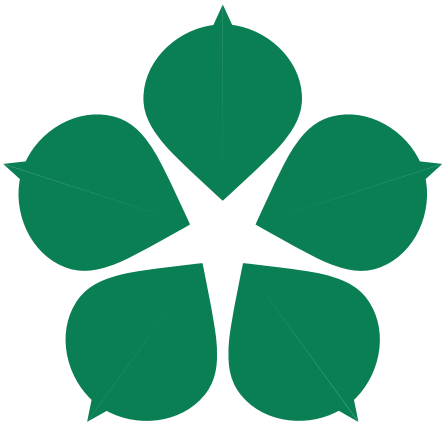
Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta



Vývoj webové aplikace pro navigaci v úložišti podle standardu ASAM ODS

Bakalářská práce

Daniel Kotrč

Školitel práce: Bc. Milan Konzal

Garant práce: Ing. Jiří Jelínek CSc.

České Budějovice 2020

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKE PRÁCE

Student: Daniel Kotrč
(jméno, příjmení, tituly)

Obor – zaměření studia: Aplikovaná Informatika
Katedra/ústav PřF JU: Ústav Aplikované Informatiky

Garant z PřF JU: Ing. Jiří Jelínek CSc., Ústav Aplikované Informatiky
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: Bc. Milan Konzal, [redacted]
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce:
Vývoj webové aplikace pro navigaci v úložišti podle standardu ASAM ODS.

Cíle práce:

1. Odborná rešerše ke standardu ASAM ODS – principy, technické řešení a implementace, příklady užití.
2. Dostupné frontendy pro přístup do backendu ASAM ODS, jejich analýza a porovnání na základě navržených jasných a zejména kvantitativních kritérií.
3. Návrh webové aplikace jako náhrady dostupných frontendů úložiště ASAM ODS (Open Data Services). Návrh bude proveden standardním postupem pro vývoj SW s využitím UML a bude vycházet z jasně definované funkcionality. (Funkcionalita: webová platforma, možnost read only přístupu, přístup pro práci s libovolným platným aplikačním modelem ASAM ODS standardu na úrovni standardních frontendů, zobrazování informací o aplikačním modelu a hodnotách instancí, výběr a zobrazení kompatibilní jednotky s možností přepočtu hodnoty do jiné jednotky (př. °C → °F, in → cm, mV → V), zobrazování grafů jednotky, možnost stahování souborů dle nastaveného výběru.)
4. Implementace a testování aplikace pomocí integračních a jednotkových testů.
5. Pilotní provoz aplikace v organizaci školitele, analýza výsledků a zapracování jejích výstupů do aplikace. Vyjádření organizace školitele k výslednému produktu a jeho použitelnosti v praxi.

Základní doporučená literatura:

- Pavel Herout, Učebnice jazyka Java - 5. vydání, ISBN 978-80-7232-398-2
- Erich Gamma, Návrh programů pomocí vzorů - stavební kameny objektově orientovaných vzorů, ISBN 80-247-0302-5
- Jeff Langr, Pragmatic Unit Testing in Java 8 with JUnit, ISBN 978-1-94122-259-1
- Robin Wieruch, The Road to learn React, ISBN 978-1720043997
- Materiály a dokumentace z knihoven, články z Internetu.

Financování práce .....

Školitel práce M. Konzal .....podpis: .....

U externích vedoucích fakultní garant práce

J. Jelínek .....podpis: .....

Garant oboru bak. studia (nepožaduje se u oboru biologie)

.....podpis: .....

Vedoucí katedry/ústavu PřF JU, kde proběhne obhajoba

R. Vohnout .....podpis: .....

Případný souhlas vedoucího ústavu AV .....podpis: .....

V Českých Budějovicích dne 21. 11. 2019 .....Podpis studenta .....

Bibliografické údaje

Kotrč D., 2020: Vývoj webové aplikace pro navigaci v úložišti podle standardu ASAM ODS. [Development of web application for navigation in storage according to ASAM ODS standard. Bc. Thesis, in Czech] – 42 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Tématem této práce je odborná rešerše Standardu ASAM ODS, jeho popis a využití, průzkum a porovnání již existujících řešení, návrh aplikace jako náhrady již dostupných řešení za pomocí UML a podle předem definované funkcionality. Implementace aplikace, její testování a nasazení aplikace do pilotního provozu.

Annotation

The topic of this thesis is research of the ASAM ODS Standard, description of the standard and its use in practice, research and comparison of existing solutions, design of application as replacement of existing solutions with use of UML and with predefined functionality of application. Implementation, testing and deployment of the application.

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejich internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne …… ……..

Daniel Kotrč

Poděkování

Tímto bych rád poděkoval Bc. Milanu Konzalovi za jeho strávený čas při konzultacích, skvělou podporu, cenné rady nejen k této práci a trpělivost. Dále bych rád poděkoval Ing. Jiřímu Jelínkovi, CSc. za cenné rady, trpělivost a za zastřešení práce pro Přírodovědeckou fakultu Jihočeské univerzity. Uživatelům pilotního nasazení za otestování aplikace a získání zpětné vazby pro vylepšení aplikace a následně také rodině za podporu během mého studia.

Obsah

1 Úvod

Automobilový průmysl se neustále vyvíjí, a to i v případě použití nových softwarových technologií ve výrobních a testovacích procesech. Současné systémy pro testování, vyhodnocování a simulaci dat mají ve většině případů vlastní formáty ukládání dat, které se od sebe velmi liší, pokud jde o popis konfigurace a způsob ukládání výsledků. Každým dnem přicházejí nové firmy a jejich nová softwarové řešení s jinou filozofií, nahrazením jiného systému nebo jiným zpětně nekompatibilním formátem.

Asociace pro Standardizaci Automatizace a Měřících Systémů, zkráceně ASAM [1], se tímto problémem zabývá již několik let. Vyvinuli proto standard ODS (Open Data Services) [2] z důvodu ulehčení, sjednocení a zobecnění formátu ukládání těchto dat nezávisle na architektuře. S tímto standardem by měl být umožněn rychlejší vývoj a snadnější použití v jiných odděleních ve společnosti nebo u dodavatelů. K vytvoření tohoto standardu se spojily přední společnosti v automobilovém průmyslu jako: Audi AG, BMW AG, Porsche AG, MAN Truck & Bus AG, Robert Bosch GmbH, Daimler AG, Volkswagen AG, apod.

Standard popisuje fyzické ukládání informací a rozhraní služeb, která mohou být využívána jakoukoliv součástí testovacích prostředí (testbench) k ukládání nebo načítání dat potřebných pro funkci. Standardním přístupem k rozhraním je rozhraní *CORBA* [3], který vyžaduje objektově orientovaný návrh systému, stará se o transparentní sítový provoz tím že specifikuje jakým způsobem může být rozhraní identifikováno a připojeno k jakékoli součásti, která jej chce využít, avšak nespecifikuje služby rozhraní. Původně první specifikace rozhraní ASAM ODS využívala *RPC [4]* jako komunikační metodu, které je a bude z důvodu kompatibility podporováno, hlavním rozhraním je ale objektově orientované API (OO-API) [5]. V nové verzi standardu (6.0.0) je nově podporováno i HTTP API [6].

ASAM ODS poskytuje společný datový model (Základní model) pro jednoznačné a úplné uložení dat. Tento model přidává datům sémantiku, což umožňuje různým systémům interpretovat data stejným způsobem a lze model tímto považovat za hrubou klasifikaci dat. Pokrývá mnoho oblastí a jejich potřeby, je přizpůsobitelný požadavkům konkrétního systému či projektům díky vytvoření tzv. aplikačního modelu ze základního modelu.

API umožnují přístup k metainformacím skutečného modelu a také přístup k datům kompatibilních systémů a nástrojů. Standard poskytuje také databázový model pro relační databáze, čímž specifikuje fyzické uložení dat a umožňuje výměnu databázových souborů mezi systémy se stejným systémem řízení báze dat (DBMS). V případě že není k dispozici vhodná serverová aplikace ODS, poskytuje snadný přístup prostřednictvím SQL dotazů i na různých platformách a mezi databázemi s různými DBMS [7].

1.1. Cíl práce

Cílem této bakalářské práce je vytvořit webovou aplikaci, která je schopná pracovat s libovolným aplikačním modelem standardu ASAM ODS přistupovat k úložišti prostřednictvím ASAM ODS OO-API, procházet jím, zobrazovat data i metadata modelu, umožnit přístup k souborům uloženým v úložišti, a především zrychlit načítání dat pro lepší práci s daty, jelikož aktuální řešení používané v organizaci školitele je nedostatečně výkonné a v případě přístupu ze vzdálených pracovišť velmi pomalé.

Aplikace bude zahrnovat integrační a jednotkové testy *JUnit [8]*. V organizaci školitele bude proveden pilotní provoz aplikace a zjištěna zpětná vazba uživatelů. Vývoj aplikace je průběžně konzultován s budoucími uživateli aplikace. Aplikace bude obsahovat jednoduché grafické řešení v souladu s nejnovějšími technickými požadavky.

2 Analýza

2.1. Standard ASAM ODS

Zavedení standardu a za největším přínosem pro standard stojí především společnosti *BMW*, *Daimler* a *Porsche*. Na vývoji standardu samozřejmě mají přínos většina větších automobilových společností ve světě.

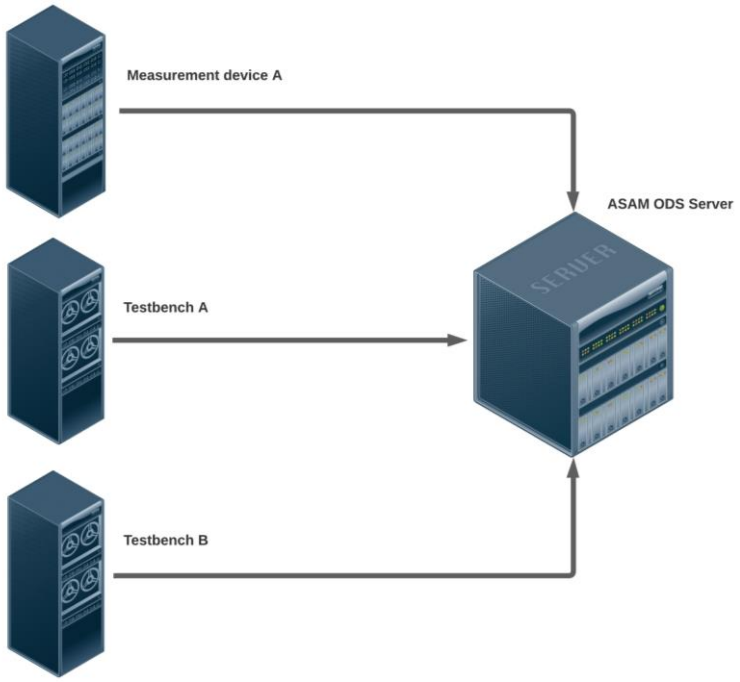
Hlavní síla standardu *ODS* ve srovnání s nestandardizovanými řešení ukládaní dat je v tom, že přístup k datům je nezávislý na architektuře a jeho datový model je velice přizpůsobivý a dobře definovaný pro různé aplikační scénáře.

Základní model se používá jako rodič pro odvození konkrétních aplikačních modelů. Poskytuje hrubou klasifikaci dat, aby mohli klientské nástroje různých dodavatelů správně interpretovat data.

Aplikační modely pokrývají potřeby specifické oblasti aplikace pro ukládání dat. Standard poskytuje předdefinované aplikační modely pro geometrii testovacích objektů, testování *NVH*, kalibraci testbenchů, datové sběrnice a pracovních postupů.

Servery *ODS* fungují jako datová centra (viz. **Obrázek 1**: ODS Server jako Datacenter) pro různé testovací stanice a měřící zařízení od různých dodavatelů. Servery umějí pracovat s hromadnými daty, které jsou uložené v externích souborech (*ASAM* MDF [9]), kde databáze obsahuje pouze popisná metadata ukazatele na toto externí úložiště. K interním i extérním datům lze přistupovat nezávisle na jejich původu pomocí stejných metod a rozhraní

Servery *ODS* jsou navíc škálovatelné, což umožňuje rozšířit datové modely a přidat do celkového řetězce nástrojů více klientů bez nutnosti nastavení nového serveru pro každé rozšíření.

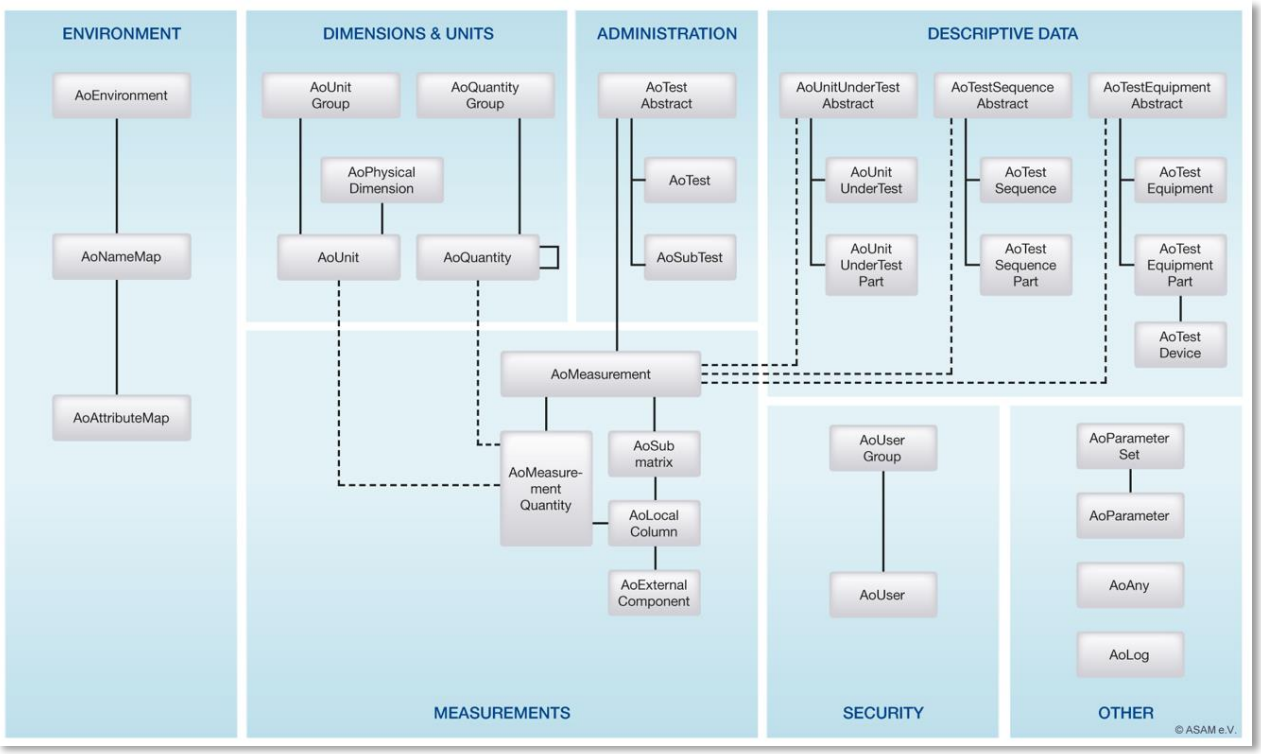


Obrázek 1: ODS Server jako Datacenter

2.1.1. Základní model

Tento model si můžeme představit jako **Entitně-Relační model** databáze. Je to již definovaný model podle standardu, od kterého se rozvíjí Aplikační modely, které jsou vysvětleny v následující kapitole. Díky tomu že je model neměnný, zajišťuje tím pomocí prvků definovanou sémantiku, čímž zajišťuje stejné chápání a možnost čtení a porozumění jakýmkoliv klientem.

