
    "personInfo": {
        "laterality": "right-handed"
    }
  }
```

Výpis 5.4: Zápis metadat subjektu v JSON-LD

Poslední ukázkou je výpis 5.4. Ten obsahuje zapsaná metadata subjektu do formátu JSON-LD. Je zde vidět věk, pohlaví, lateralita a zdravotní potíže subjektu. Mohli bychom dále zobrazit další množství příkladu převodu metadat experimentů. Pro ukázku pár těchto základních bude stačit.

```
"electrodes": {
  "referenceElectrode": {
    "type": "E21-9S DISK ELECTRODE",
    "description": "......."
  },
  "groundElectrode": {
    "model": "E5-9S EAR ELECTRODE",
    "description": "......."
  }
}
```

Výpis 5.5: Zápis metadat elektrod v JSON-LD

5.5.2 Knihovna Nixio

Jedná se o knihovnu pro datový standard NIX. Tato knihovna umožňuje vytvářet soubory tohoto standardu, číst jejich obsah a zapisovat do nich. Do souboru NIX lze zapsat data i metadata experimentu. Zapsaná data a metadata dodržují strukturu, která je popsána v sekci 3.1.1. Momentální verze této knihovny je 1.4.9. Tuto knihovnu lze nainstalovat zadáním příkazu `pip install nixio` do příkazové řádky Linuxu. Příkazem se nainstaluje aktuální stabilní verze Nixio [11].

Této knihovny bude využito při implementaci aplikace. Následující text bude obsahovat ukázky práce s touto knihovnou. Výpis 5.6 představuje vytvoření NIX souboru. První řádka umožňuje importovat Nixio knihovnu. Další úsek kódu vytváří NIX soubor s příponou h5. První parametr obsahuje cestu k souboru a jeho název. Druhý představuje způsob otevření souboru. Kromě způsobu `Overwrite` (vytvoření souboru) lze využít i `ReadOnly` (jen pro čtení) a `ReadWrite` (pro čtení a zápis) [10, 13].

```
import nixio as nix

nix_file = nix.File.open('example.h5',
                         nix.FileMode.Overwrite)
```

Výpis 5.6: Vytvoření NIX souboru [13]

Block lze vytvořit funkcí `nix_file.create_block()`. Jedná se o entitu, která seskupuje další datové entity patřící například do provedení jednoho měření na subjektu. Výpis 5.7 představuje ukázku provádění operací s touto entitou. První řádka tohoto kódu vybírá objekt Block, s kterým se bude pracovat. Dále tento výpis obsahuje tři úseky, které jsou odděleny komentáři. První úsek slouží pro přiřazení metadat k objektu Block. Druhý ukazuje, jak získat jeho metadata. Třetí slouží k odstranění metadat bloku [10].

```
block = nix_file.blocks[some_id]

# prirazeni metadat k bloku
section = nix_file.sections[sec_id]
block.metadata = section

# ziskani metadat z bloku
block.metadata

# odstraneni metadat z bloku
block.metadata = None
```

Výpis 5.7: Použití entity Block v NIX souboru [10]

Výpis 5.8 představuje ukázku s datovou částí uvnitř objektu Block. Pole `observations` obsahuje hodnoty, z kterých bude vytvořeno nové datové pole. K tomuto dochází v kódu funkcí `create_data_array()`. Úsek pod touto funkcí slouží k přiřazení popiskům dimenze. Ty jsou uloženy v proměnné `categories`. Jsou v ní obsaženy měsíce v roce. Tato ukázka kódu slouží k vytvoření histogramu [13].

```
observations = [0, 0, 5, 20, 45, 40,
                          28, 12, 2, 0, 1, 0]
categories = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
              'Jul', 'Aug','Sep','Oct','Nov', 'Dec']
data = block.create_data_array("observations",
                    "nix.histogram", data=observations)
dim = data.append_set_dimension()
dim.labels = categories
```

Výpis 5.8: Práce s datovou částí entity Block [13]

Dále by šlo uvést další množství příkladů. Stránka [13] jich obsahuje několik. Zdroj [10] obsahuje funkce pro práci s datovou částí modelu NIX. Funkce pro práci metadatovou částí modelu zobrazuje stránka [12]. Při implementaci bude využito funkcí z obou jmenovaných zdrojů.

5.5.3 Knihovna RDFLib

V jazyce Python existuje spousta knihoven, které lze využít pro práci s daty sémantického webu. RDFLib se ukázala jako nejpoužitelnější z nich. Je tomu především z důvodu podpory formátu JSON-LD. Knihovna prošla už řadou verzí a postupně se dále vyvíjí. Její momentální verze je 5.0.0.

RDFLib knihovna poskytuje řadu funkcionalit. Nejdůležitější z nich je možnost zpracování dat v různých formátech pomocí funkce `parse()`. Tato funkce umožňuje převést data do podoby RDF grafu (trojic) a dále s nimi pracovat. Množinu trojic lze poté jednoduše rozšířit nebo jednotlivé elementy této množiny upravit. Podporovanými formáty jsou například N3, RDF/XML, NTriples, N-Quads, Turtle, TriX, RDFa a JSON-LD. Knihovna taktéž umožňuje převést množinu trojic zpět do těchto zmíněných formátů.

Další funkcionalitou RDFLib je umožnění tvorby dotazů. Tato knihovna totiž obsahuje dotazovací jazyk SPARQL. Pomocí různých dotazů lze upravovat RDF graf nebo vybrat část jeho podgrafu. Výhodou je především jednoduchost zápisu dotazů. Použití dotazovacího jazyku SPARQL bude vhodné pro vyhledávání v metadatech experimentů. Knihovna obsahuje taktéž další funkcionality. Avšak tyto zmíněné jsou nejdůležitější [34].

Výpis 5.9 obsahuje ukázku použití této knihovny. První řádka importuje knihovnu RDFLib. Třetí řádka v kódu vytváří RDF graf. Ten je následně naplněn pomocí funkce `parse()`. Proměnná `url` obsahuje odkaz na data, která budou použita.

```
import rdflib

g = rdflib.Graph()
url = "http://www.w3.org/People/Berners-Lee/card"
result = g.parse(url)

for subj, pred, obj in g:
    if (subj, pred, obj) not in g:
        raise Exception("It better be!")

print("graph has {} statements.".format(len(g)))
print(g.serialize(format="turtle").decode("utf-8"))
```

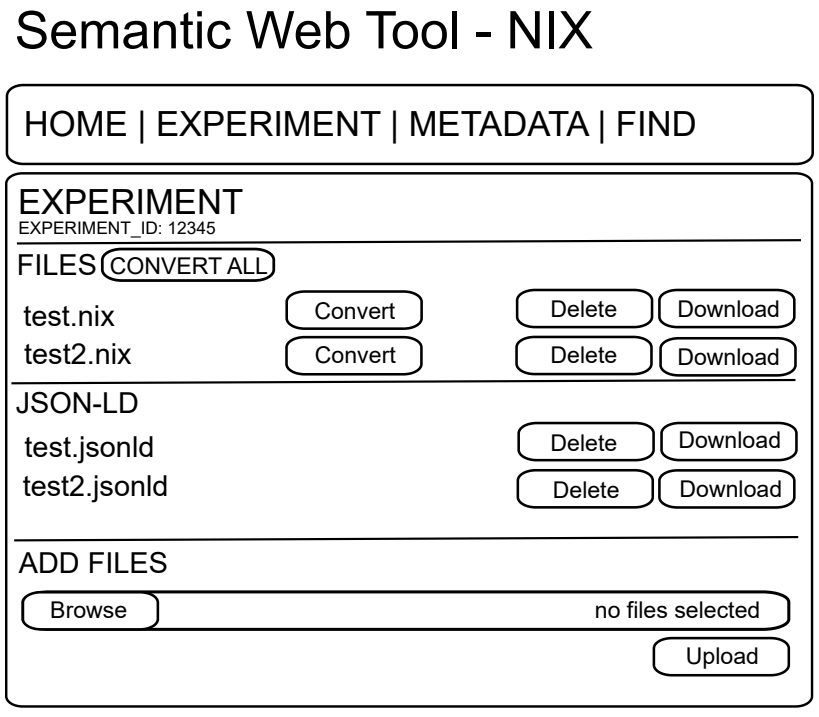
Výpis 5.9: Ukázka práce s knihovnou RDFLib [35]

Následující úsek kódu obsahuje cyklus, který projde všechny trojice nacházející se v RDF grafu. Uvnitř cyklu se nachází podmínka. Jejím úkolem je ověření výskytu alespoň jedné trojice. Při nenalezení žádné trojice dojde

k vyvolání výjimky. Konec kódu obsahuje dva výpisy. První z nich vypíše počet nalezených trojic. Pro uvedenou adresu, která je uvnitř kódu, je tato hodnota 86. Poslední řádka kódu poté vypíše data RDF grafu ve formátu Turtle [35].

5.5.4 Rozvržení webové aplikace

Webová aplikace bude obsahovat hlavní navigační panel. Tento panel bude dynamicky zobrazovat odkazy. Home bude jediný viditelný z nich po celou dobu. Ten bude odkazovat na hlavní stránku aplikace. Tato stránka bude obsahovat nahrávání experimentů pro datový standard NIX. Jeden soubor tohoto datového standardu reprezentuje právě jeden experiment. Půjde nahrát jeden nebo více souborů s příponou nix nebo h5. V případě vyskytnutí chyby u nahrávání souborů se zobrazí příslušná hláška. Jelikož soubory mohou mít až stovky megabytů, tak bude součástí této stránky animace pro jejich nahrávání.



Obrázek 5.3: Stránka pro správu experimentu

Při úspěšném nahrání NIX souborů dojde k vytvoření nového experimentu a přesměrování na jeho správu. Taktéž se zobrazí další odkazy v hlavní navigační liště. Těmito odkazy budou Experiment, Metadata a Find. Správa

experimentu bude zobrazena právě pod prvním jmenovaným z nich. Bude zde obsaženo nahrávání NIX souborů, vygenerovaný identifikátor experimentu a dvě tabulky. První tabulka bude obsahovat nahrané NIX soubory. V druhé budou JSON-LD soubory s převedenými metadaty. Z obou tabulek půjde tyto soubory smazat i stáhnout. Pro první zde bude navíc i umožněno převést metadata nahraných NIX souborů do formátu JSON-LD. Tyto soubory půjdou převést jednotlivě nebo hromadně najednou. Obrázek 5.3 zobrazuje ukázku rovržení této stránky.

Odkaz Metadata bude obsahovat seznam JSON-LD souborů s převedenými metadaty. Z tohoto seznamu půjde vybrat jeden z nich. Kliknutím na tlačítko zobrazit dojde k výpisu obsahu souboru v textovém poli. Když nebude experiment obsahovat žádné převedené soubory s metadaty, tak se na této stránce zobrazí příslušná hláška, která o tom informuje.

Stránka Find bude sloužit pro dotazování se na metadata experimentů. Bude zde k dispozici textové pole, kde půjde napsat dotaz v jazyce SPARQL. Taktéž zde bude seznam JSON-LD souborů, ze kterého půjde jeden z nich vybrat. Kliknutí na tlačítko najít se provede vyhodnocení tohoto dotazu a zobrazí příslušný výsledek v nově zobrazeném textovém poli. Při nenalezení žádných JSON-LD souborů se zde taktéž zobrazí příslušná hláška, která o tom informuje. Tento návrh popisuje základní rozložení obsahu webové aplikace. Podrobnější funkcionality aplikace budou záležet na konkrétní implementaci řešení.

6 Implementace

Tato kapitola se zabývá implementací webové aplikace na základě popisu jejího návrhu. V úvodu této kapitoly bude proveden souhrn použitých technologií a jejich verzí. Další část bude obsahovat popis adresářové struktury vytvořeného projektu ve frameworku Django a souborů obsažených uvnitř této struktury. Následně poté budou popsané vybrané funkce, algoritmy nebo zajímavé části webové aplikace.

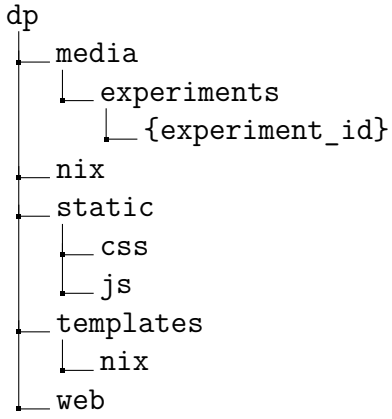
6.1 Použité technologie

Webová aplikace byla programována v operačním systému Manjaro. Jedná se o jednu z distribucí Linuxu. Jeho použitá verze byla 20.0.3. Vytvořená aplikace byla vyvíjena ve vývojovém prostředí PyCharm verze 2020.1.2. Tato sekce dále obsahuje verze důležitých použitých jazyků, knihoven a frameworků, které byly použity v rámci implementace webové aplikace. Tyto technologie zde nebudou popsány, neboť některé z nich už byly popsány v rámci návrhu aplikace a zbylé z nich jsou obecně známé. Použitými technologiemi jsou:

- Python 3.8.3
- Django 3.0.7
- HTML5
- CSS3 (Cascading Style Sheets 3)
- Bootstrap 4.5.0
- jQuery 3.5.1
- Nixio 1.5.0b4
- RDFLib 5.0.0

6.2 Struktura webové aplikace

Tato sekce zobrazuje strukturu webové aplikace ve frameworku Django. Bude zde popsán účel jednotlivých složek a souborů, které jsou jejich součástí. Obrázek 6.1 znázorňuje stromovou strukturu adresářů projektu.



Obrázek 6.1: Adresářová struktura webové aplikace

Hlavní složkou stromové struktury je `dp`. Ta obsahuje dalších pět adresářů. Těmi jsou `media`, `nix`, `static`, `templates` a `web`. První zmíněná složka z nich obsahuje adresář `experiments`. Do tohoto adresáře se ukládají vytvořené experimenty. Pro každý experiment je zde vytvořena nová složka, která má název podle jeho identifikátoru. Tento identifikátor má unikátní číslo pětimístné délky. Každý z těchto experimentů dále obsahuje soubory, které mohou mít příponu `nix`, `h5` nebo `jsonld`. První dvě přípony označují nahrané experimenty, které půjde spravovat hromadně. Soubory s `jsonld` příponou obsahují převedená metadata NIX souborů do formátu JSON-LD.

Statické soubory aplikace jsou uloženy uvnitř adresáře `static`. Tento adresář dále obsahuje podsložky `css` a `js`. Složka `css` obsahuje soubor `style.css`, ve kterém jsou napsané vlastní kaskádové styly. Uvnitř druhé zmíněno podsložky je soubor s názvem `script.js`. Ten obsahuje funkce napsané v jazyce JavaScript s využitím knihovny jQuery.

Složka `templates` obsahuje šablony tohoto projektu. Uvnitř tohoto adresáře jsou čtyři soubory. Jsou jimi 400.html, 403.html, 404.html a 500.html. Jedná se o šablony pro zobrazení stavového kódu HTTP (Hypertext Transfer Protocol) protokolu při odpovědi serveru na klientský požadavek. Tyto šablony jsou obecné a šlo by je použít v rámci více aplikací. Dále je součástí `templates` adresář `nix`, který obsahuje šablony týkající naší vytvořené aplikace. Tato struktura ukládání šablon je doporučená. Je to z důvodu možnosti rozšířit projekt o další aplikace. Uvnitř adresáře `nix`, který se nachází ve složce `templates` jsou soubory:

- default.html - Jedná se o základní šablonu webové aplikace. Je zde definována hlavička webové stránky. Uvnitř ní je použit element `link` pro vložení kaskádových stylů. Elementu `script` je zde využito pro

odkázání na vlastní i externí soubory, které obsahují kód v jazyce JavaScript. Součástí této šablony je hlavní navigační lišta a zobrazení úspěšných i chybových hlášek. Další vytvořené šablony jsou tvořeny jako rozšíření této šablony. To odstraní například duplicitu opakovaného zapsání stejné hlavičky nebo navigační lišty v každé šabloně.

- upload_experiment.html - Tato šablona slouží pro vytvoření experimentu. Součástí šablony je nahrání experimentů datového standardu NIX. Při úspěšném zpracování formuláře se vytvoří experiment, který umožní spravovat nahrané experimenty stejného charakteru najednou.

- experiment.html - Tato šablona slouží pro správu experimentu. Jsou zde k dispozici dvě tabulky. První z nich obsahuje nahrané soubory datového standardu NIX. V druhé jsou převedená metadata experimentů do formátu JSON-DL. Z obou těchto tabulek jdou soubory stahovat i mazat. První tabulka navíc umožňuje jednotlivě nebo hromadně převést metadata experimentů do formátu JSON-LD. Součástí této šablony je i nahrávání experimentů. To umožňuje experiment rozšířit o další NIX soubory.

- metadata.html - Tato šablona umožňuje zobrazit obsah metadat experimentů. Je zde výběrové pole, kde se vybere jeden z experimentu. Kliknutím na tlačítko `Show metadata` se následně zobrazí metadata vybraného souboru.

- find.html - Tato šablona umožňuje vyhledávat metadata v experimentech. Je zde textové pole pro zápis dotazu pomocí dotazovacího jazyka SPARQL. Součástí této stránky je i výběrové pole se seznamem experimentů. Nad vybraným experimentem se poté provede dotaz.

Další zmíněným adresářem bude `web`. Ten se nachází uvnitř hlavního adresáře `dp`. Jsou zde konfigurační soubory pro nastavení celého projektu. Tato složka obsahuje několik souborů. Jsou jimi `__init__.py`, `asgi.py`, `urls.py`, `settings.py`, `wsgi.py`. První z nich má prázdných obsah a existuje v projektu pouze k označení adresáře jako balíčku pro interpret jazyka Python. Soubor `settings.py` obsahuje základní konfigurace uvnitř projektu. Tento soubor obsahuje nastavení šablon, databáze, časové zóny, nainstalované aplikace, definovaní URL pro statické soubory, zapnutí ladění aplikace a další. Soubor `urls.py` obsahuje informace o adresách URL na úrovni projektu. Všechny URL jsou zpracovány přes tento soubor. Na základě zpracování URL tímto souborem dojde k přesměrování požadavku do příslušné aplikace

v projektu. Soubory `wsgi.py` a `asgi.py` obsahují nastavení pro nasazení webových aplikací na server.

Na závěr se dostáváme k adresáři `nix`. Tato složka obsahuje soubory týkající se vytvořené webové aplikace pro správu elektroencefalografických experimentů. Uvnitř tohoto adresáře je opět soubor `__init__.py`. Jeho účel je totožný jako u adresáře `web`. Dále je součástí `nix` soubor `urls.py`. Jsou zde definovány vzory URL adres pro tuto aplikaci. Na základě vyhodnocení URL tímto souborem poté dojde k zpracování požadavku příslušnou funkcí. Soubor `apps.py` je konfiguračním souborem pro tuto aplikaci. Soubor `views` obsahuje funkce pro zpracování požadavků. U nich se opakují parametry `request` (data požadavku), `id` (identifikátor experimentu) a `name` (název souboru). Důležitými funkcemi uvnitř tohoto souboru jsou:

- show_home_page(request) - Zobrazí hlavní stránku aplikace. Ta slouží k vytvoření experimentu.
- show_experiment_page(request, id) - Zobrazí správu experimentu dle identifikátoru experimentu.
- download_file(request, id, name) - Stáhne soubor dle identifikátoru experimentu a názvu souboru.
- show_metadata_page(request, id) - Zobrazí stránku pro zobrazení metadat experimentů podle identifikátoru experimentu.
- delete_file(request, id, name) - Smaže soubor podle identifikátoru experimentu a názvu souboru.
- show_find_page(request, id) - Zobrazí stránku pro vyhledávání v metadatech experimentů podle identifikátoru experimentu.
- upload_experiment(request) - Vytvoří experiment včetně nahrátí NIX souborů.
- show_metadata(request, id) - Zobrazí metadata vybraného experimentu.
- process_query(request, id) - Zpracuje dotaz v jazyce SPARQL a zobrazí výsledek.
- upload_files(request, id) - Slouží k přidání NIX souborů k experimentu.
- convert_all(request, id) - Převede všechny NIX soubory experimentu do formátu JSON-LD.

- convert_file(request, id, name) - Převede NIX soubor podle identifikátoru experimentu a názvu souboru do formátu JSON-LD.

Dalším souborem je `experiment.py`. Ten obsahuje funkce, které jsou použity například pro generování identifikátoru experimentu, uložení NIX souborů, přečtení obsahu souboru a další. Posledním souborem v adresáři `nix` je `convert.py`. Ten obsahuje funkce pro převedení metadat NIX souboru do formátu JSON-LD.

V adresáři `dp` se nacházejí dva soubory. Jsou jimi `requirements.txt` a `manage.py`. Druhý jmenovaný slouží jako nástroj příkazového řádku pro provádění příkazů frameworku Django v rámci projektu. Umožňuje například vytvářet aplikace, spustit server pro zavádění webových aplikací nebo pracovat s databázemi. Soubor `requirements.txt` obsahuje soupis použitých knihoven a jejich vybrané verze. Webovou aplikaci včetně nainstalovaní příslušných knihoven lze ze složky `dp` spustit postupným zadáním příkazů z výpisu 6.1 do konzole Linuxu. Pak lze k aplikaci přistoupit adresou http://127.0.0.1:8000/ ve webovém prohlížeči.