Základní model se ve skutečnosti nepoužívá přímo jako model pro databáze, kvůli jeho obecnosti, používá se jako rodič pro odvození konkrétních aplikačních modelů. Model popisuje relace (vztahy) mezi elementy a zajišťuje kategorizaci dat. Relace, které se nacházejí ve stejné kategorizaci dat jsou zobrazené nepřerušovanou černou čarou a relace které ukazují do jiné kategorizace dat jsou zobrazené přerušovanou čarou na **Obrázek 2**: Základní model, Zdroj: [a].

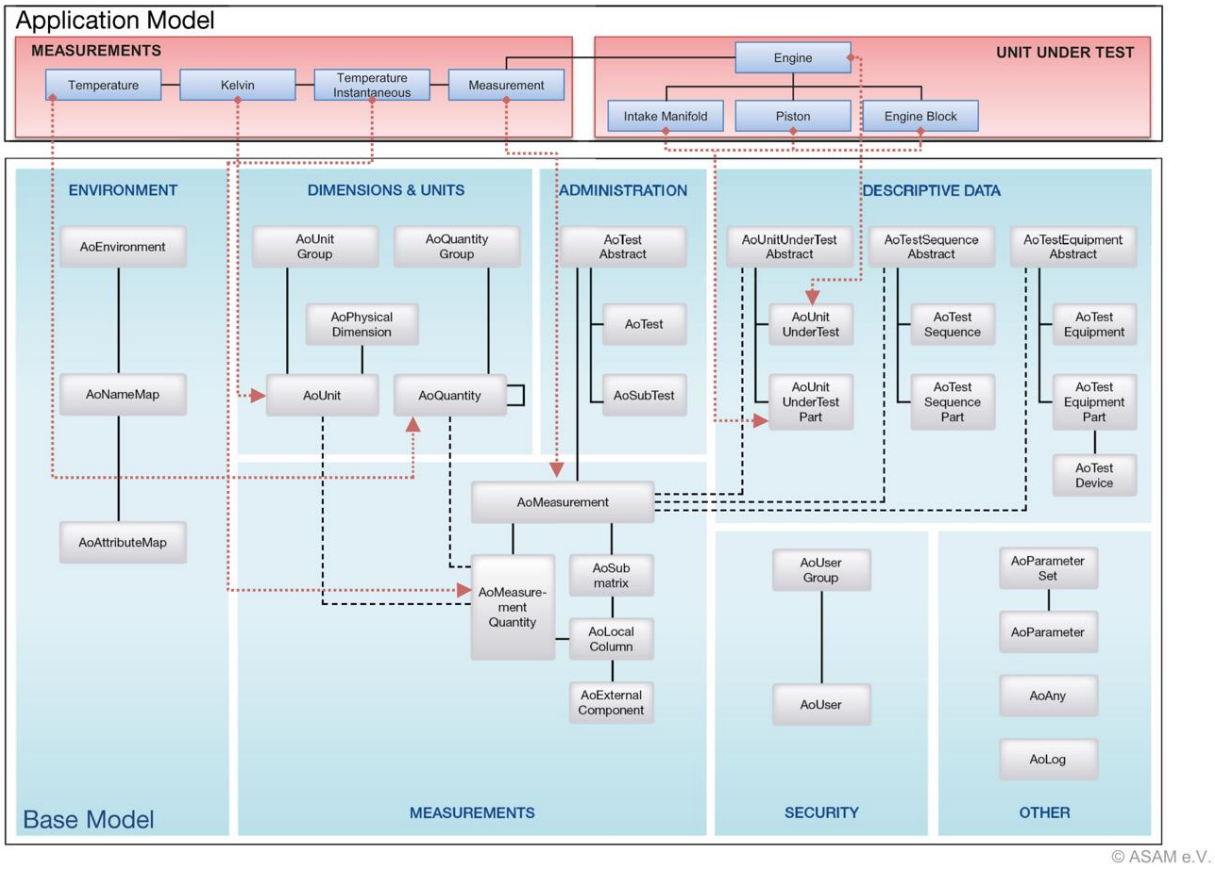


Obrázek 2: Základní model, Zdroj: [a]

2.1.2. Aplikační model

Aplikační model si můžeme představit jako **Relační model** databáze. Aplikační modely pokrývají specifické oblasti použití, jsou odvozeny od základního modelu a definují, které prvky základního modelu jsou v databázi skutečně použity. Každý prvek aplikačního modelu vždy odkazuje na prvek *základního modelu*, čímž přidává význam datům.

Vytvoření aplikačního modelu pro *ODS* znamená mapovat prvky specifické pro aplikaci na prvky základního modelu. V praxi to znamená vytvoření vlastní upravené instance Základního modelu s upravenými či vlastními atributy pro specifické použití organizace. Aplikační modely poskytují pouze meta-informace o struktuře dat.



Obrázek 3: Aplikační model, Zdroj: [b]

Příklad:

- AoUnitUnderTest (Engine)
  - AoUnitUnderTestPart (Engine Block)
    - AoUnitUnderTestPart (Piston)
    - AoUnitUnderTestPart (Intake Manifold)
  - AoMeasurement (Measurement)
    - AoMeasurementQuantity (Temperature Instantenious)
      - AoQuantity (Temperature)
      - AoUnit (Kelvin)

Prostřednictvím relací *aplikačního modelu* dochází k odvození od *základního modelu.* Díky těmto relacím zobrazenými červenou přerušovanou čarou na **Obrázek 3**: Aplikační model, Zdroj: [b] dokážou klientské aplikace porozumět významu konkrétních prvků datového modelu.

Kromě těchto základních relací může designer aplikačního modelu vytvářet nové relace mezi aplikačními prvky podle potřeby. *Aplikační model* může tedy mít další vztahy a atributy, které v základním modelu neexistují.

Standard aktuálně obsahuje 5. předdefinovaných modelů:

1. Test object geometry
2. NVH testing (Noise, Vibration and Harness)
3. BUS data
4. Test stand calibration data
5. Testing workflows

U těchto modelů se očekává že je uživatelé přizpůsobí svým potřebám přidáním položek specifických pro společnost.

Standard je kompatibilní i s modelem ISO 22240 [10] (Model informací o bezpečnosti vozidla nebo *VSIM*) který standardizuje ukládání údajů o zkouškách bezpečnosti vozidla, je ve vlastnictví *ISO* tudíž není zahrnut do standardu.

2.1.3. Datový model

Datový model je pro *ODS* důležitý kvůli určení významu dat a jejich uložení. Uložená data lze správně použít pouze pokud máme dobře definovaný *aplikační model*. Datový model vzniká vytvořením instancí aplikačních elementů z Aplikačního modelu, tím že jsou tyto instance naplněny naměřenými daty nebo metainformacemi. Základní i Aplikační model popisují strukturu uložení dat, reálné hodnoty jsou však uloženy v instancích aplikačních elementů.

Příklad

- AoMeasurement (Provedené měření)
- AoUnitUnderTest (Testovaná jednotka)
- AoTestEquipment (Použité zařízení k testování)
- AoTestSequence (Provedená zkouška)
- AoAny (Účel zkoušky)
- AoUser (Uživatel který provedl zkoušku)

2.1.4. Formát fyzického úložiště

Část standardu, která určuje, jak by měla být relační databáze zpočátku konstruována pro kompatibilní ukládání dat. Zaměřuje se na nejčastěji používanou databázovou technologii v oboru, a to relační databáze.

Kvůli ukládání dat do relačních databází je nutné předem určit které tabulky musí být v databázi k dispozici, jaké sloupce musí obsahovat a jaké tabulky mají být unikátní (používají se jako klíče). Meta data jsou uložena ve třech statických tabulkách:

1. SVCENT
    a. Tabulka elementů aplikačního modelu.
2. SVCATTR
    a. Tabulka atributů aplikačního modelu.
3. SVCREF
    a. Tabulka relací aplikace

Hodnoty instancí aplikačního modelu jsou uloženy v tabulkách instancí.

1. Tabulka instančních atributů
2. Tabulka obousměrných relací (n-m)
3. Tabulka instancí a atributů
4. Tabulky bezpečnostních informací

2.1.5. API

Pro práci s různými datovými modely ODS můžeme využít několik možností rozhraní. Tyto rozhraní umožňují načítat data a metadata standardu, které následně umožní porozumět klientovi jaký aplikační model je využíván.

1. RPC

Nejstarší a první implementované API pro přístup k datům.

2. OO-API

Druhé implementované API založené na architektuře CORBA.

3. HTTP-API

Podporované od verze standardu 6.0.0, umožňuje práci s daty na ODS serveru pomocí http dotazů. **Pro vývoj aplikace nebylo použito**, jelikož v organizaci školitele v době vývoje aplikace **neexistoval systém**, který by podporoval verzi 6.0.0 a vyšší.

2.1.6. Využití

Standard *ODS* se využívá především se systémy automatizovaných testů. Poprvé byl využit v testovacích laboratořích *NVH* a laboratořích specializujících se v nárazových testech, dynamometrií podvozku a pohonných jednotek.

Typickou oblastí využití jsou testbenche pro vstřikovače paliva, katalyzátory, tlumiče a brzdy motorových vozidel, využívají především *ODS* servery pro ukládání nezpracovaných a klasifikačních dat, také naměřená data z klimatických komor, větrných tunelů, zátěžových a vytrvalostních zkoušek či elektrických motorů, baterií a elektroniky. Silnou stránkou *ODS* je možnost sledovat vývoj komponent v průběhu času díky schopnosti kombinovat testovací data s údaji testované jednotky.

ODS se také využívá pro výměnu dat mezi výrobcem a dodavatelem, obvykle pomocí výměny souborů. Obsah *ODS* lze exportovat do souboru formátu *XML*. Soubor obsahuje jak *datový model* tak testovací data.

2.2. Existující řešení

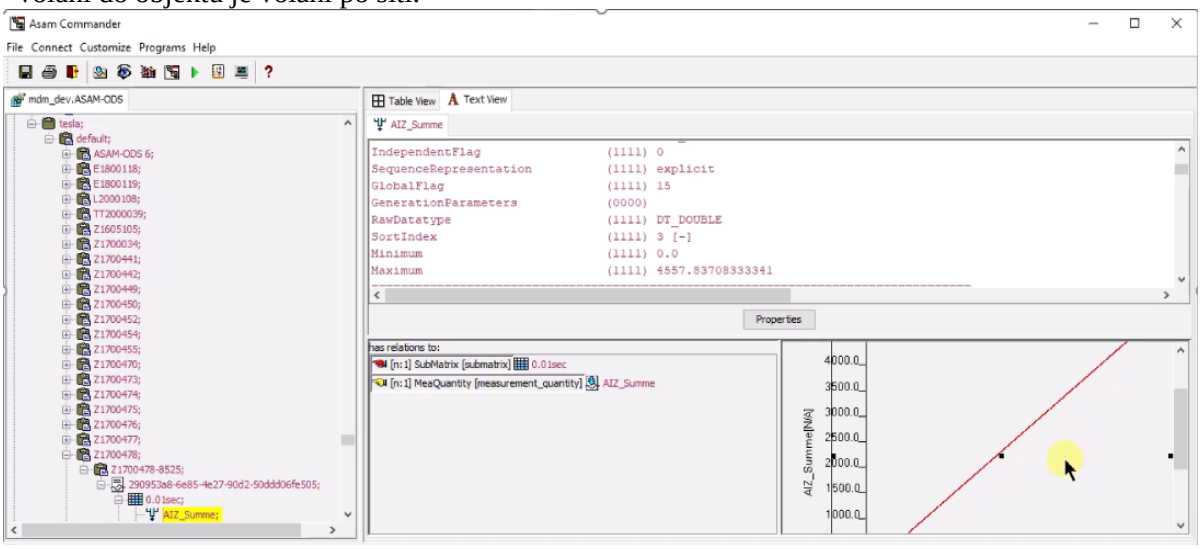
Na trhu již existují aplikace, které dokážou komunikovat se servery ODS. Aplikace převážně využívají rozhraní CORBA, většina z nich jsou placené, nativní a jejich grafické rozhraní odpovídá standardním rozhraním vytvořených v minulém desetiletí. Umějí procházet strukturu ODS modelu.

Aplikace	**Úprava dat**	**Více připojení**	**Vyhledávání**	**Webová aplikace**	**Zdarma**
HighQSoft ASAM Commander	ANO	ANO	NE	NE	NE
UniPlot Addon Corba	ANO	NE	NE	NE	NE
UniPlot Addon DataFinder	ANO	ANO	NE	NE	NE
openMDM	NE	NE	ANO	ANO	ANO
META ASAM ODS Browser	NE	NE	ANO	NE	NE
Mé řešení	NE	ANO	ANO	ANO	ANO

Tabulka 1: Existující řešení

2.2.1. HighQSoft ASAM Commander

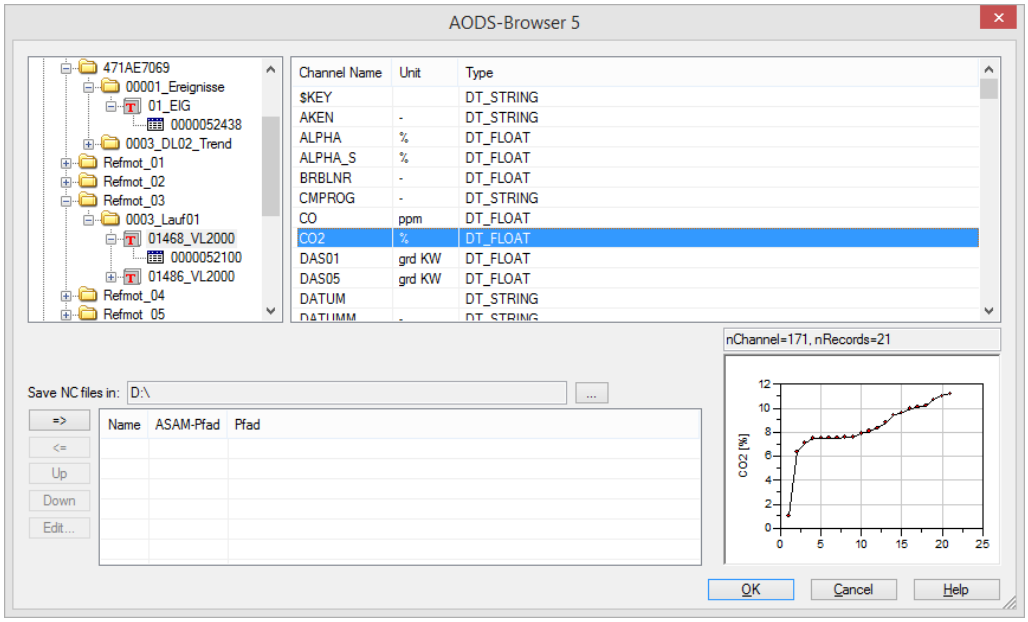
Nástroj, který je součástí serverové aplikace *Avalon Server Suite [11]*, kterou vyvíjí společnost *HighQSoft [12]*. Možnost instalace offline při stažení *Avalon Server Suite* nebo jako online verze aplikace *Java Web Start (.jnpl)[13]* která umí komunikovat pouze se servery *ODS* které mají stejnou verzi a nelze otevírat lokální soubory. Aplikace je velmi pomalá a má zastaralé grafické rozhraní. Využívá OO-API za použití *iterátorových volání* [14], kde výsledkem jsou hromadná data. Tyto data, postupně prochází pomocí iterátoru a následně je filtruje. Funcionalita je pomalá především kvůli tomu že skoro každá interakce v aplikaci je volání do objektu, ale každé volání do objektu je volání po síti.