```
virtualenv venv
source ./venv/bin/activate
pip install -r requirements.txt
python manage.py runserver
```

Výpis 6.1: Spuštění webové aplikace přes konzoli

6.3 Ukázky kódu

V této sekci budou zobrazeny a popsány vybrané algoritmy, funkce nebo úseky kódu. Výpis 6.2 zobrazuje kód funkce pro ověření existence vytvořeného experimentu. Jejím parametrem je identifikátor experimentu. Funkce vrací `True` v případě existence experimentu. V opačném případě je vrácena hodnota `False`.

```
def experiment_exists(experiment_id):
    fs = FileSystemStorage()
    if fs.exists('experiments/' + experiment_id + '/'):
        return True
    return False
```

Výpis 6.2: Kontrola existence experimentu

Výpis 6.3 obsahuje kód pro generování identifikátoru experimentu. Tento kód je uvnitř funkce `generate_experiment_id()`. Pro generování náhodného čísla je použita knihovna `random`. Uvnitř tohoto kódu je nekonečný cyklus, který má zastavující podmínku. Ta zastaví tento cyklus při úspěšném vygenerování náhodného čísla.

```
while True:
    experiment_id = str(random.randint(10000, 99999))
    if not experiment_exists(experiment_id):
        break
return experiment_id
```

Výpis 6.3: Generování identifikátoru experimentu

Následující funkce kontroluje přípony souborů. Této funkce je využito při vytváření experimentu a i při doplnění experimentu o další soubory. Kód tento funkce zobrazuje výpis 6.4. Parametrem funkce je pole názvů souborů. V kódu jsou obsaženy dva cykly, které jsou do sebe vnořeny. První z nich prochází jednotlivě soubory. Druhý vnořený pak slouží pro kontrolu přípony souboru. Povolené přípony jsou uloženy v seznamu s názvem `file_extensions`. Tato funkce vrátí `True` při správnosti přípon všech souborů. V opačném případě je návratová hodnota `False`.

```
def check_file_extensions(files):
    correct_suffix = False
    for file in files:
        for i in range(len(file_extensions)):
            if file.name.endswith(file_extensions[i]):
                correct_suffix = True
                break
        if not correct_suffix:
            return False
        correct_suffix = False
    return True
```

Výpis 6.4: Kontrola přípon souborů

Výpis 6.5 slouží pro otevření experimentu datového standardu NIX. První řádka tohoto kódu přiřadí proměnné `directory` cestu k aktuálnímu adresáři. Následující řádka poté definuje cestu k požadovanému souboru. Je zde využito proměnné `experiment_id` a `file_name`. První z nich určí adresář experimentu, ze kterého chceme vybrat soubor. Obsahem druhé proměnné je název souboru. Poslední řádka kódu poté otevře soubor v režimu jen pro

čtení. To nám následně umožní číst data a metadata tohoto tohoto souboru. Tento úsek kódu je z funkce `open_nix_file(experiment_id, file_name)`.

```
directory = os.path.dirname(__file__)
path = os.path.join(directory, '../media/experiments/'
                    + experiment_id + '/' + file_name)
file = nix.File.open(path, nix.FileMode.ReadOnly)
```

Výpis 6.5: Otevření NIX souboru

Uvnitř výpisu 6.6 je zobrazena funkce, která vrací seznam NIX souborů. Tyto soubory jsou vybrány z adresáře na základě parametru této funkce. Třetí řádka kódu přiřadí do proměnné `all_files` seznam všech souborů, které se nacházejí v adresáři. Mohou zde být i soubory s příponou jsonld. Součástí kódu jsou dva vnořené cykly do sebe. Ty mají za úkol vybrat z proměnné `all_files` jenom NIX soubory dle přípony. Ty jsou přidávány postupně do seznamu `nix_files`. Předposlední řádka poté provede abecední seřazení získaných NIX souborů.

```
def get_nix_files(experiment_id):
    fs = FileSystemStorage()
    all_files = fs.listdir('experiments/'
                    + experiment_id + '/')[1]
    nix_files = list()
    for file in all_files:
        for i in range(len(file_extensions)):
            if file.endswith(file_extensions[i]):
                nix_files.append(file)
                break
    nix_files.sort()
    return nix_files
```

Výpis 6.6: Vybrání NIX souborů z adresáře experimentu

Další vybraná funkce (výpis 6.7) slouží pro zobrazení správy experimentu. Parametr `request` obsahuje metadata požadavku na server. Druhým parametrem je identifikátor experimentu. Na začátku kódu je ověřena existence experimentu pomocí podmínky. Když nebude experiment existovat, tak se zobrazí stránka, která o tom informuje (šablona 404.html). V případě existence experimentu dojde vykreslení šablony pro správu experimentu na základě definovaného kontextu.

```
def show_experiment_page(request, id):
    if not utils.experiment_exists(id):
        return render(request, '404.html')

    return render(request, 'nix/experiment.html', {
      'experiment_id': id,
      'transformed_files': utils.get_json_ld_files(id),
      'files': utils.get_nix_files(id)
    })
```

Výpis 6.7: Funkce pro zobrazení správy experimentu

Všechny funkce webové aplikace a jednotlivé jejich úseky kódu jsou komentovány. Poslední funkce, která bude zde bude uvedena, je ze souboru `convert.py`. Více jich tu nebylo z tohoto souboru zobrazeno z důvodu rozsáhlosti kódu. Tato poslední funkce zpracovává metadata experimentu a převádí je do formátu JSON-LD (výpis 6.8). Parametrem této funkce je NIX soubor. První podmínka kódu ověřuje existenci bloků uvnitř souboru. Při jejím splnění se zavolá funkce `parse_blocks(blocks)`. Ta zpracuje metadata datové části experimentu. Druhá podmínka ve výpisu ověřuje existenci sekcí. Při splnění podmínky dojde postupně k rekurzivnímu průchodu jednotlivých sekcí a zpracování obsahu jejich metadat. Druhý parametr rekurzivní funkce má charakterizovat vnoření sekcí a jejich vlastností. Tohoto čísla je poté využito při formátování zápisu metadat do formátu JSON-LD.

```
def parse_metadata(nix_file):
    if nix_file.blocks is not None:
        parse_blocks(nix_file.blocks)
    if nix_file.sections is not None:
        for i in range(len(nix_file.sections)):
            content = recursive_section_search(
                            nix_file.sections[i], 1)
            add_content(content)
```

Výpis 6.8: Zpracování metadat NIX souboru

7 Ověření výsledků

Tato kapitola se zabývá prioritně ověřením výsledku převodu metadat experimentů datového standardu NIX. Součástí této kapitoly je i ověření správnosti ostatních funkcionalit webové aplikace. Při testování aplikace byly použity experimenty z repozitáře https://gin.g-node.org/jsedivy. Je zde sedm charakteristicky rozdílných typů experimentů. Každý z těchto typů experimentů obsahuje několik NIX souborů. Jeden NIX soubor reprezentuje právě jeden experiment. Pro testování funkcionalit webové aplikace byly vybrány všechny dostupné NIX soubory. Jejich počet byl 360.

Při testování byl pro každý typ experimentu ze zmíněného repozitáře vytvořen jeden experiment. Při vytvoření každého z nich došlo k nahrání odpovídajících NIX souborů jeho typu. U těchto experimentů došlo pomocí tlačítka `Conver All` k převodu metadat všech nahraných NIX souborů do formátu JSON-LD každého experimentu. U okolo 50 vybraných převedených souborů došlo k jejich smazání a byl zopakován jejich převod pomocí tlačítka `Convert`. Při převodu metadat experimentů do formátu JSON-LD nedošlo k žádné chybě. Taktéž bylo provedeno testování stažení 30 souborů, které opět proběhlo v pořádku.

Dále byl vytvořen nový experiment pro testování zatížení webové aplikace. Při tomto testu došlo k nahrání všech zmíněných NIX souborů (celkem 360) najednou. Následně poté byl proveden hromadný převod metadat těchto experimentů najednou. Opět i v tomto případě proběhlo vše v pořádku. Aplikace se ukázala jako použitelná pro správu většího množství NIX souborů. Převod metadat všech 360 NIX souborů do formátu JSON-LD trval okolo 8 minut.

Od testování vytvoření experimentu, nahrání NIX souborů a dalších funkcionalit se dostáváme momentálně k ověření výsledků převedených metadat experimentů do formátu JSON-LD. Byly provedeny kontroly, které mají zaručit správnost tohoto procesu převodu. První z nich ověřila, že došlo k převodu všech potřebných metadat. Při této kontrole došlo u každého typu experimentu k výběru čtyř experimentů. Dohromady tedy bylo vybráno 28 NIX souborů. Pro každý z nich byl vypsán obsah jeho metadat pomocí funkce `pprint()` z Nixio knihovny. Ten byl vizuálně kontrolován s obsahem převedeného souboru. Dále byl využit program HDFView (verze 3.1.0) pro zobrazení obsahu metadat 10 z vybraných NIX souborů. To mělo především ověřit, že funkce z Nixio knihovny vypisuje obsah všech metadat správně. Při této kontrole se ukázalo, že rekurzivní průchod jednotlivých sekcí meta-

dat funguje správně. Převedené soubory taktéž obsahovaly všechna potřebná metadata bloků experimentů. Výpis 7.1 zobrazuje část metadat jednoho ze souborů po použití funkce `pprint()`.