Obrázek 4: HighQSoft ASAM Commander

2.2.2. UniPlot Addon Corba

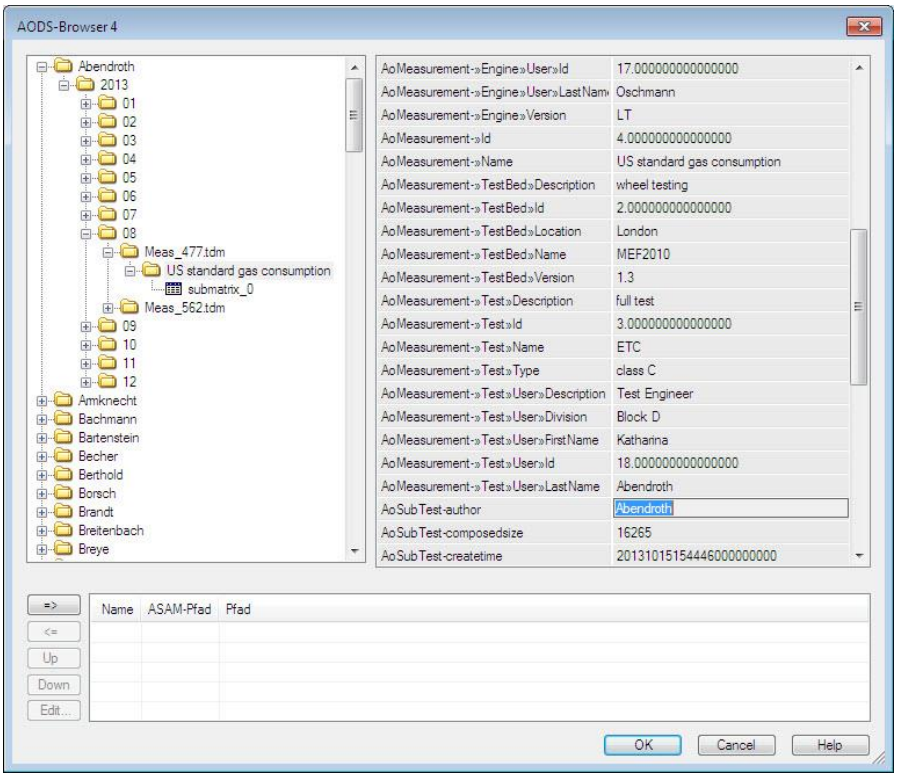
Rozšíření pro aplikaci *UniPlot [15]* využívající rozhraní *CORBA*. Dokáže se připojit pouze k jedné databázi současně. Uživatel může data, jak zobrazit, tak i upravovat. Grafické rozhraní se odvíjí od hlavní aplikace *UniPlot*.



Obrázek 5: UniPlot Addon Corba, Zdroj: [c]

2.2.3. UniPlot Addon DataFinder

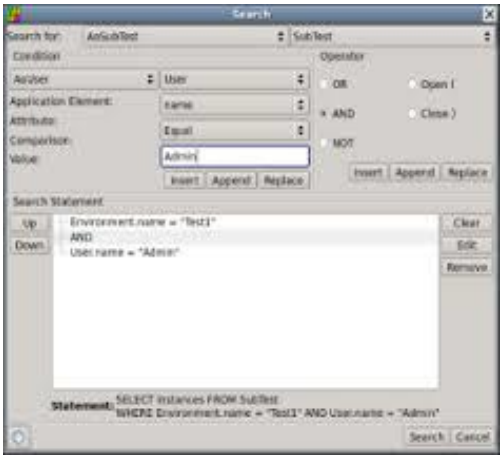
Také rozšíření pro aplikaci *UniPlot* [15] od společnosti *National Instruments [16]*. Využívá *CORBA* rozhraní (API 4). Umí ukládat více nastavených připojení, umí však přistupovat pouze k jedné současně. Grafické rozhraní se taktéž odvíjí od hlavní aplikace *UniPlot*.



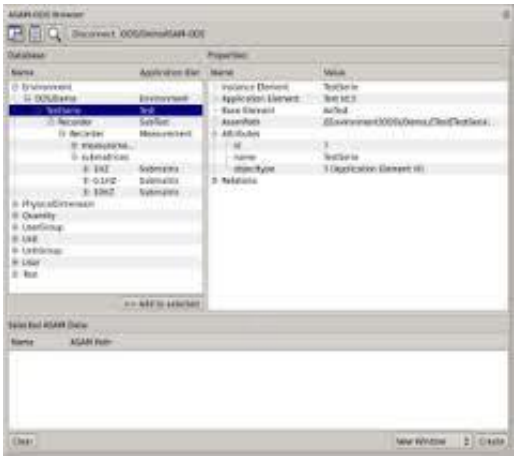
Obrázek 6: UniPlot Addon DataFinder, Zdroj: [d]

2.2.4. META ASAM ODS Browser

Nativní aplikace vyvinutá společností *BETA CAE Systems* [17], podporuje vyhledávání v metadatech a navigaci v úložišti za pomocí SQL dotazů, jelikož se jedná o placenou aplikaci tak jsem bohužel nemohl získat více informací ohledně fungování aplikace.



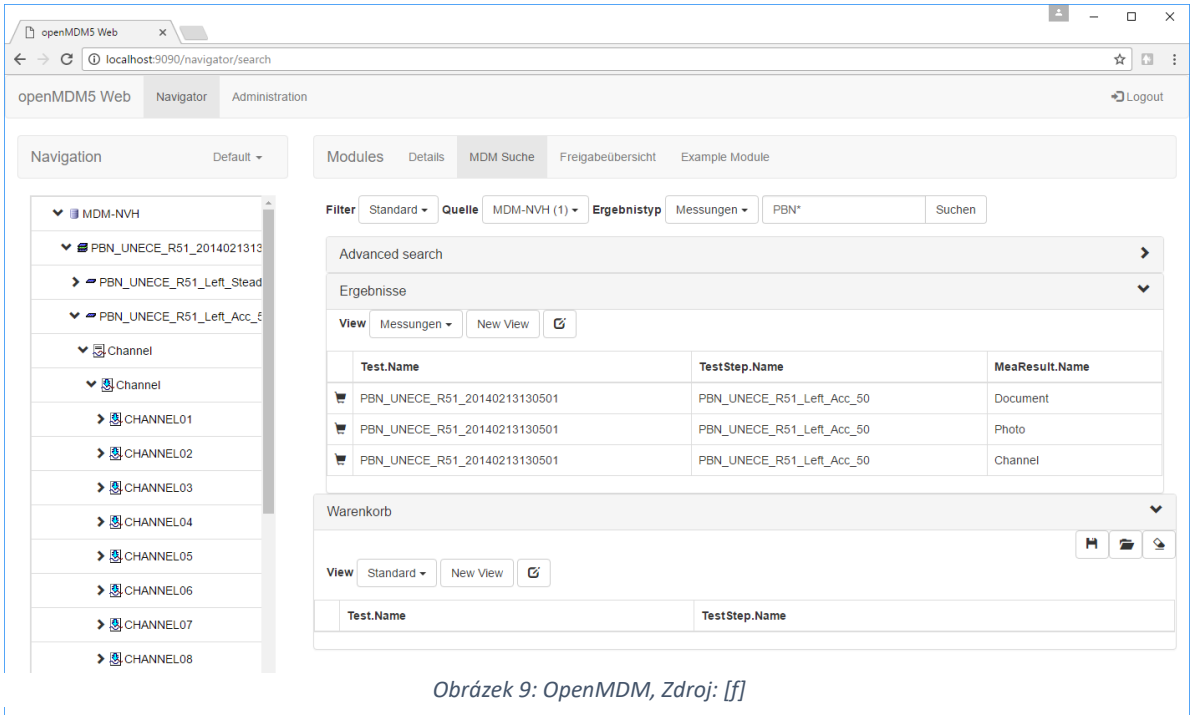
Obrázek 7: META ASAM ODS Browser, Zdroj: [e]



Obrázek 8: META ASAM ODS Browser 2, Zdroj: [e]

2.2.5. Open MDM

Balíček nástrojů *openMDM [18]* běžící pod nadací *Eclipse Foundation [19]*, původní vlastník projektu *openMDM* byla firma Audi AG. Nástroje umí spolupracovat s mnoha jinými aplikacemi (např. Matlab, Famos, Diadem, Artemis, …), umí importovat/exportovat data do/z různých formátů (např. MDM-XML, MS Excel, RPC II, ADTF, MDF3, …) a také i v datech



Obrázek 9: OpenMDM, Zdroj: [f]

vyhledávat.

2.3. Specifikace softwarových požadavků

Hlavním počátečním krokem vývoje softwaru je analýza požadavků aplikace pro uživatele a následné vytvoření uživatelských scénářů. Tyto požadavky pak dále rozdělujeme na *funkční* a *nefunkční* požadavky.

2.3.1. Funkční požadavky

1. Správa připojení
    a. Možnost vytvářet více připojení
    b. Možnost úpravy a mazání připojení
    c. Ukládání vytvořených připojení lokálně pomocí prohlížeče
2. Zobrazování informací
    a. Zobrazovat strom aplikačních elementů
    b. Zobrazovat instance elementů
    c. Zobrazovat relace a atributy elementů a instancí
    d. Zobrazovat informace o aplikačním modelu
    e. Zobrazovat hodnoty instancí
3. Správa jednotek
    a. Možnost výběru kompatibilní jednotky
    b. Možnost přepočtu hodnoty do jiné jednotky
    c. Zobrazování grafu měřených hodnot
4. Správa souborů
    a. Možnost stahování obsahu základních elementů
5. Možnost zpětné vazby

2.3.2. Nefunkční požadavky

1. Aplikace s webovým grafickým rozhraním
2. Přístup k datům pouze pro čtení
3. Práce s libovolným aplikačním modelem
4. Optimalizované rozhraní pro aktuální prohlížeče
5. Víceuživatelský přístup

2.4. Metodika vývoje

V dnešní době je standardem agilní vývoj softwaru. Agilní metodika je způsob jak si rozvrhnout a zorganizovat práci a nemusí se jednat pouze o vývoj softwaru. Agilní metodika je přímo konkrétní postup jak zvýšit produktivitu práce či dodržet specifikace.

Pro vývoj této aplikace jsem zvolil agilní metodiku vývoje *KANBAN [20],* která se využívá nejen při vývoji softwaru ale i v jiných odvětvích, například v callcentrech s využitím webového nástroje *Trello* [21].

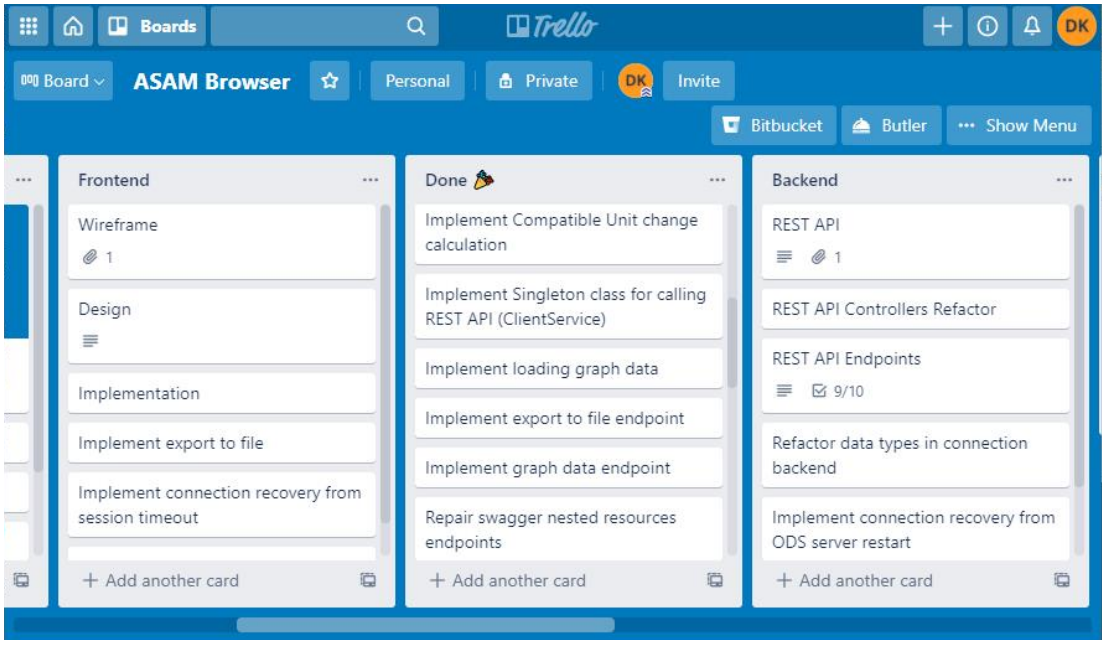
2.4.1. Kanban

Výrazně jednodušší agilní metoda vývoje než *SCRUM [22]*. Oproti metodě SCRUM nedefinuje tato metoda žádné role vývojářů či meetingy. Slovo „*Kanban*“ pochází z japonského jazyka a v překladu znamená „tabule“. V historii se kanbanem řídily počty lidí v chrámu ve starém Japonsku, kde časem tuto metodu převzala firma Toyota a s touto metodou začala používat pro řízení výroby v automobilovém průmyslu.

Kanban se řídí třemi pravidly:

1. Vizualizovat části projektu do tabule
2. Minimalizovat čas vývoje
3. Limitovat rozdělané úkoly

Pro mé užití jsem si *Kanban* upravil a tabuli rozdělil do několika sloupců jako: Frontend (úkoly týkající se Frontendu), Backend (úkoly týkající se Backendu), Bugs (zjištěné chyby aplikace), kvůli lepší přehlednosti, jelikož normálně by úkoly byly pomíchané v jednom sloupci a do sloupců In Progress (rozpracované úkoly) a Done (dokončené úkoly/bugy/testy).



Obrázek 10: Trello - Kanban

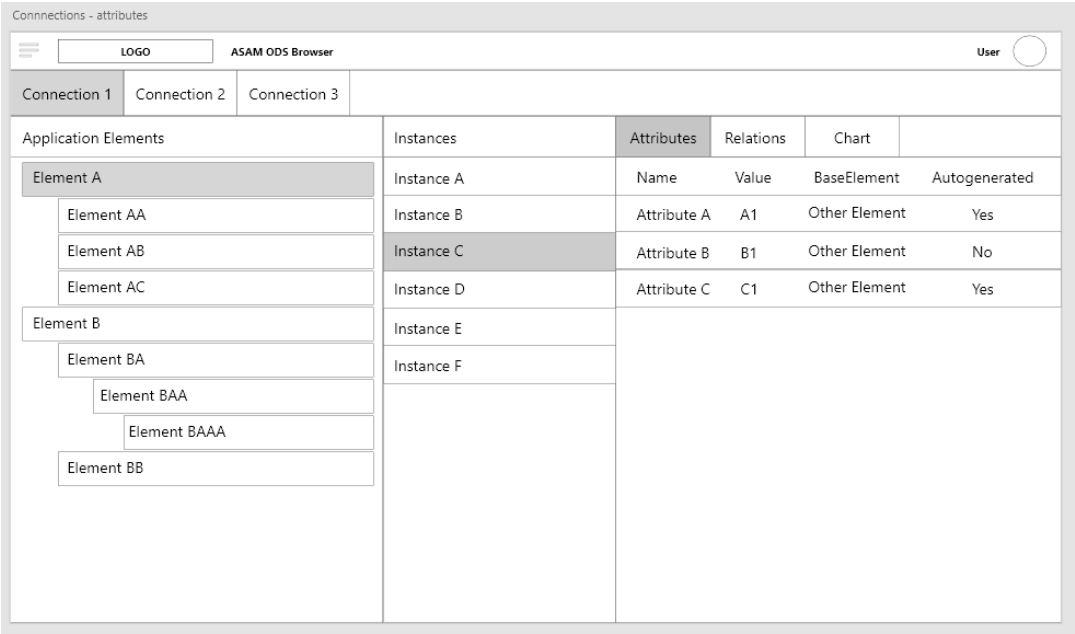
3 Design

3.1. Wireframes

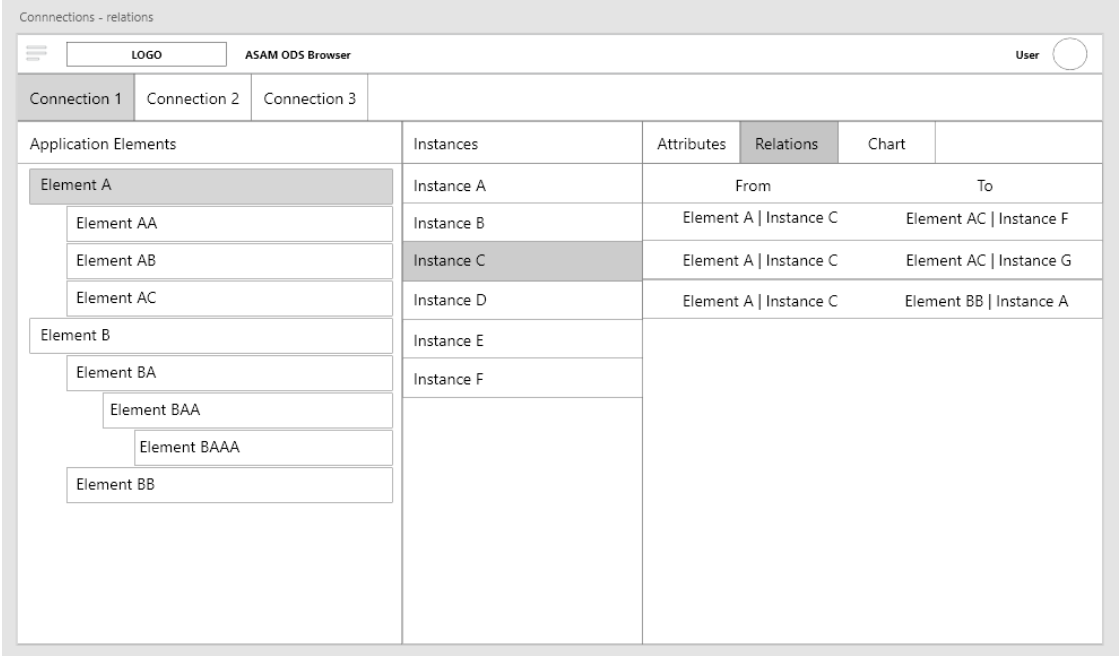
Schématický obrázek známý také jako drátový model, který naznačuje budoucí grafickou verzi stránky. Standardním postupem je vytvoření sady drátových modelů pro různé typy stránek (úvodní stránka, detail produktu, dashboard, nastavení). Hlavní účel wireframu je ukázat strukturu a předpokládaný obsah stránky. Pro vytvoření drátových modelů bylo využito nástroje *Adobe XD* [23].

Connections

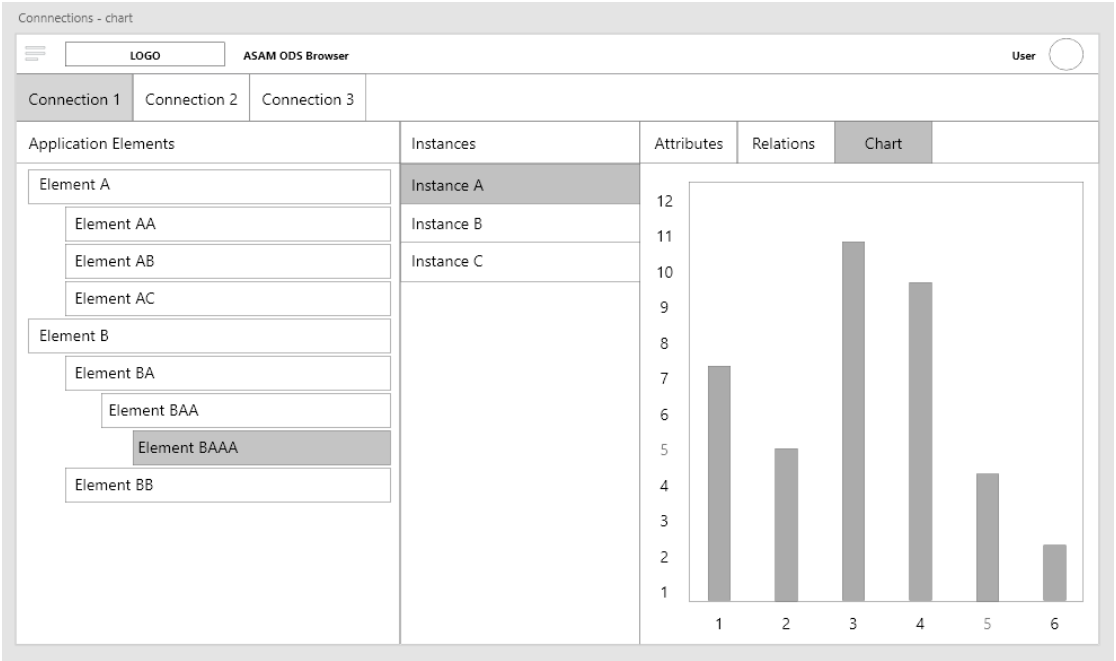
Jedná se o hlavní stránku aplikace, která bude uživatelem využívána nejčastěji. Primární funkcí této stránky je zobrazování dat uživateli, poskytuje funkcionalitu pro procházení dat.
V horní části stránky se nachází záložky pro možnost rychlého přepínání mezi připojeními. Dále je aplikace rozdělena do tří sloupců, kde v levém sloupci najdeme hierarchický strom aplikačních elementů a šedě označený vybraný element. V prostředním sloupci list instancí zvoleného elementu a šedě označenou vybranou instanci. Pravý sloupec slouží pro následné zobrazení metainformací o aplikačním elementu, atributech nebo relací.



Obrázek 11: Wireframe - Connection - Attributes



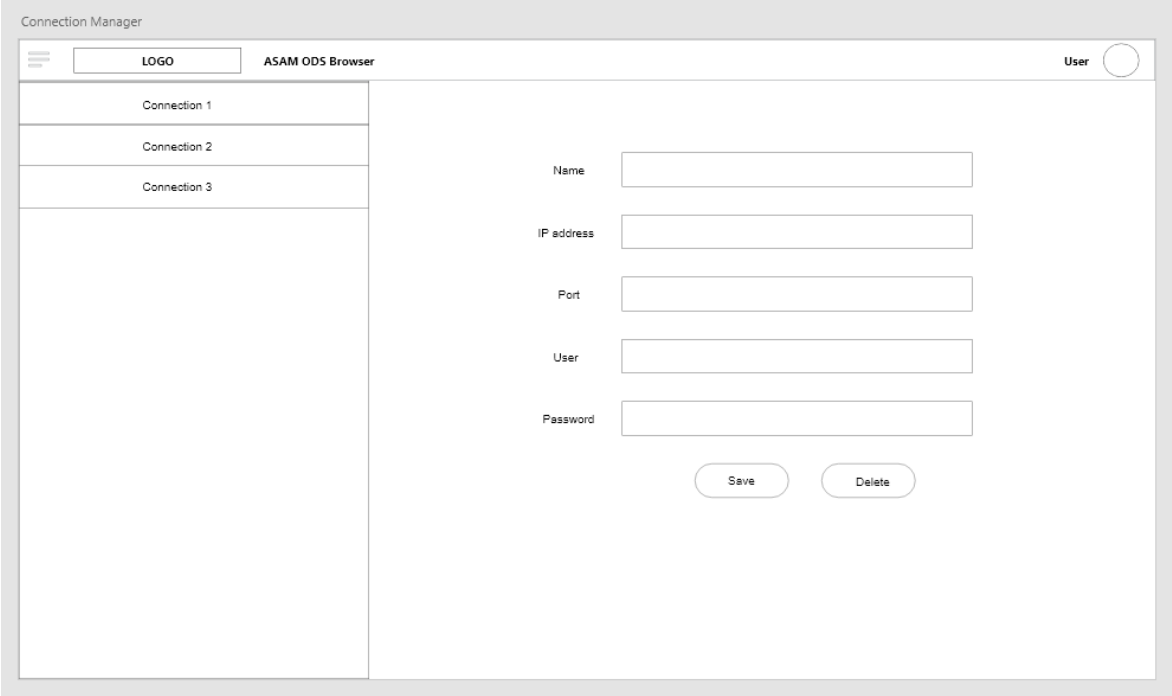
Obrázek 12: Wireframe - Connection - Relations



Obrázek 13: Wireframe - Connection - Chart

Connection manager

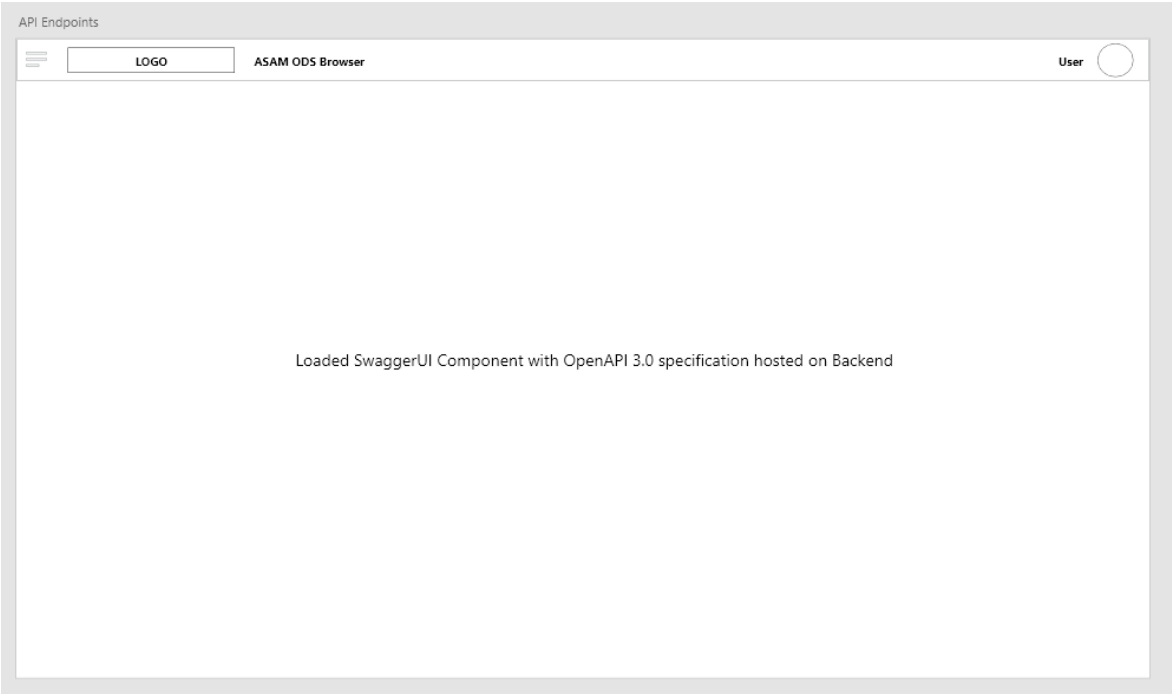
Tato stránka slouží pro vytvoření vlastních připojení k serverům ODS. Uživatel si může dané připojení pojmenovat dle svého uvážení, následně zadá IP adresu nebo host-adresu serveru, port služby a přihlašovací údaje.



Obrázek 14: Wireframe - Connection manager

API Endpoints

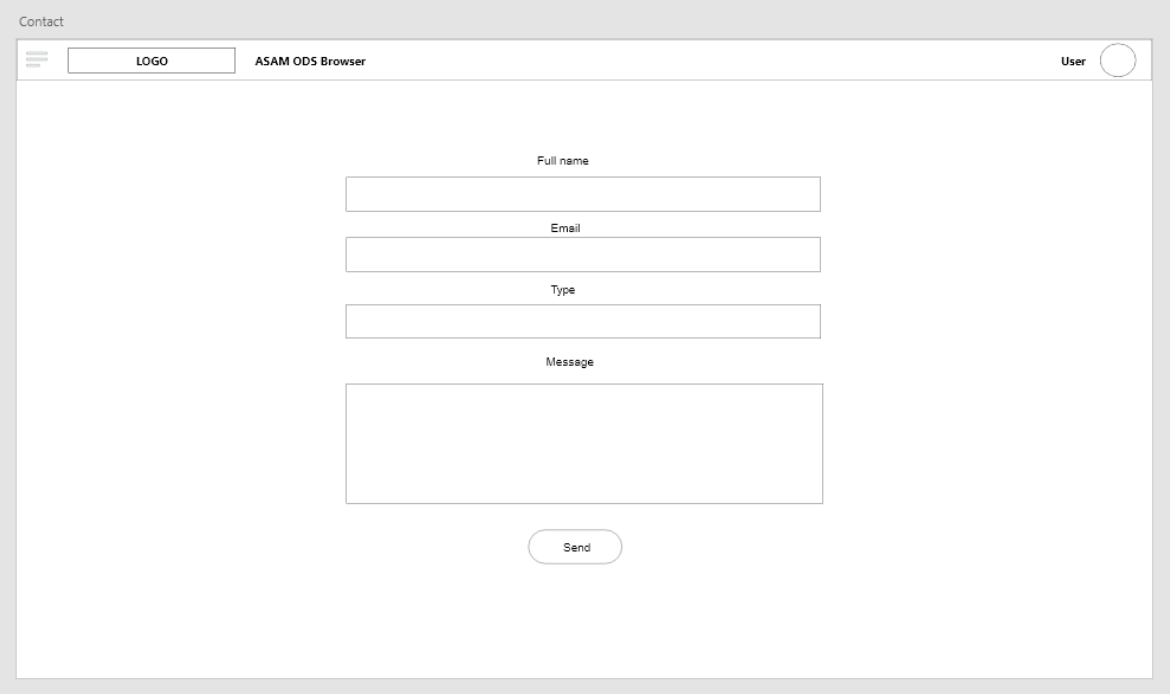
Tato stránka slouží pouze pro zobrazení dokumentace Rest API backendu za pomocí *Swagger UI [24]* componenty a OpenAPI specifikace [25] hostované na straně backendu.