```
Session metadata [nix.metadata.session]
      |- odML date: ('2015-07-16',)
      |- odML version: ('1',)
    Experiment [experiment]
        |- Private Experiment: (False,)
        |- ProjectName: ('AUDITORY ERP PROTOCOL...',)
    Environment [environment]
        |- RoomTemperature: ('23',)
    Subject [subject]
        |- Gender: ('F',)
        |- Age: ('8y 7m',)
    Weather [environment]
        |- Weather: ('Sunny',)
        |- Description: ('Sunny weather',)
```

Výpis 7.1: Ukázka výpisu metadat NIX souboru funkcí pprint()

Dále se dostáváme k ověření vyhledávání v metadatech. Na okolo 100 NIX souborech ze 360 byl proveden alespoň jeden SPARQL dotaz. Výpis 7.2 znázorňuje nejpoužívanější z nich. Důvodem použití dotazu nad více soubory bylo především ověření správnosti zápisu metadat ve formátu JSON-LD. Při zpracovávání dotazu se totiž nejprve z JSON-LD souboru vytváří RDF graf. V případě, že by nebyl JSON-LD soubor úspěšně zpracován, tak se vypíše chybová hláška, která o tom informuje. I při sebemenší chybě ve struktuře JSON-LD souboru může dojít k nevytvoření RDF grafu. Proto bylo nutné ověřit to na více experimentech. Při tomto testování se prokázalo, že dochází ke správnému zápisu struktury metadat experimentů do formátu JSON-LD.

```
SELECT ?subject ?predicate ?object
WHERE {?subject ?predicate ?object}
```

Výpis 7.2: Dotaz pro vyhledání všech trojic RDF grafu

U 20 vybraných experimentů bylo poté provedeno větší množství různých dotazů pro ověření výsledků. Teď si pár z nich ukážeme na experimentu `Experiment 149 - driver_s attention and sleep deprivation.nix`.

```
SELECT ?object
WHERE
{?subject <http://example.com/eeg/fullName> ?object}
```

Výpis 7.3: Dotaz pro vyhledání jmen experimentátorů

Výpis 7.3 obsahuje zápis dotazu pro vyhledání jmen všech experimentátorů. Tabulka 7.1 zobrazuje výsledky pro tento dotaz. Výsledkem je nalezení dvou experimentátorů.

fullName
Ing. Roman Mouček, Ph.D.
Ing. Pavel Mautner, Ph.D.

Tabulka 7.1: Výsledky pro vyhledání jmen experimentátorů

Další dotaz obsahuje výpis 7.4. Slouží pro nalezení role vybraného experimentátora. Pro definované jméno experimentátora v tomto dotazu byl vrácen výsledek `Experimenter`.

```
PREFIX eeg: <http://example.com/eeg/>
SELECT ?role
WHERE
{
  ?s eeg:fullName "Ing. Roman Moucek, Ph.D." .
  ?s eeg:role ?role
}
```

Výpis 7.4: Dotaz pro vyhledání role experimentátora

Výpis 7.5 obsahuje dotaz pro nalezení modelu použitého elektrogelu. Výsledkem byla získaná hodnota `ELECTRO-GEL NET:16`.

```
PREFIX eeg: <http://example.com/eeg/>
SELECT ?model
WHERE
{
  ?s eeg:electroGel ?o .
  ?o eeg:model ?model
}
```

Výpis 7.5: Dotaz pro vyhledání modelu elektrogelu

Dále by šlo uvést další množství dotazů pro ověření výsledků. Výpis 7.6 obsahuje poslední. Tento dotaz slouží pro nalezení informací o vybraném kanálu. Tabulka 7.2 obsahuje získané výsledky pro tento dotaz. Je zde zobrazen soupis vlastností a hodnot vybraného kanálu.

```
PREFIX eeg: <http://example.com/eeg/>
SELECT ?p ?o
WHERE
{
  ?s eeg:2_14_2_2013Metadata ?settings .
  ?settings eeg:hardwareSettings ?hs .
  ?hs eeg:channels ?channels .
  ?channels eeg:channel-14 ?channel .
  ?channel ?p ?o
}
```

Výpis 7.6: Dotaz pro vyhledání informací o vybraném kanálu

Vlasnost	**Hodnota**
http://example.com/eeg/AIResolution	1e-06
http://example.com/eeg/impedance	0.0
http://example.com/eeg/AIUsedChannelCount	16
http://example.com/eeg/name	Cz
http://example.com/eeg/impedanceLowerBound	0.0
http://example.com/eeg/impedanceUpperBound	100.0

Tabulka 7.2: Vyhledané vlastnosti a hodnoty kanálu

Vytvořená webová aplikace byla testována ve třech webových prohlížečích. Prvním z nich je Chromium verze 83.0.4103.97. Druhým je Mozilla Firefox verze 77.0.1. Posledním je Opera verze 70.0.3728.71. Dále se uvažovalo u webové aplikace nad provedením dalších testů. Bralo se v potaz provedení například jednotkových testů. Nakonec se od dalšího testování aplikace upustilo. Webová aplikace byla dostatečně testována nad všemi dostupnými NIX soubory (experimenty) v rámci neuroinformatické laboratoře. Použití RDF a dotazovacího jazyku SPARQL se ukázalo jako vhodným řešením pro vyhledávání v metadatech experimentů.

8 Závěr

Vytvořená diplomová práce má požadovaný rozsah a došlo k splnění všech bodů jejího zadání. V druhé kapitole byly popsány technologie a jazyky sémantického webu nebo jeho principy. Sémantický web má mnoho zastánců i odpůrců. Jeho komunita se v posledních letech ustálila. Její členové spíše ubývají než přibývají. Je to z důvodu celkové složitosti použití jednotlivých technologií a jazyků sémantického webu. Nejpoužívanější technologie z nich je RDF. Tato technologie a dotazovací jazyk SPARQL byly vybrány pro použití při implementaci aplikace.

V třetí kapitole byly popsány vybrané datové standardy pro elektrofyziologii. Na základě této kapitoly byl v analýze aplikace popsán jejich souhrn a došlo k definování kritérií pro výběr jednoho z nich. Vybraným datovým standardem pro implementaci byl NIX. Ve čtvrté kapitole byla popsána neuroinformatická laboratoř Fakulty aplikovaných věd Západočeské univerzity v Plzni. Bylo zde popsáno její vybavení, portál EEGBase, provedené experimenty, datové formáty, modely a další informace.

Pátá kapitola této práce se zabývala analýzou a návrhem webové aplikace. Byla zde popsána specifikace požadavků. Kromě výběru datového standardu NIX zde také došlo k výběru webového frameworku Django. Tato kapitola také obsahuje vybrané knihovny, řešení převodu metadat experimentů do formátu JSON-LD a popis rozvržení webové aplikace.

Na základě páté kapitoly došlo k implementaci webové aplikace. Tato aplikace umožňuje vytvářet experimenty. Jeden vytvořený experiment může spravovat více nahraných experimentů najednou. Nahrané soubory NIX jdou stahovat, smazat nebo rozšířit o další. Součástí je i funkcionalita převodu metadat do formátu JSON-LD. Lze převést experimenty hromadně nebo jednotlivě. Soubory s převedenými metadaty lze stáhnout nebo smazat. Dále je umožněno zobrazit obsah převedených metadat experimentů nebo vyhledávat v nich pomocí dotazovacího jazyku SPARQL. Kapitola šest popisuje implementaci této webové aplikace. Taktéž byla vytvořena dokumentace této aplikace.

Sedmá kapitola se zabývá ověřením výsledků. Aplikace byla testována na sedmi charakteristicky rozdílných typech experimentů. Bylo vybráno 360 experimentů z repozitáře https://gin.g-node.org/jsedivy. Na těchto experimentech byly testovány jednotlivé funkcionality aplikace. Použití RDF a dotazovacího jazyku SPARQL se ukázalo jako vhodným řešením pro vyhledávání v metadatech experimentů.

Přehled zkratek

API - Application Programming Interface

BIDS - Braing Imaging Data Structure

CSS3 - Cascading Style Sheets 3

ECoG - Elektrokortikografie

EDF - European Data Format

EEG - Elektroencefalografie

ERP - Event-related potential

fMRI - Funkční magnetická rezonance

HDF - Hierarchical Data Format

HTML - Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

INCF - International Neuroinformatics Coordinating Facility

IRI - Internationalized Resource Identifier

JSON-LD - JavaScript Object Notation for Linked Data

MEG - Magnetoencefalografie

NEMO - Neural ElectroMagnetic Ontology

NIF - Neuroscience Information Framework

NIX - Neuroscience Information Exchange Format

NS - Namespace

NWB:N - Neurodata Without Borders: Neurophysiology

OBI - Ontology for Biomedical Investigations

odML - Open Metadata Markup Language

OEN - Ontology for Experimental Neurophysiology

OWL - Web Ontology Language

RDF - Resource Description Framework

RDFa - Resource Description Framework in attributes

RDFS - Resource Description Framework Schema

REST - Representational State Transfer

RQL - Relationship Query Language

SOAP - Simple Object Access Protocol

SPARQL - SPARQL Protocol and RDF Query Language

TriX - Triples in XML

Turtle - Terse RDF Triple Language

UCS - Universal Coded Character Set

URI - Uniform Resource Identifier

XML - Extensible Markup Language

Literatura

[1] *1. Format Overview* [online]. 2020. [cit. 2020/04/20]. Dostupné z: https://nwb-schema.readthedocs.io/en/stable/format.html.