Obrázek 15: Wireframe - API Endpoints

Contact

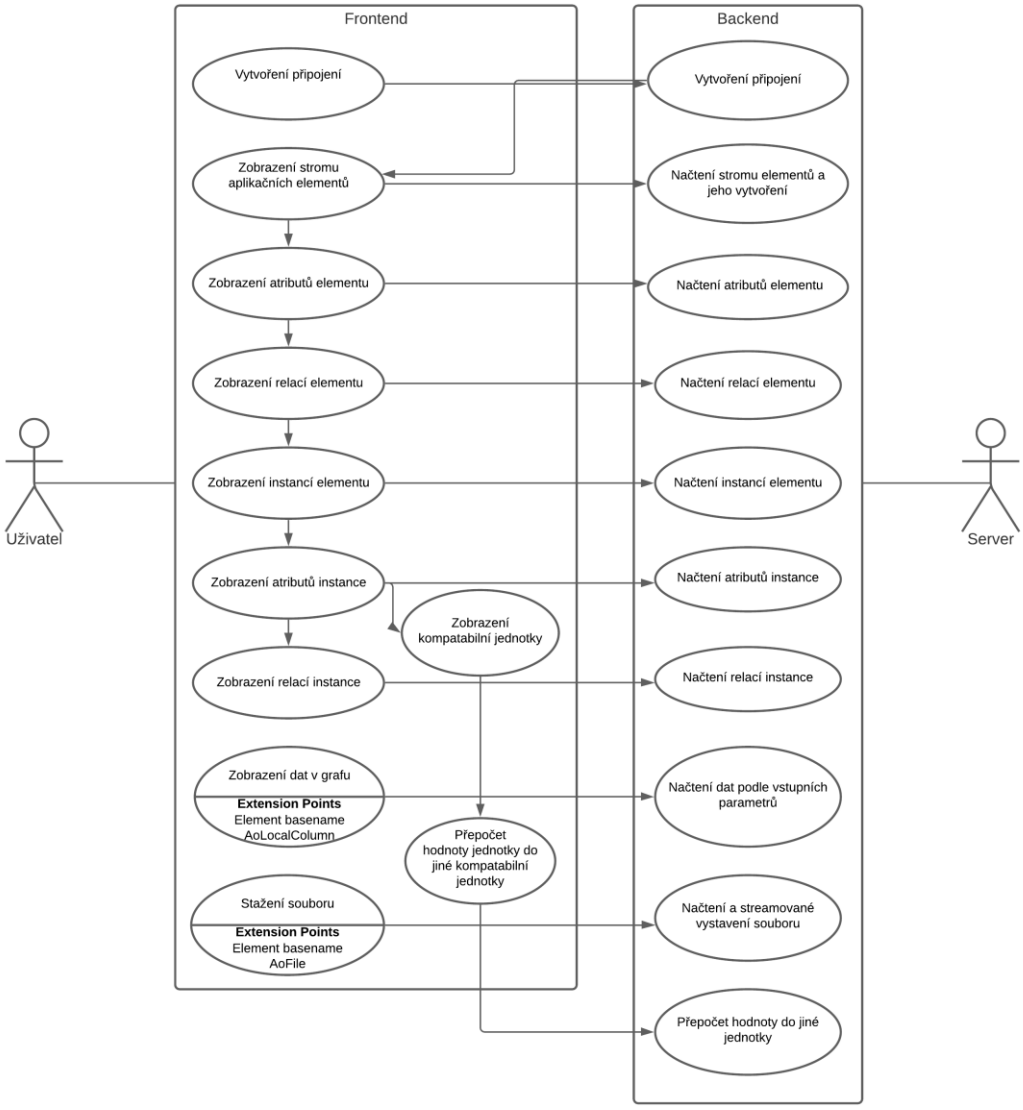
Na této stránce najdeme kontaktní formulář pro možnost získání zpětné vazby od uživatelů, kde si uživatel může vybrat z několika možností jako žádost o novou funkcionalitu, ohlášení chyby v aplikaci nebo položení své vlastní otázky správci aplikace.



Obrázek 16:Wireframe - Contact

3.2. Případ užití

Případy užití nebo také diagramy užití v jazyce UML se používají pro zjištění rozsahu aplikace. Je to srozumitelný graf jak pro vývojáře tak i zákazníka. Zobrazuje základní případy využití aplikace uživatelem, akce různých součástí aplikace a následné vztahy mezi akcemi či



Obrázek 17: Diagram užití

službami.

Funkčnost hlavního užití aplikace se podařilo zachytit do jednoho diagramu užití kdy na levé straně diagramu se nachází aktér *Uživatel* a na pravé straně aktér *Server* (myšleno ASAM ODS Server). *Uživatel* využívá akcí Frontendu který dále využívá akcí Backendu a ten následně *Serveru*. V grafu je následně vidět posloupnost akcí aplikace a základního postupu kdy aktér uživatel začíná v levém horním rohu od vytvoření připojení až po následné zobrazení konkrétních dat a jiných akcí zobrazené pomocí vztahů mezi akcemi.

3.3. Technologický stack

Autor využívá technologie se kterými má již předešlou zkušenost. Vzhledem k charakteru tohoto projektu byli zvoleny moderní technologie pro vývoj webových aplikací. Aplikace je **škálovatelná** a do budoucna je možné snadno nasadit novou funkcionalitu. Funguje tzv. **Out of the box** tím odpadá potřeba složité konfigurace či instalace na straně uživatele.

Git

Distribuovaný verzovací systém, obsahuje specifické nástroje pro vizualizaci a navigaci v historii vývoje projektu, původně vyvinut pro vývoj jádra Linuxu. [26]

Docker

OpenSource software pro kontejnerizaci aplikací. Jednoduchá standardizovaná jednotka která zaobalí aplikaci a všechny související náležitosti (kód, závislosti, knihovny, nastavení) do jednoho kontejneru pro rychlý a stabilní provoz aplikace bez nutnosti nastavování složitého vlastního prostření uživatelem či jiným vývojářem. [27]

GitLab

Webový git repozitář s podporou sledování chyb. [28]

GitLab Pipeline

Rozšíření webové aplikace Gitlab pro podporu CI/CD, pro snadnou automatizaci podmíněných úkolů a nasazení aplikací. [29]

OpenShift

Služba vyvinutá společností RedHat [30]. Je to služba pro Docker kontejnery spravované za pomocí Kubernetes [32], umožňující snadné nasazení těchto kontejnerů. [31]

Angular

Stěžejní technologií na klientské straně aplikace je framework Angular. Tento framework je postaven na tzv. komponentách. Je postaven na základech objektově orientovaného programování díky TypeScript třídám. K těmto třídám dále navazují HTML šablony či CSS šablony za pomocí tzv. direktiv.Je to rychlá webová a velmi rozvinutá webová platforma s mnoha možnostmi rozšíření. [33]

NPM

Nástroj pro správu balíčků a sestavení projektů pro JavaScript, převážně Node.js[34]. [35]

Material UI

je balík předepsaných komponent které lze upravovat, měnit motivy a hlavně využít jejich předepsanou funkcionalitu a grafický návrh využívaný a inspirovaný firmou Google a jejich systémem Android. [36]

Java EE

Součást platformy Java určená pro psaní a vývoj primárně podnikových aplikací s využitím vnitřní komunity různých firem a opensource vývojářů. Je velmi rozšířená ve web a cloud technologii, podporuje nejnovější webové standardy, které vývojáři požadují. [37]

Apache Maven

Nástroj pro správu závislostí a sestavování projektů. Primárně určený pro jazyk Java, ale dá se použít i v jiných programovacích jazycích. Jeho základní kámen je tzv. POM, kde jsou předepsané informace v XML jazyce o sestavení aplikace, verzi, balíku a závislostech projektu [39]

Rest Easy

Je projekt JBoss/Red Hat který nabízí různé frameworky pro tvorbu webových aplikací založených na Rest API [38], využívající HTTP protokol. [40]

JBoss EAP

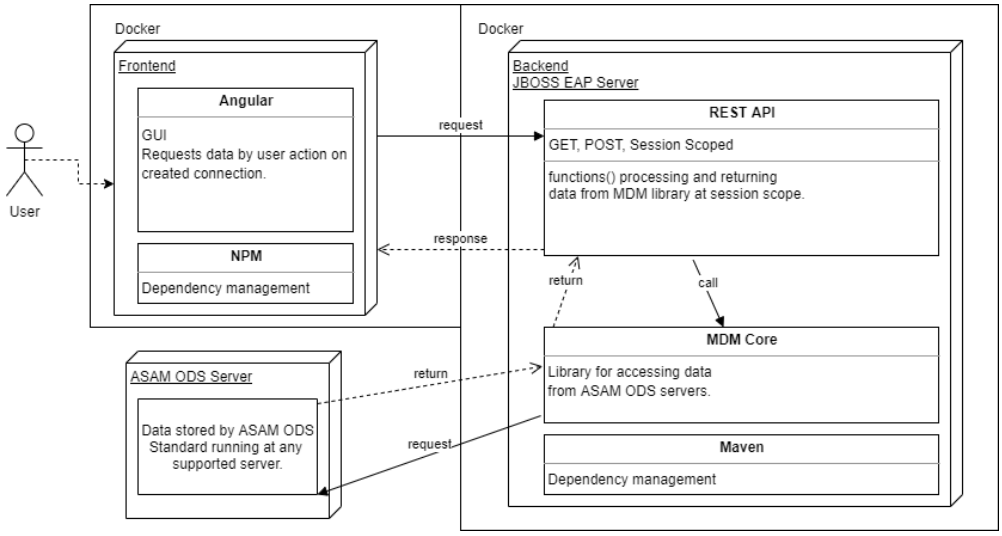
Aplikační server běžící na Java EE platformě. Je založen na open source aplikačním serveru Wildfly. Využívá Enterprise JavaBeans API a kontejnery pro správu transakcí a logiky. Využívá plnou logiku Java EE Software Stack (Web services engine, Scripted administration interface, Data grid solution, Transaction management service) pro běh komplexních aplikací. Podporuje různé frameworky a projekty (RestEasy, Hibernate, OSGI) např. pro moderní web-based aplikace (Spring, Angular, jQuery, GoogleWebToolkit). [41]

Knihovna MDM Core

Je část projektu z organizace školitele. Tato knihovna je především OO-API optimalizované na OO-API ASAM ODS. Je ve vývoji již od roku 2012. Dokáže data zpracovávat, ukládat do objektů, filtrovat i přepočítávat. V největší možné míře optimalizuje počet volání ODS serveru pomocí cache nebo upřednostněním voláním pro hromadná data před voláním objektovým, resp. iterátorovým. K potřebám vývoje aplikace bylo využito převážně funkcionality pro práci s dotazy, vytváření připojení, dotazování se na metadata modelu a ValueMatrix obsahující naměřená data.

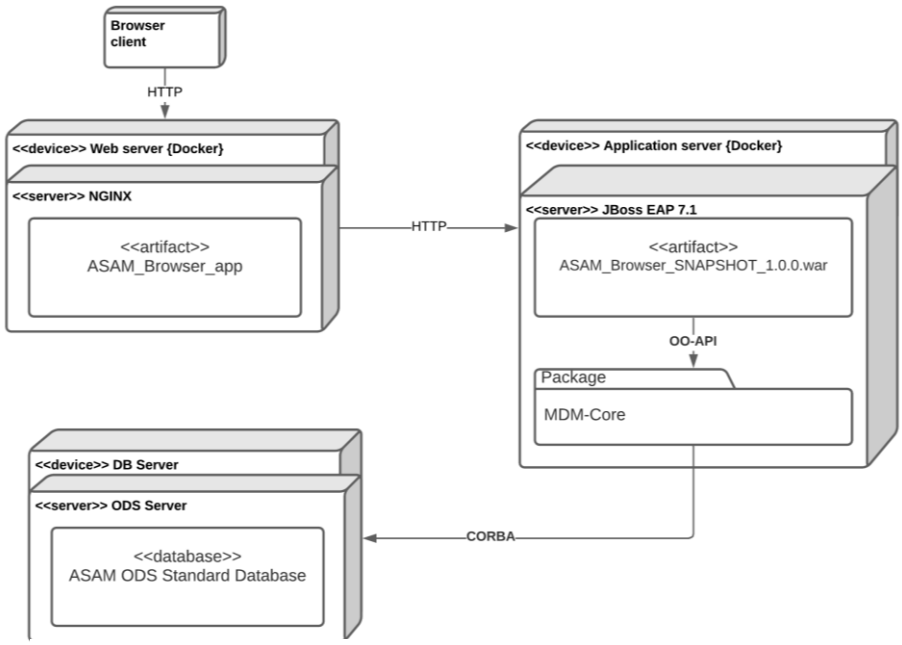
3.4. Architektura aplikace

Architektura aplikace se odvíjí od vybraných technologií. Aplikace je rozdělená na Backend a Frontend. Frontend je reaktivní za pomocí frameworku Angular a reguje na interakce uživatele podle kterých si žádá data z backendu pomocí Rest API. Využívá metod GET a POST a rozeznává uživatele pomocí vytvořených připojení s univerzálním časově vygenerovaným identifikátorem daného uživatele (session), která uchovává následně vytvořené připojení knihovny *MDM* s *ASAM ODS* Serverem kde jsou uložená data. Po vytvoření požadavku komunikuje Backend s knihovnou *MDM Core*, pomocí které vytváří dotazy pro získání dat ze serveru *ODS* a databáze. Frontend i backend je zvlášť izolován pomocí kontejnerů za využití softwaru Docker.



Obrázek 18: Architektura aplikace

Na obrázku je vidět diagram architektury, jak probíhají dotazy a vrácení dat či stavů z různých součástí aplikace. Následně je také přiložen diagram nasazení, který nám ukazuje rozmístění zdrojů.



Obrázek 19: Diagram nasazení

4 Implementace

4.1. Vývojové nástroje

Pro vývoj aplikace je použito jazyka Java a vývojového prostředí IntelliJ Idea, které poskytuje spousty možností rozšíření například podporu jiných programovacích jazyků jako zvolený Angular a Typescript. Hlavní použité rozšíření je nástroj SonarLint pro identifikaci a hlášení, označování možných hlubších chyb a navrhování vylepšení kódu či design patternů v reálném čase. Napomáhá k celkové lepší konzistenci kódu. Nástroj je zcela nastavitelný a je čistě na uživateli jaká pravidla pro psaní kódu si vybere.

Programovací konvence:

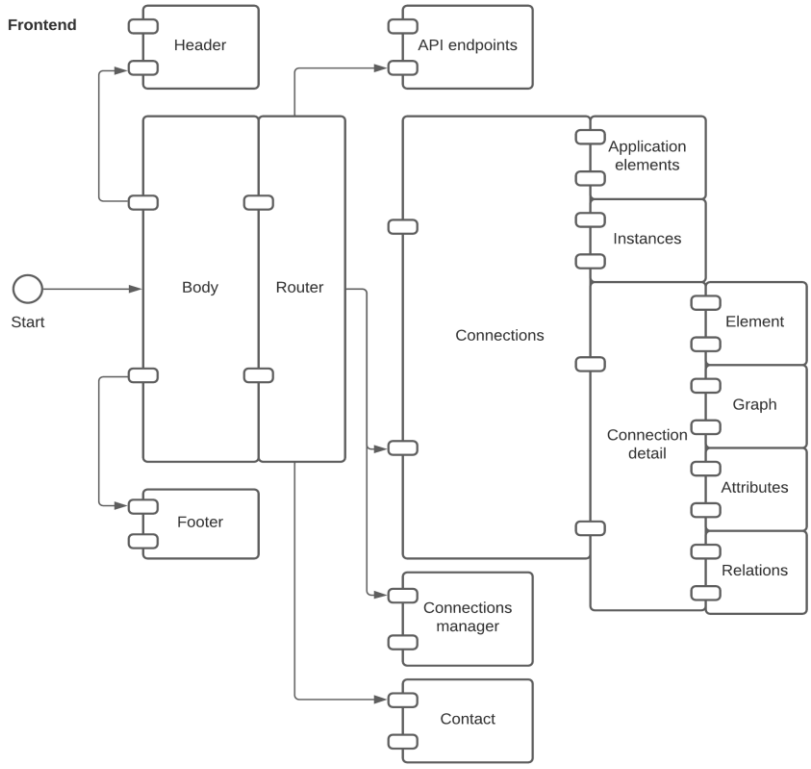
- Kontejnerizace aplikace za pomocí nástroje Docker
- SonarLint (Čistota, spravovatelnost, udržitelnost kódu)
- Znovupoužitelnost komponent
- Znovupoužitelnost RestAPI jinými vývojáři
- Verzování (Git)
- CI/CD (Continuous Integration/Continuous Delivery)

4.2. Frontend

Tato kapitola popisuje implementaci klientské části aplikace. Klientská část vzešla z předešlých návrhů a designů. Funguje jako samostaný projekt, zvlášt verzovaný za pomocí nástroje Git.