[2] *1. Overview* [online]. [cit. 2020/04/20]. Dostupné z: https://nwb-schema.readthedocs.io/en/stable/format_description.html.

[3] *About NWB* [online]. [cit. 2020/04/18]. Dostupné z: https://www.nwb.org/about-nwb/.

[4] *Analysis and Visualization Tools* [online]. [cit. 2020/04/18]. Dostupné z: https://www.nwb.org/tools/.

[5] Appelhoff, S. – Suleman, D. S. – Gill, S. *Matching Pennies: A Brain Computer Interface Implementation Dataset* [online]. 2019. [cit. 2020/04/18]. Dostupné z: https://osf.io/cj2dr.

[6] Birckley, D. – Guha, R. V. – McBride, B. *RDF Schema 1.1* [online]. 2014. [cit. 2020/03/08]. Dostupné z: https://www.w3.org/TR/rdf-schema/.

[7] Cardiff, J. The Evolution of the Semantic Web. 01 2009, 534.

[8] *Difference between MVC and MVT design patterns* [online]. [cit. 2020/08/08]. Dostupné z: https://www.geeksforgeeks.org/difference-between-mvc-and-mvt-design-patterns/.

[9] *Django Tutorial Part 5: Creating our home page* [online]. [cit. 2020/08/08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Home_page.

[10] *API Documentation for Data* [online]. [cit. 2020/07/15]. Dostupné z: https://nixpy.readthedocs.io/en/latest/api_data.html.

[11] *Installation Guidelines* [online]. [cit. 2020/07/15]. Dostupné z: https://nixpy.readthedocs.io/en/latest/install.html.

[12] *API Documentation for Metadata* [online]. [cit. 2020/07/15]. Dostupné z: https://nixpy.readthedocs.io/en/latest/api_metadata.html.

[13] *Overview* [online]. [cit. 2020/07/15]. Dostupné z: https://nixpy.readthedocs.io/en/latest/overview.html.

[14] Grewe, J. *About* [online]. 2015. [cit. 2020/04/08]. Dostupné z: https://github.com/G-Node/nix/wiki/About.

[15] Grewe, J. *Home* [online]. 2019. [cit. 2020/04/09]. Dostupné z: https://github.com/G-Node/nix/wiki.

[16] Grewe, J. *Model Definition* [online]. 2017. [cit. 2020/04/10]. Dostupné z: https://github.com/G-Node/nix/wiki/Model-Definition.

[17] Grewe, J. *The Model* [online]. 2015. [cit. 2020/04/15]. Dostupné z: https://github.com/G-Node/nix/wiki/The-Model.

[18] Grewe, J. *Libraries & tools* [online]. 2019. [cit. 2020/04/14]. Dostupné z: https://nixio.readthedocs.io/en/latest/ecosystem.html.

[19] *HDF5* [online]. 2020. [cit. 2020/04/08]. Dostupné z: https://portal.hdfgroup.org/display/HDF5.

[20] Klyne, G. – Carroll, J. J. – McBride, B. *RDF 1.1 Concepts and Abstract Syntax* [online]. 2014. [cit. 2020/03/08]. Dostupné z: https://www.w3.org/TR/rdf11-concepts/.

[21] Koivunen, M.-R. – Miller, E. *W3C Semantic Web Activity* [online]. 2001. [cit. 2019/12/28]. Dostupné z: https://www.w3.org/2001/12/semweb-fin/w3csw.

[22] Malvi, K. *Top 30 Python Web Frameworks to Use in 2020* [online]. 2020. [cit. 2020/06/30]. Dostupné z: https://www.mindinventory.com/blog/best-python-web-frameworks-2019/.

[23] McGuinness, D. L. – van Harmelen, F. *OWL Web Ontology Language Overview* [online]. 2004. [cit. 2019/12/29]. Dostupné z: https://www.w3.org/TR/owl-features/.

[24] Mouček, R. et al. Event-related potential data from a guess the number brain-computer interface experiment on school children. *Scientific data*. 2017, 4, 1, s. 1–11.

[25] Mouček, R. et al. Software and hardware infrastructure for research in electrophysiology. *Frontiers in Neuroinformatics*. 2014, 8, s. 20. ISSN 1662-5196. doi: 10.3389/fninf.2014.00020.

[26] *Coordination disorder in children* [online]. NEUROINFORMATICS research group, 2015. [cit. 2020/06/15]. Dostupné z: http://neuroinformatics.kiv.zcu.cz/articles/read/coordination-disorder-in-children_2015-01-20.

[27] *Driver's attention* [online]. NEUROINFORMATICS research group, 2015. [cit. 2020/06/15]. Dostupné z: http://neuroinformatics.kiv.zcu.cz/articles/read/drivers-attention_2015-01-20.

[28] *NIX data model* [online]. [cit. 2020/03/23]. Dostupné z: https://nixio.readthedocs.io/en/latest/data_model.html.

[29] *odML Terminologies* [online]. [cit. 2020/04/15]. Dostupné z: https://terminologies.g-node.org/v1.1/terminologies.xml.

[30] *Organization of Schemas* [online]. [cit. 2020/07/02]. Dostupné z: https://schema.org/docs/schemas.html.

[31] *OWL 2 Web Ontology Language Document Overview* [online]. 2012. [cit. 2020/03/07]. Dostupné z: https://www.w3.org/TR/owl2-overview/.

[32] Pernet, C. R. et al. EEG-BIDS, an extension to the brain imaging data structure for electroencephalography. *Scientific data*. 2019, 6, 1, s. 1–5. doi: 10.1038/s41597-019-0104-8.

[33] Prud'hommeaux, E. – Seaborne, A. *SPARQL Query Language for RDF* [online]. 2008. [cit. 2020/03/18]. Dostupné z: https://www.w3.org/TR/rdf-sparql-query/.

[34] *RDFLib 5.0.0* [online]. [cit. 2020/07/08]. Dostupné z: https://rdflib.readthedocs.io/en/stable/.

[35] *Getting started with RDFLib* [online]. [cit. 2020/07/08]. Dostupné z: https://rdflib.readthedocs.io/en/stable/gettingstarted.html.

[36] *Resource Description Framework* [online]. 2014. [cit. 2020/03/08]. Dostupné z: https://www.w3.org/RDF/.

[37] *Schema.org Extensions* [online]. [cit. 2020/07/02]. Dostupné z: https://schema.org/docs/extension.html.

[38] Smith, M. K. – Welty, C. – McGuiness, D. L. *OWL Web Ontology Language Guide* [online]. 2004. [cit. 2019/12/29]. Dostupné z: https://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[39] *The NWB Software Ecosystem* [online]. [cit. 2020/04/18]. Dostupné z: https://www.nwb.org/nwb-software/.

[40] *The NWB:N Data Standard* [online]. [cit. 2020/04/18]. Dostupné z: https://www.nwb.org/nwb-neurophysiology/.

[41] Vaněk, J. – Mouček, R. On data and medatata formats for electrophysiological experiments. 2015, s. 290–295.

[42] *Welcome to Schema.org* [online]. [cit. 2020/07/02]. Dostupné z: `https://schema.org/`.

[43] *XML RDF* [online]. [cit. 2020/03/09]. Dostupné z: `https://www.w3schools.com/xml/xml_rdf.asp`.

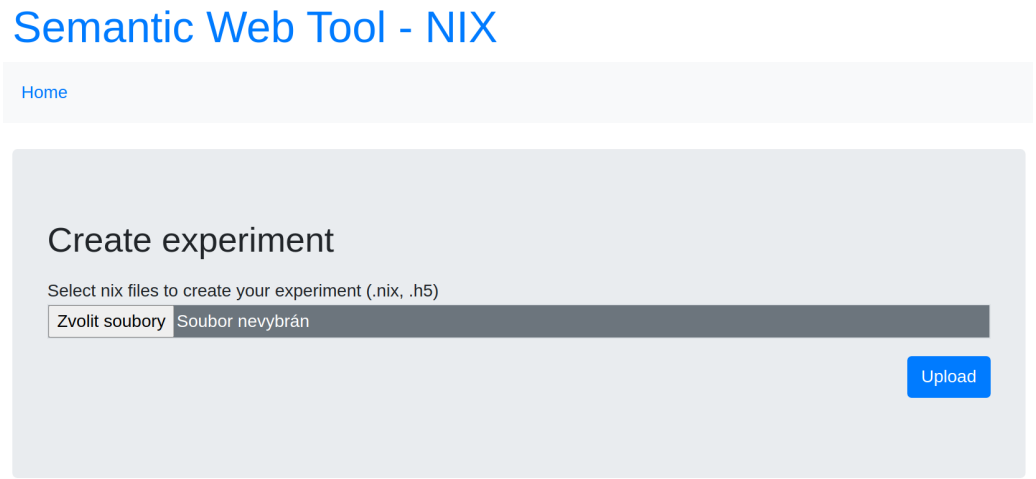
Příloha

A Dokumentace

Tato sekce přílohy popisuje funkcionality vytvořené webovou aplikací. Bude zde popsáno vytvoření experimentu, jeho správa, zobrazení metadat nahraných souborů a vyhledávání v metadatech.

A.1 Vytvoření experimentu

Na obrázku 1 je zobrazena hlavní stránka aplikace. Ta slouží k vytvoření experimentu. Kliknutím na tlačítko `Zvolit soubory` uživatel vybere soubory, které mají být nahrané v rámci vytvoření experimentu. Lze nahrát jeden nebo více experimentů datového standardu NIX s příponou nix nebo h5. To umožní nahranou sadu experimentů spravovat hromadně. Pro úspěšné zpracování formuláře musí být vždy vybrán alespoň jeden soubor. Formulář se odešle kliknutím na tlačítko `Upload`. Pří úspěšném jeho zpracování dojde k vytvoření experimentu a k přesměrování na jeho správu.

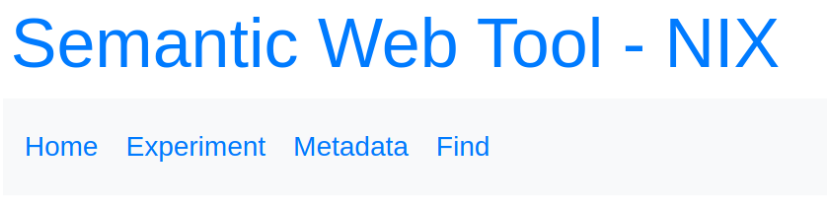


Obrázek 1: Stránka pro vytvoření experimentu

A.2 Navigační lišta

Obrázek 2 zobrazuje hlavní navigační lištu aplikace. Při zobrazení hlavní stránky je vidět jen odkaz `Home`. Ten odkazuje na stránku pro vytvoření experimentu. Další odkazy v navigační liště se zobrazí až následně po vytvo-

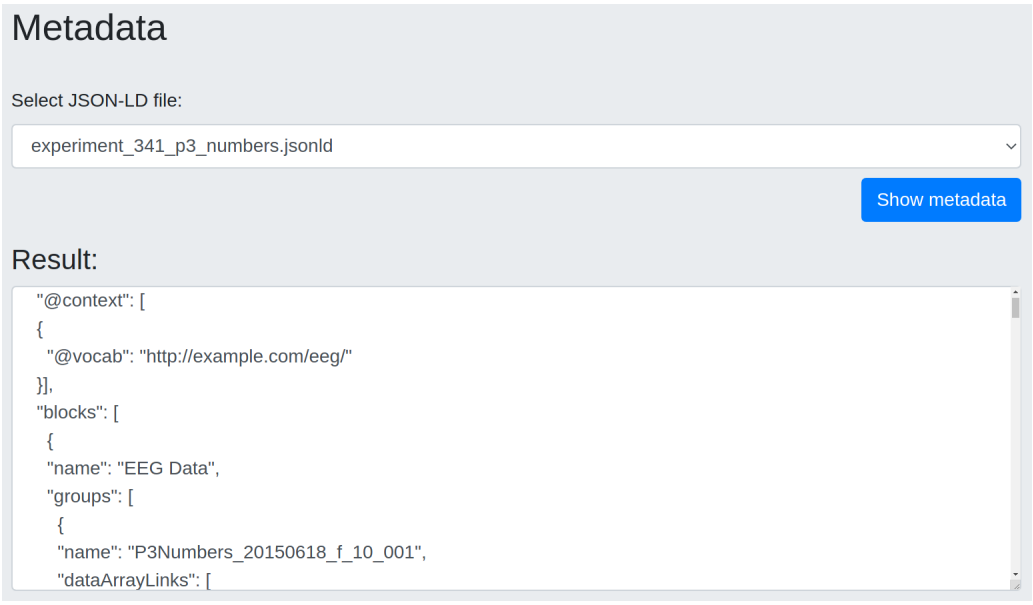
ření experimentu. Kliknutím na odkaz `Experiment` se zobrazí stránka, která umožňuje spravovat experiment. Stránka pro zobrazení metadat převedených souborů ve formátu JSON-LD se otevře kliknutím na odkaz `Metadata`. Kliknutím na odkaz `Find` dojde k přesměrování na stránku pro vyhledávání v metadatech experimentů.



Obrázek 2: Hlavní navigační lišta aplikace

A.3 Zobrazení metadat experimentu

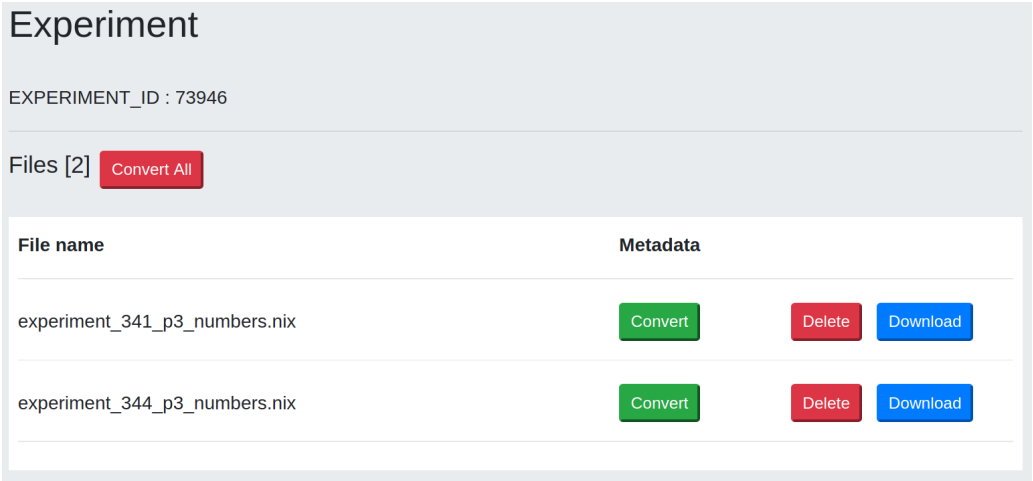
Na obrázku 3 je vidět stránka pro zobrazení metadat experimentů. Je zde výběrové pole. To slouží pro vybraní souboru, jehož obsah chceme zobrazit. Kliknutím na tlačítko `Show metadata` se provede zpracování formuláře. Metadata vybraného souboru se následně zobrazí pod tímto tlačítkem v textovém poli. Pro umožnění zobrazení metadat musí být alespoň jeden NIX soubor experimentu převeden do formátu JSON-LD.



Obrázek 3: Stránka pro zobrazení metadat

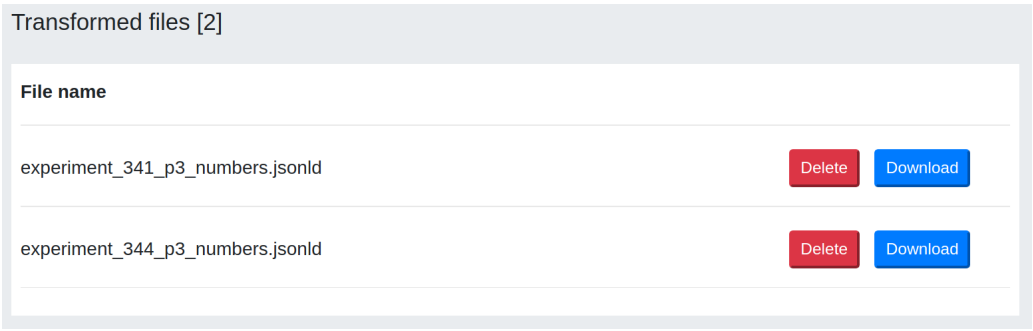
A.4 Správa experimentu

Kliknutím na odkaz `Experiment` v hlavní navigační se zobrazí správa experimentu. Na obrázku 4 je zobrazena tabulka nahraných NIX souborů. Jednotlivé soubory v této tabulce lze stáhnout nebo smazat. Součástí každého souboru je tlačítko `Convert` pro převedení metadat do formátu JSON-LD. Na obrázku je zobrazeno i tlačítko `Convert All`. To umožňuje převést metadata všech souborů v této tabulce najednou. Tato stránka zobrazuje i vygenerované identifikační číslo experimentu.



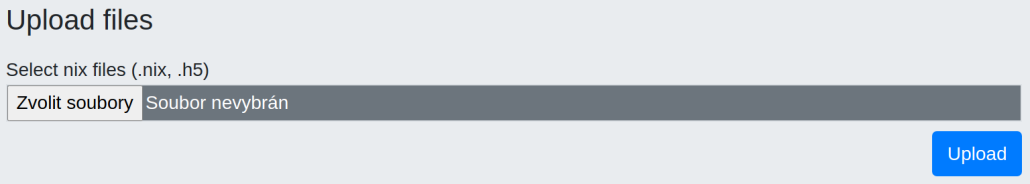
Obrázek 4: Seznam nahraných souborů experimentu

Obrázek 5 zobrazuje tabulku převedených metadat experimentů do formátu JSON-LD. Tyto soubory lze v této tabulce stáhnout nebo smazat.



Obrázek 5: Seznam souborů s převedenými metadaty

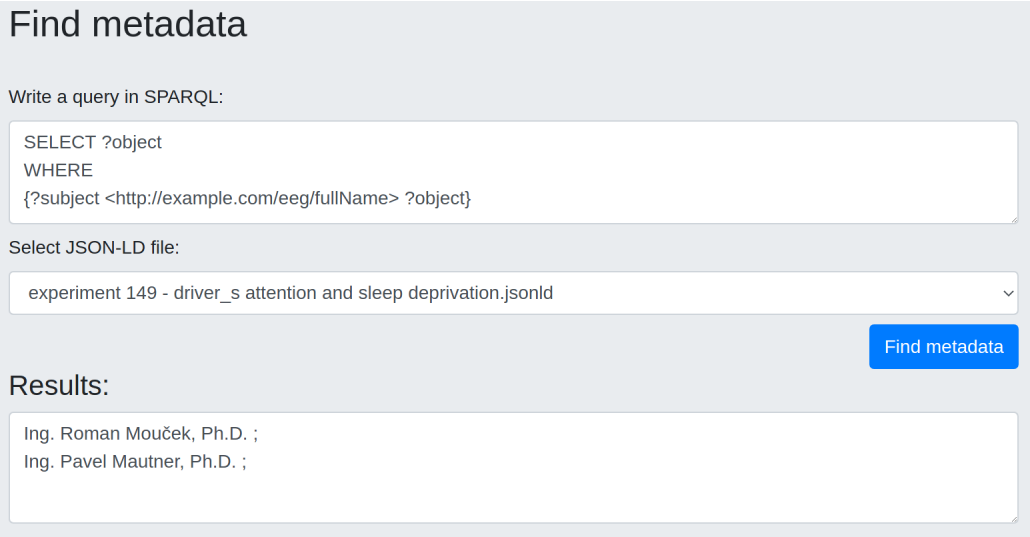
Správa experimentu umožňuje k experimentu přidávat další NIX soubory. Tato funkcionalita je zobrazena na obrázku 6. Po kliknutí na tlačítko `Zvolit soubory` si lze vybrat soubory k nahrání. K odeslání formuláře dojde po kliknutí na tlačítko `Upload`.



Obrázek 6: Přidání nových NIX souborů

A.5 Vyhledávání v metadatech experimentu

Poslední funkcionalita této webové aplikace umožňuje vyhledávat v metadatech. Je obsažena na stránce pod odkazem `Find`. Tuto stránku zobrazuje obrázek 7. Je zde textové pole pro napsání dotazu v jazyce SPARQL. Součástí je výběrové pole pro vybrání experimentu, nad kterým chceme dotaz provést. Formulář se odešle kliknutím na tlačítko `Find metadata`. Výsledek dotazu se následně zobrazí v textovém poli pod tímto tlačítkem. Jednotlivé získané záznamy jsou v poli ukončeny znakem středník a odřádkovány. Pro umožnění vyhledávání v metadatech musí být alespoň jeden NIX soubor experimentu převeden do formátu JSON-LD.



Obrázek 7: Stránka pro vyhledávání metadat

B Obsah DVD

Součástí práce je DVD, které obsahuje:

- readme.txt - Obsahuje informace o zprovoznění aplikace.
- dp.pdf - Soubor s textem diplomové práce.
- dptext - Složka, která obsahuje zdrojové soubory práce psané v LaTeXu. Uvnitř této složky je adresář `images`. V tom se nacházejí obrázky.
- webapp - Tento adresář obsahuje složku `dp`. V té se nacházejí všechny zdrojové soubory vytvořené webové aplikace.
- poster - Obsahuje vytvořený poster v programu Microsoft Publisher. Taktéž je zde tento poster uložen v podobě s příponou pdf.