Základní struktura projektu je rozdělení na komponenty. Komponenty mezi sebou komunikují, předávají si data a také reagují na změny z jiných komponent či na aktivitu uživatele. Díky využití technologie Angular mají komponenty jasně definovanou strukturu. Jak mezi sebou budou komponenty komunikovat již ale určuje vývojář.

Základem bylo rozdělení na známé aspekty jako je Header neboli hlavička či záhlaví stránky která se stará o zobrazení loga, přepínání oken či jednoduché zobrazení informací o uživateli. Footer, neboli zápatí stránky slouží v případě této aplikace pouze pro zobrazení kontaktu a informace o vývojáři/majiteli projektu. Základní komponentou je především komponenta Body která obsahuje rozšířený komponent Router který slouží pro navigaci v aplikaci za pomocí URL odkazů. Router podle předem specifikovaného listu URL cest rozhoduje jakou komponentu má dále zobrazit. Podle zvolené cesty následně zobrazuje jednu z další úrovně komponent (API Endpoints, Connections, Connections manager, Contact), které se dále zanořují jak je zobrazeno na přiloženém diagramu komponent.



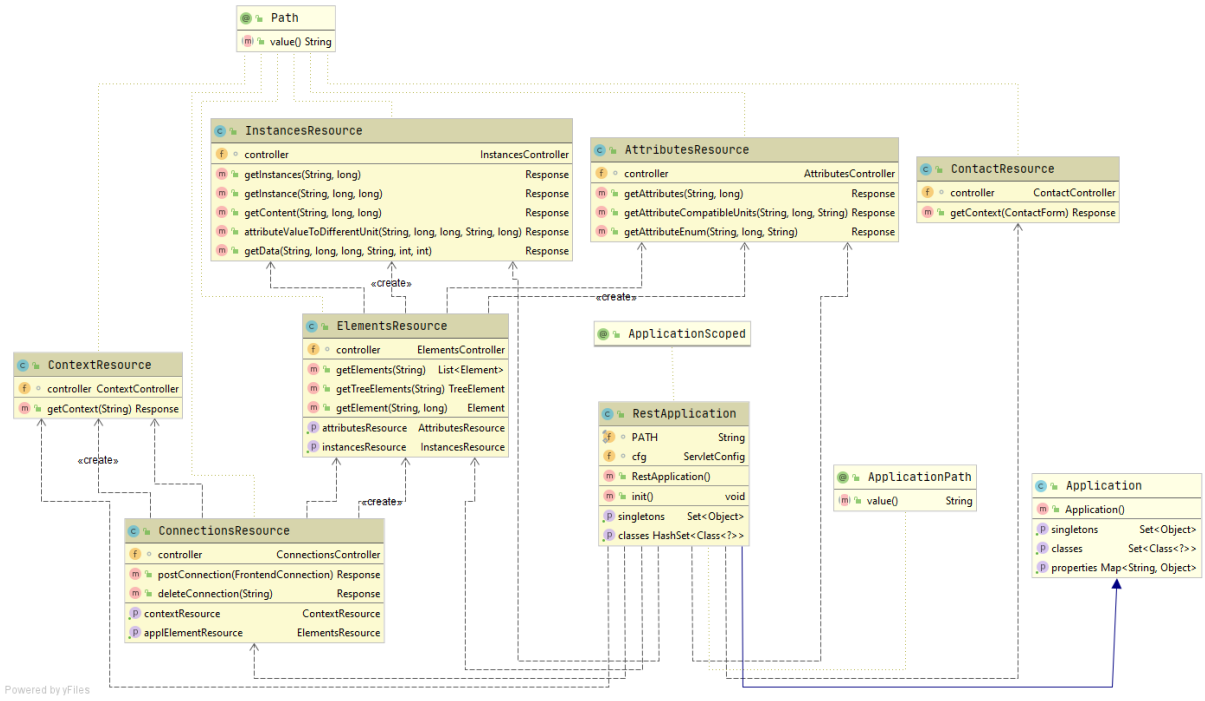
Obrázek 20: Diagram komponent

Komponenty obsahují vlastní funkcionalitu, kdy si podle vstupů uživatele volají pro svá specifická data díky službě apiService, která obsahuje předdefinované metody volání http požadavků napsané obecně pro daný endpoint na serverové části aplikace (backend).

4.3. Backend

V této kapitole je popsána serverová část aplikace, která vychází z návrhu architektury. Je to také samostatný verzovaný projekt za pomocí nástroje Git. Komunikace mezi klientem a serverem je jednoduchá, klient pošle požadavek pomocí HTTP protokolu na REST API na server, kde tento požadavek zpracuje a přiřadí příslušnému *controlleru*. Jako odpověď server pošle samotný stavový kód HTTP nebo stavový kód a požadovaná data ve formátu JSON [42]. Jak je vidět na diagramech tříd, u složitějších tříd je využito Builder design patternu [43], což je návrhový vzor, který slouží pro abstrakci tvorby složitějších objektů.

Základní strukturou tohoto projektu je rozdělení aplikace na tzv. *resources* neboli *prostředky*, které obsahují specifickou funkcionalitu balíčku *RestEasy* pro komunikaci přes http protokol. Každý *resource* má vlastní *controller*, který izoluje složitější funkcionalitu ohledně dotazování dat či vytváření objektů, což znamená využití MVC design patternu [44].

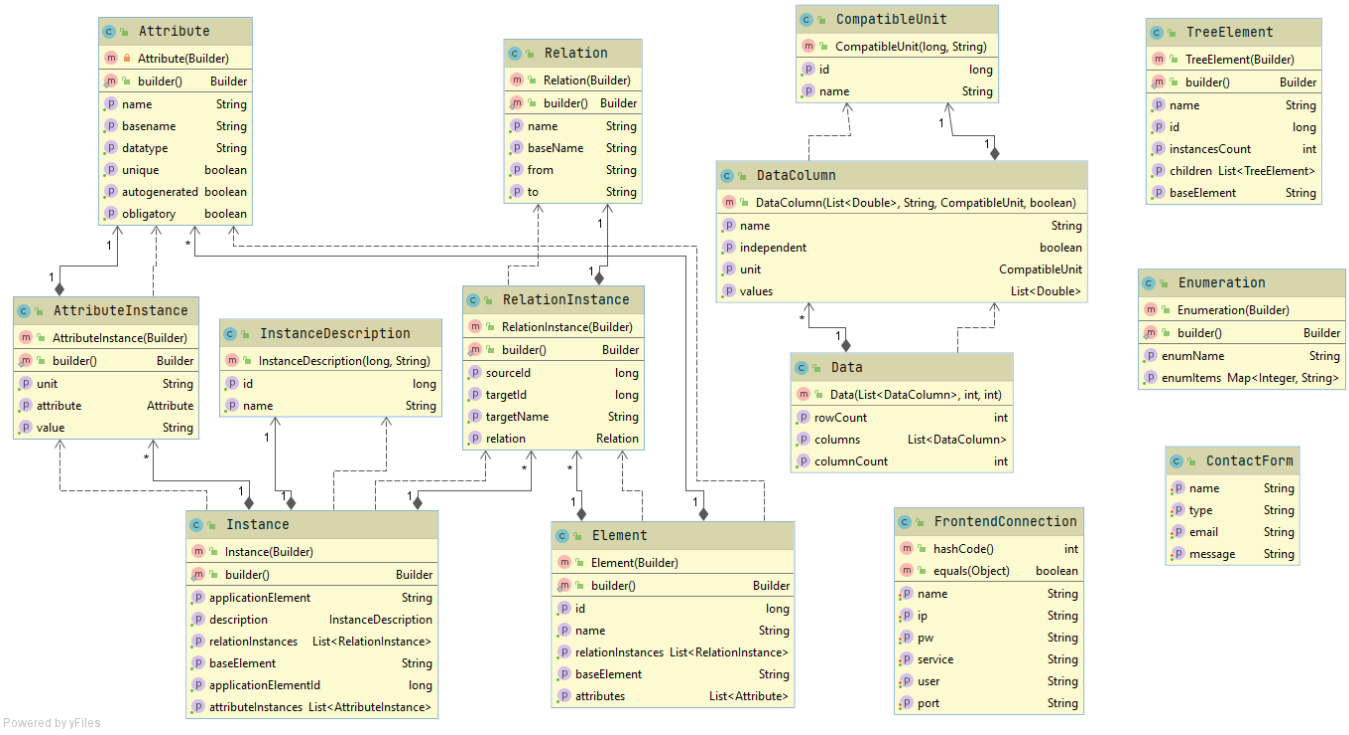


Obrázek 21: Diagram tříd - Resources

Tyto *resources* mají mezi sebou určitou vazbu kde první hlavní resource *RestApplication* slouží jako první úroveň dotázání na server za pomocí URL která se skládá z „*https://hostname:port/**rest***“, kde rest zastupuje třída *RestApplication*, která následně vytváří další úrovně, které v URL skládá za sebe jako vnořený *resource* kde například reprezentace třídy *ConnectionsResource* pomocí URL vypadá takto: „*https://hostname:port/rest/**connections***“.

Tato reprezentace třídy ovšem může obsahovat i vstupní parametry u některých metod které chceme za pomocí http dotazu předat také za pomocí již zmiňované URL: „*https://hostname:port /rest/connections/{**vstupní_parametr**}*“.

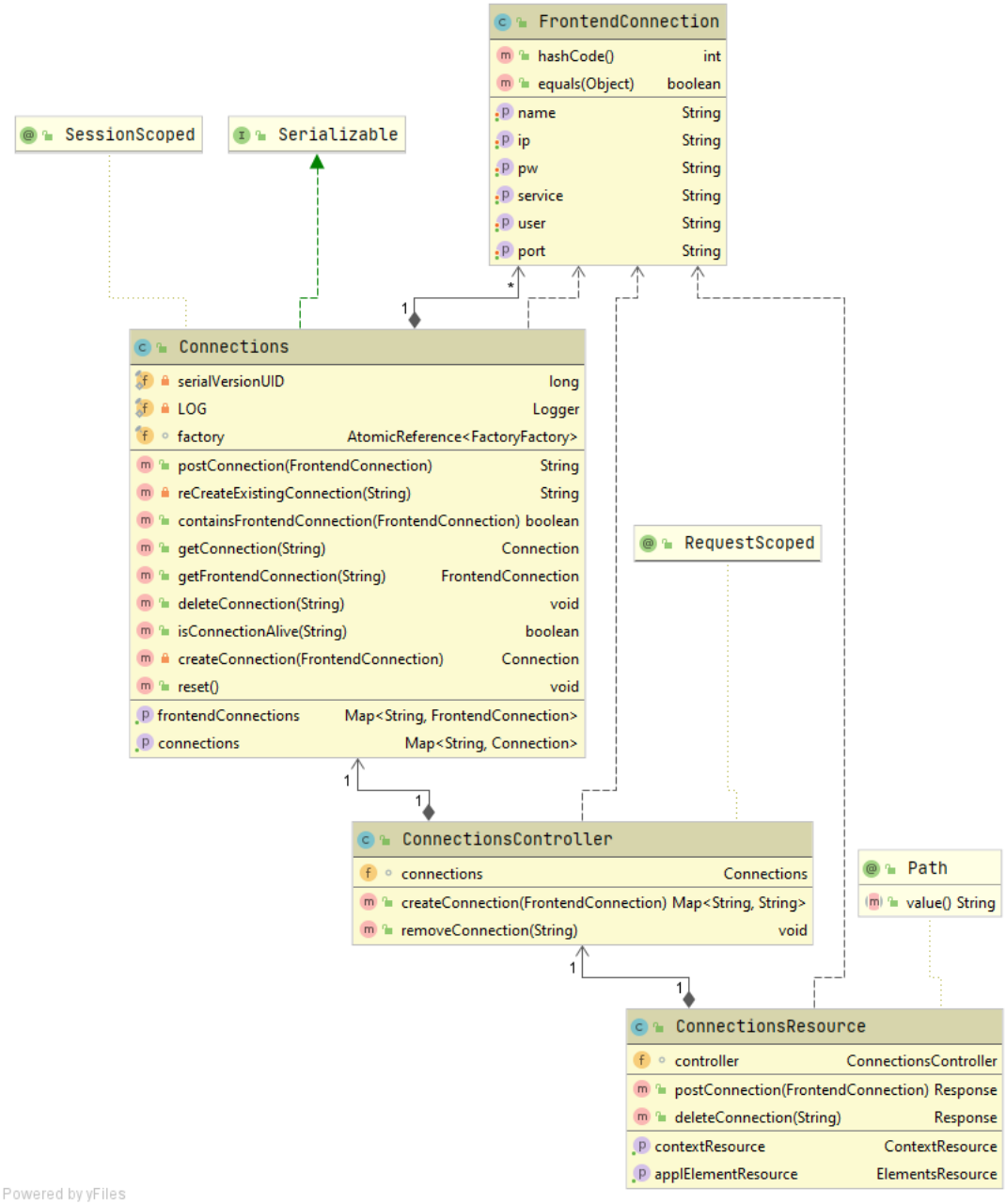
Tímto způsobem na sebe mohou *resources* navzovat i více do hloubky. Jak již bylo zmíněno, každý *resource* má svůj vlastní *controller*, který zajišťuje vytváření a plnění instancí tříd *models.* Takovýto model může být ve výsledku i složitější struktura zanořených jiných či dokonce stejných modelů. Zde je diagram tříd který obsahuje všechny vytvořené modely a jejich případné zanoření.



Obrázek 22: Diagram tříd Models

4.3.1. Connections resource

Jedná se o *resource*, který slouží pro vytváření, mazání, získávání či zjišťování stavu připojení na server ASAM ODS. Třída *ConnectionsResource* obsahuje metody volané pomocí http dotazu, kde například u metody *postConnection(Frontend Connection)* je jako vstupní parametr třída *FrontendConnection* , která je původně jako vstupní struktura za pomocí http requestu v body ve formátu JSON a automaticky se za pomocí knihovny RestEasy a implementace JAX-RS zkonvertuje do dané třídy vstupního modelu *FrontendConnection*. V tomto případě se volané metody třídy *ConnectionsResource* volají znova díky odizolování funkcionalit ve třídě *ConnectionsController*, která následně využívá metod třídy *Connections***,** která má předepsanou funkcionalitu jak zacházet a pracovat s připojeními přímo s objekty a dotazy balíčku MDM Core, který není na diagramech zobrazen, kvůli složitosti.



Obrázek 23: Diagram tříd - Connections resource

4.3.2. Elements resource

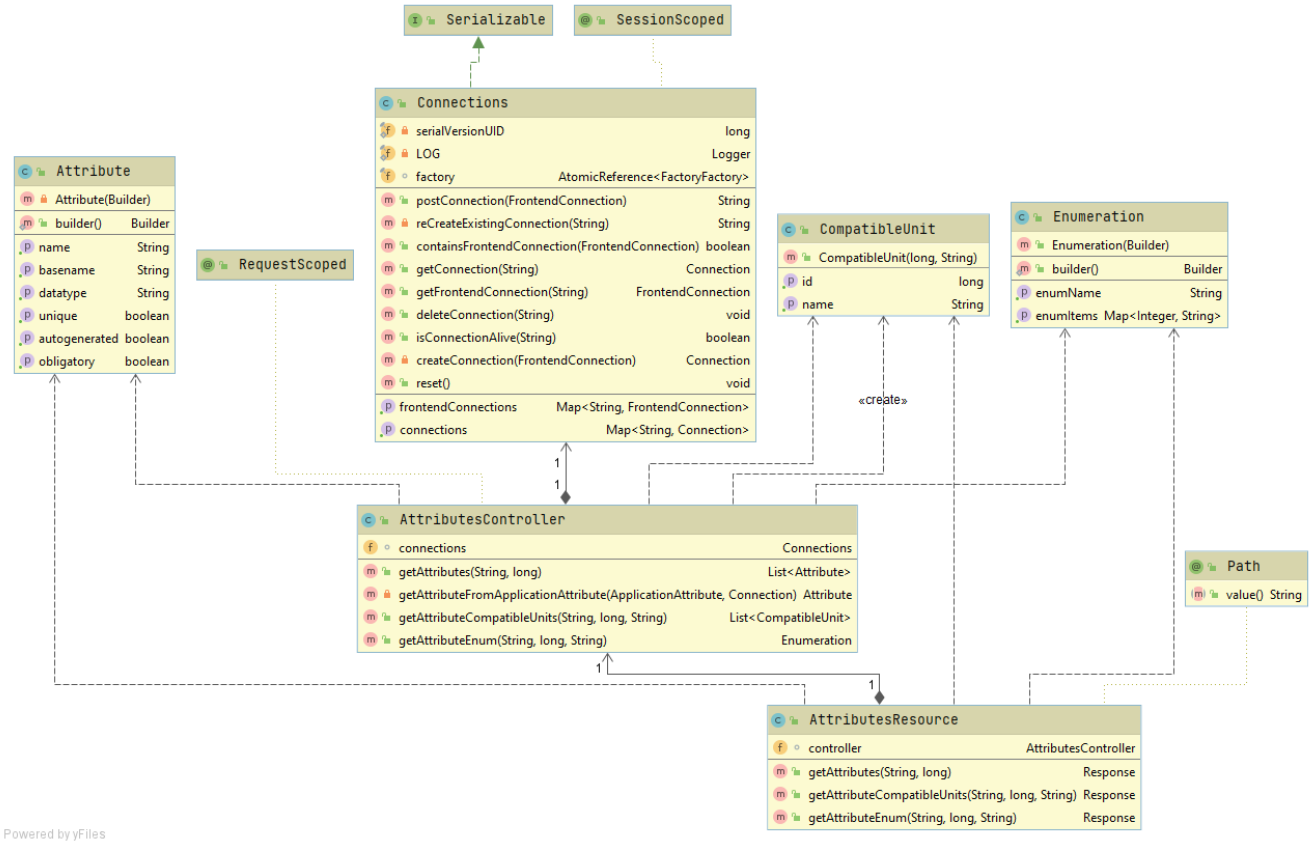
Tato struktura tříd slouží k dotazování dat ohledně Aplikačních elementů ASAM ODS aplikačního modelu za pomocí již vytvořeného připojení na daný ASAM ODS server, díky *Connections resource*. Metoda get*TreeElement(String connectionId)* vytváří pro Frontend zanořenou stromovou strukturu třídy TreeElement, která reprezentuje žádaná data z ASAM ODS serveru. Třída TreeElement nezobrazuje rekurzivní hierarchie na úrovni aplikačního elementu ale obsahuje základní informace o aplikačním elementu, které potom není potřeba opakovaně vyčítat. Dále máme také metody pro získání samotného aplikačního elementu za pomocí metody *getElement(String connectionId, long elementId)*, kde vstupní parametr *long* je id aplikačního elementu a *string* je identifikátor připojení, díky těmto vstupním parametrům jsou data identifikována a získána za pomocí dotazu na server ASAM ODS. Modelové struktury (*TreeElement, Element*) se díky *RestEasy* také automaticky transformují z instancí tříd jazyka Java do struktury formátu JSON pro použití v těle při odpovědi http dotazu volaného uživatelem, toto pravidlo automatické transformace je dosaženo díky Java EE anotacím a je použito u každého vstupního či výstupního modelu přes Rest API.

Obrázek 24: Diagram tříd - Elements resource

4.3.3. Attributes resource

Tento *resource* slouží převážně k získaní atributů Aplikačního elementu ASAM ODS aplikačního modelu. Díky získání předem vytvořeného připojení uživatele, se dotazuje také za pomocí balíčku MDM Core a jeho dotazovacího objektového rozhraní serveru ASAM ODS, podobným ale přesto specifickým způsobem díky jiné struktuře dat jako v *Elements resource*.

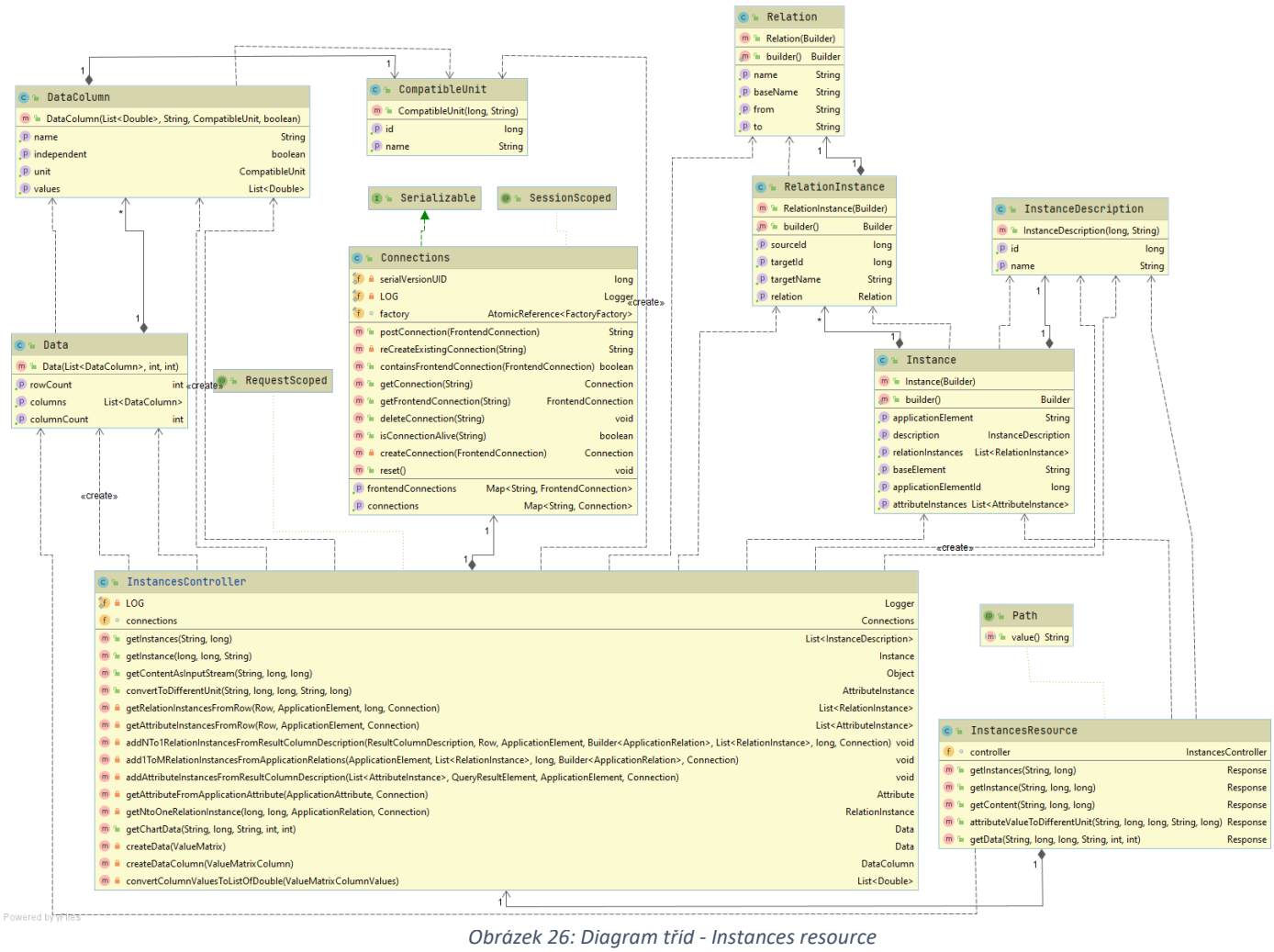
Metoda *getAttribute(String connectionId, long elementId)* slouží k získávání listu všech atributů aplikačního elementu daného za pomocí unikátního identifikátoru, dále *getCompatibleUnits(String connectionId, long elementId, String unitId)* metoda, která získává list všech kompatibilních jednotek daného atributu. Kompatabilní jednotky jsou kompatibilní tím že mají stejnou fyzikální veličinou (např. °C, K, °F a jejich fyzikální veličinu je teplota), tím nemůže nastat, že by přepočet hodnoty jednotky nešel logicky převést. Metoda *getAttributeEnum(String connectionId, long elementId, String attributeName)* získává enumeraci neboli celkový výčet informací o daném atributu, pokud je atribut datového typu enum. Aplikace nepodporuje sekvenční datové typy z ASAM ODS.



Obrázek 25: Diagram tříd - Attributes resource

4.3.4. Instances resource

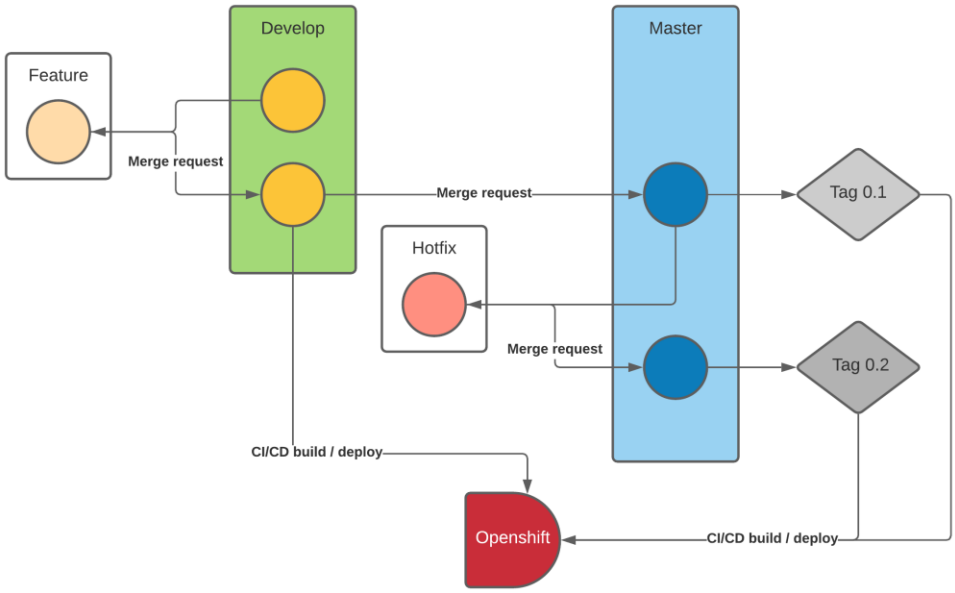
Nejsložitější struktura v celém projektu, která zastává velkou část funkcionality. Za pomocí podobného způsobu jako ostatní *resources*, získává data o instancích aplikačních elementů ASAM ODS aplikačního modelu. Je složitější převážně z důvodu že zastává čtení velkého množství dat uložených pod instancí specifického aplikačního elementu *AoLocalColumn* pro zobrazení na Frontendu za pomocí grafu, či získáváním relací instance, přepočítávání hodnot jednotek nebo také stahování uložených souborů pod instancí specifického aplikačního elementu *AoFile*.



Obrázek 26: Diagram tříd - Instances resource

4.4. Nasazení aplikace

Nasazení aplikace funguje na principu Git-Flow [45] s využitím GitLab Pipelines a nástroje GitLab a OpenShift. Vývojář tvoří novou funkcionalitu v nové feature větvi, kam nahrává své verze úprav aplikace, po implementaci funkcionality vytvoří merge request a následně po kontrole kódu sloučí feature větev do develop větve. Díky využití pipeline templatu je následně automaticky sestaven z develop větve artefakt aplikace a zaobalen docker kontejnerem s aplikačním serverem, který je automaticky nahrán do nástroje OpenShift který nám aplikaci spustí a bude dále udržovat a hostovat. Podobný postup následuje i ve větvi Master která slouží již pro stabilní verzi aplikace, kdy vývojář může označit neboli zmrazit verzi aplikace která je taktéž nahrána a hostována pomocí nástroje OpenShift na jiné produkční doméně.



Obrázek 27: GitFlow a Deployment

4.5. Testování

Testování aplikace probíhalo v rámci všech částí vývoje. Z počátku byl vytvořen interaktivní prototyp aplikace v programu Adobe XD, díky kterému bylo možné otestovat jak bude uživatel přecházet mezi okny a procházet data za inspirace již existujících řešení.

Testování technické funkčnosti Backendu je prováděno automaticky za pomocí jednotkových testů (JUnit) při sestavení aplikace za pomocí nástroje Maven. Díky použití nástroje Maven a spojením těchto jednotkových testů se nám provede tzv. jednotný integrační test.

Testování veškerých Rest API endpointů bylo provedeno pomocí nástroje Postman [46]. Jednotlivé endpointy byly volány tímto nástrojem, kde se kladl důraz na vrácení správných dat, popřípadě pro otestování chybových hlášení.

Nástroj Postman také podporuje integrační testování REST API. Díky vygenerované specifikaci Open API 3.0 kterou můžeme naimportovat do nástroje, je pak velice snadné vygenerovat předdefinovaný integrační test. Následně bylo spuštěno několik iterací tohoto integračního testu.

Součástí testování byla také potřeba zjištění rychlosti načítání dat aplikace kde se porovnává uplynulý čas načítání stejných dat na stejných serverech Mého řešení s aplikací ASAM Commander což byl hlavní důvod předělání/vytvoření vlastní aplikace. Všechny hodnoty jsou uvedeny v sekundách.

	Server develop		**Server production**	
Vyžádaná data	**ASAM Commander**	**Mé řešení**	**ASAM Commander**	**Mé řešení**
Strom elementů	7,21	2,02	13,22	6,47
Element 41 instance	3,67	0,218	11,64	2,31
Element 41 atributy	2,39	0,262	2,53	0,448
Element 41 relace	6,35	0,262	6,68	0,448
Element 41 Instance 61 atributy	2,45	0,579	1,36	1,03
Element 41 Instance 61 relace	21,52	0,579	9,35	1,03
Navigace pomocí relace	7,65	3,76	9,61	5,62

Tabulka 2: Porovnání načítání dat

V poslední části bylo zavedeno samotné uživatelské testování, při kterém uživatelé používali aplikaci a podávali zpětnou vazbu pomocí kontaktního formuláře. Díky tomuto testování, autor již naimplementoval některé žádané nové funkcionality či opravil zjištěné chyby aplikace.

Seznam změn

- Zobrazení počtu instancí aplikačního elementu ve stromu elementů
- Možnost jednoduchého vyhledávání v datech
- Notifikační služba pro lepší zpětnou vazbu mezi aplikací a uživatelem
- Navigace v aplikaci i za pomocí URL v prohlížeči
- Automatické přihlášení NTLM [47] uživatele za pomocí Keycloak [48], zobrazení informací o přihlášeném uživateli
- Předvyplněný kontaktní formulář podle přihlášeného uživatele
- Oprava špatného generování stromu elementů, kdy docházelo k zacyklení, původně se strom vytvářel pomocí procházení relací mezi elementy, nyní přímo pomocí procházení dětí elementu.
- Kontrola vstupů u kontaktního formuláře
- Možnost vlastního řazení tabulek atributů, instancí
- Možnost zobrazení tabulky hodnot načtené pro zobrazení grafu hodnot
- Možnost výběru vlastního rozmezí dat pro zobrazení grafu hodnot
- Zobrazení okna listu instancí pouze pokud má element nějaké instance
- Opraven přepočet jednotek
- Opraveno nenačítání grafu hodnot při správně zadané URL

5 Plánovaný rozvoj

Jak již autor uvedl, díky nasazení a pilotnímu provozu aplikace a také uživatelskému testování bylo již vytvořeno několik nových funkcionalit a opraveny některé chyby. Uživatelé však neváhají ani na vteřinu a nápadů na vylepšení či opravu chyb mají vždy dostatek. Proto je také stále co do budoucna zlepšovat, opravovat či vymýšlet.

Tabulka relací

- Žádost A:

  U tabulky aplikačních relací by bylo skvělé použít jiné pojmenování elementu této tabulky, Target name se nehodí, lepší by bylo použít Inverse name, display arity of the relation nebo relation type (Father, Child, Info, To, Info From), také by bylo dobré použít u různých druhů relací jiné barvy a ikony směru relace.

- Žádost B:

  První sloupec u této tabulky relací bych vůbec nezobrazoval, je pokaždé stejný, také zobrazovat id targetu.

Tabulka atributů

- Chyba A:

  Nelze změnit řazení sloupce value u tabulky atributů.

- Žádost B:

  Použít nějaké barevné pozadí v tabulce atributů pro oddělení sloupců.

Aplikační elementy

- Žádost A:

  Moc se mi nelíbí užití tooltipu se zobrazením ID.

- Žádost B:

  Nelíbí se mi překreslování stromu když si zvolím jiný element.

List instancí

- Žádost A:

  Načtení instancí by se mělo načítat asynchronně, nechci čekat než se načtou všechny detaily.

Ostatní

- Žádost A:

  Bylo by dobré držet kontext načtených dat a zvolených elementů/instancí/záložek když přepínám mezi stránkami třeba na kontaktní formulář.

Správa připojení

- Žádost A:

  Bylo by super mít možnost duplikovat vytvořený připojení.

- Žádost B:

  Správce připojení notifikuje změny za pomocí červeně zbarveného textu OK, očekával bych zelenou barvu.

- Žádost C:

  Jako vstup mi přijde pojmenování IP adresa trochu matoucí, protože to může být také hostname.

Kontakt

Žádost A:

Přidat možnost nahrát přílohu v kontaktním formuláři.

6 Závěr

Cílem této práce bylo zejména navržení a implementace webové aplikace pro procházení dat podle standardu ASAM ODS, která rychleji načítá data a má moderní a uživatelsky přívětivější grafické rozhraní než již použité řešení v organizaci školitele *ASAM Commander*. Z počátku byla provedena analýza, které je součástí rešerše samotného standardu ASAM ODS, jeho vysvětlení, možnosti přístupu a využití, následně rešerše ohledně již existujících řešení. Následovala specifikace požadavků tvořené aplikace, která má sloužit jako náhrada existujícího řešení *ASAM Commander*, také byla zvolena také metodika vývoje aplikace Kanban.

Dále návrh grafického rozhraní v podobě wireframů za pomocí nástroje Adobe XD, zjištění případů užití aplikace uživatelem, zvolení technologického stacku a navrhnutí architektury celé aplikace.

Aplikace byla na straně klienta vyvinuta za pomocí frameworku Angular. Pro serverovou část bylo využito frameworku RestEasy, jazyka Java a Rest API pro komunikaci.

Aplikace je cílena pro již zkušenější uživatele, kteří mají o standardu ASAM ODS přehled, a především nějaké zkušenosti, vědí jak funguje, jsou s ním seznámeni a jaké data především hledají. Aplikace má potenciál k využití ve více odděleních v organizaci školitele, nejen jako pomocník při vývoji aplikací podporujících standard ASAM ODS.

Celý projekt byl po celou dobu vývoje distribuován pomocí systému správy verzí Git a je dostupný interně pouze v organizaci školitele, stejně jako nasazený produkt.

Díky mnoha uživatelům v organizaci školitele bylo dosaženo důsledného testování aplikace, opravení chyb a zapracování výsledků či nových funkcionalit žádaných uživateli do novějších verzí aplikace.

Bylo dosaženo všech cílů práce, nejvíce pozornosti bylo věnováno především sestavení obecně použitelného Rest API backendu aplikace a také grafické použitelnosti a přehlednosti frontendu. Autor získal během implementace mnoho zkušeností převážně díky uživatelskému testování a zjištění, jak je důležité přemýšlet nad tím, jak s aplikací bude koncový uživatel vůbec zacházet a jak je důležité vybrat si správné technologie již v brzké fázi projektu.

Organizace školitele se autorovi vyjádřila k vývoji, otestování aplikace a pilotnímu provozu aplikace v dokumentu, který je k nalezení jako příloha této práce.

.

Seznam použité literatury

[1] Home. *Home* [online]. Copyright © 2020 [cit. 09.12.2020]. Dostupné z: https://www.asam.net/

[2] *ASAM ODS. Home [online]. Copyright © 2020 [cit. 09.12.2020]. Dostupné z: https://www.asam.net/standards/detail/ods/*

[3] FAQ | CORBA. *Welcome To CORBA Web Site!* [online]. Copyright © 1997 [cit. 09.12.2020]. Dostupné z: https://www.corba.org/faq.htm

[4] Remote Procedure Call (RPC) - CIO Wiki. *CIO Wiki - IT Management Glossary* [online]. Dostupné z: https://cio-wiki.org/wiki/Remote_Procedure_Call_(RPC)

[5] *Moved* [online]. Copyright © 1995, 2020 Oracle and [cit. 09.12.2020]. Dostupné z: https://docs.oracle.com/javase/tutorial/java/concepts/interface.html

[6] REST API vs Web API (vs SOAP API) [What's the Difference?] | RapidAPI. *RapidAPI - The Next Generation API Platform* [online]. Dostupné z: https://rapidapi.com/blog/rest-api-vs-web-api/

[7] What is a Database Management System (DBMS)? - Definition from Techopedia. *Techopedia: Educating IT Professionals To Make Smarter Decisions* [online]. Copyright © 2020 Techopedia Inc. [cit. 09.12.2020]. Dostupné z: https://www.techopedia.com/definition/24361/database-management-systems-dbms

[8] JUnit – About. [online]. Copyright © 2002 [cit. 09.12.2020]. Dostupné z: https://junit.org/junit4/

[9] ASAM MDF. *Home* [online]. Copyright © 2020 [cit. 09.12.2020]. Dostupné z: https://www.asam.net/standards/detail/mdf/

[10] ISO - ISO/TS 22240:2008 - Road vehicles — Vehicles safety information model (VSIM). [online]. Copyright © All Rights Reserved [cit. 09.12.2020]. Dostupné z: https://www.iso.org/standard/40792.html

[11] ASAM ODS server - Our Avalon ODS reference Server - HighQSoft. *Measurement Data Management with ASAM ODS - HighQSoft GmbH* [online]. Dostupné z: https://www.highqsoft.com/avalon-asam-ods-server/

[12] Measurement Data Management with ASAM ODS - HighQSoft GmbH. *Measurement Data Management with ASAM ODS - HighQSoft GmbH* [online]. Dostupné z: https://www.highqsoft.com/

[13] Java Web Start. *Moved* [online]. Dostupné z: https://docs.oracle.com/javase/8/docs/technotes/guides/javaws/

[14] Iterator (Java Platform SE 8). *Moved* [online]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html

[15] Visualize and Analyze 2D and 3D Data with UniPlot for Windows — UniPlot R2020 documentation. *Visualize and Analyze 2D and 3D Data with UniPlot for Windows — UniPlot R2020 documentation* [online]. Dostupné z: https://www.uniplot.com/

[16] Automatisierte Mess- und Prüfsysteme - NI. *301 Moved Permanently* [online]. Copyright © [cit. 09.12.2020]. Dostupné z: https://www.ni.com/de-de.html

[17] BETA CAE Systems - Home page. *BETA CAE Systems - Home page* [online]. Copyright © Copyright 2020 BETA CAE Systems All rights reserved [cit. 09.12.2020]. Dostupné z: https://www.beta-cae.com/

[18] OpenMDM - Measured Data Management | The Eclipse Foundation. *OpenMDM - Measured Data Management | The Eclipse Foundation* [online]. Copyright © Eclipse Foundation, Inc. All Rights Reserved. [cit. 09.12.2020]. Dostupné z: https://openmdm.org/

[19] Eclipse Foundation | The Eclipse Foundation. *Enabling Open Innovation & Collaboration | The Eclipse Foundation* [online]. Copyright © Eclipse Foundation, Inc. All Rights Reserved. [cit. 09.12.2020]. Dostupné z: https://www.eclipse.org/org/foundation/

[20] What is a Kanban Board? | Atlassian. *Atlassian | Software Development and Collaboration Tools* [online]. Copyright © 2020 Atlassian [cit. 09.12.2020]. Dostupné z: https://www.atlassian.com/agile/kanban/boards

[21] Trello. *Trello* [online]. Copyright © Copyright 2020. Všechna práva vyhrazena. [cit. 09.12.2020]. Dostupné z: https://trello.com/cs

[22] What is Scrum?. *Home | Scrum.org* [online]. Copyright © 2020 [cit. 09.12.2020]. Dostupné z: https://www.scrum.org/resources/what-is-scrum

[23] Adobe XD | Fast & Powerful UI/UX Design & Collaboration Tool. [online]. Dostupné z: https://www.adobe.com/products/xd.html

[24] REST API Documentation Tool | Swagger UI. *API Documentation & Design Tools for Teams | Swagger* [online]. Copyright © 2020 SmartBear Software. All Rights Reserved. [cit. 09.12.2020]. Dostupné z: https://swagger.io/tools/swagger-ui/

[25] What is OpenAPI 3.0? | Swagger Blog. *API Documentation & Design Tools for Teams | Swagger* [online]. Copyright © 2020 SmartBear Software. All Rights Reserved. [cit. 09.12.2020]. Dostupné z: https://swagger.io/blog/news/whats-new-in-openapi-3-0/

[26] Git. *Git* [online]. Dostupné z: https://git-scm.com/

[27] Empowering App Development for Developers | Docker. *Empowering App Development for Developers | Docker* [online]. Copyright © 2020 Docker Inc. All rights reserved [cit. 09.12.2020]. Dostupné z: https://www.docker.com/

[28] GitLab [online]. Dostupné z: https://about.gitlab.com/

[29] CI/CD pipelines | GitLab. *GitLab Documentation* [online]. Dostupné z: https://docs.gitlab.com/ee/ci/pipelines/

[30] Red Hat - We make open source technologies for the enterprise. [online]. Copyright ©2020 Red Hat, Inc. [cit. 09.12.2020]. Dostupné z: https://www.redhat.com/en

[31] What is OpenShift - Red Hat OpenShift. *Red Hat OpenShift, the open hybrid cloud platform built on Kubernetes* [online]. Copyright © 2020 [cit. 09.12.2020]. Dostupné z: https://www.openshift.com/learn/what-is-openshift

[32] Kubernetes. *Kubernetes* [online]. Copyright © 2020 The Kubernetes Authors [cit. 09.12.2020]. Dostupné z: https://kubernetes.io/

[33] Angular. *Angular* [online]. Dostupné z: https://angular.io/

[34] Node.js. [online]. Copyright © OpenJS Foundation. All Rights Reserved. Portions of this site originally [cit. 09.12.2020]. Dostupné z: https://nodejs.org/en/

[35] npm | build amazing things. *npm | build amazing things* [online]. Dostupné z: https://www.npmjs.com/

[36] Angular Material UI component library. *Angular Material UI component library* [online]. Dostupné z: https://material.angular.io/

[37] Java Platform, Enterprise Edition (Java EE) | Oracle Technology Network | Oracle. [online]. Copyright © 2020 Oracle [cit. 09.12.2020]. Dostupné z: https://www.oracle.com/java/technologies/java-ee-glance.html

[38] What is a REST API?. [online]. Copyright ©2020 Red Hat, Inc. [cit. 09.12.2020]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[39] Maven – Welcome to Apache Maven. *Maven – Welcome to Apache Maven* [online]. Dostupné z: https://maven.apache.org/

[40] RESTEasy - JBoss Community. *RESTEasy - JBoss Community* [online]. Dostupné z: https://resteasy.github.io/

[41] Red Hat JBoss Enterprise Application Platform (JBoss EAP). [online]. Copyright ©2020 Red Hat, Inc. [cit. 09.12.2020]. Dostupné z: https://www.redhat.com/en/technologies/jboss-middleware/application-platform

[42] JSON. *JSON* [online]. Dostupné z: https://www.json.org/json-en.html

[43] Builder. *Refactoring and Design Patterns* [online]. Dostupné z: https://refactoring.guru/design-patterns/builder

[44] MVC Design Pattern - GeeksforGeeks. *GeeksforGeeks | A computer science portal for geeks* [online]. Dostupné z: https://www.geeksforgeeks.org/mvc-design-pattern/

[45] Gitflow Workflow | Atlassian Git Tutorial. *Atlassian | Software Development and Collaboration Tools* [online]. Dostupné z: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

[46] Postman | The Collaboration Platform for API Development. *Postman | The Collaboration Platform for API Development* [online]. Copyright © 2020 Postman, Inc. All rights reserved. [cit. 09.12.2020]. Dostupné z: https://www.postman.com/

[47] NTLM Overview | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 09.12.2020]. Dostupné z: https://docs.microsoft.com/en-us/windows-server/security/kerberos/ntlm-overview

[48] Keycloak. *Keycloak* [online]. Dostupné z: https://www.keycloak.org/

[49] HEROUT, Pavel. *Učebnice jazyka Java*. 5., rozš. vyd. České Budějovice: Kopp, 2010. ISBN 978-80-7232-398-2.

[50] GAMMA, Erich. *Návrh programů pomocí vzorů: stavební kameny objektově orientovaných programů*. Praha: Grada, 2003. Moderní programování. ISBN 80-247-0302-5.

[51] Jeff Langr, Pragmatic Unit Testing in Java 8 with JUnit, ISBN 978-1-94122-259-1

Seznam obrázků

Zdroje obrázků

[a] *ASAM ODS Wiki – Base model* [online]. [cit. 2020-11-28]. Dostupné z: https://www.asam.net/short_url

[b] *ASAM ODS Wiki – Application model* [online]. [cit. 2020-11-28]. Dostupné z: https://www.asam.net/short_url

[c] *Uniplot Documents – AODS 4 Browser* [online]. [cit. 2020-11-28]. Dostupné z: https://www.uniplot.de/documents/en/_images/aods4-browser.png

[d] *National Instruments – ASAM Browswer* [online]. [cit. 2020-11-28]. Dostupné z: https://www.ni.com/cms/images/devzone/tut/ASAM_Browswer.jpg

[e] *BETA Simulation Solutions – META ASAM ODS Browser* [online]. [cit. 2020-11-28]. Dostupné z:https://www.beta-cae.com/pdf/meta_asam_ods_browser.pdf

[f] *Automotive Testing Blog – openMDM* [online]. [cit. 2020-11-28]. Dostupné z https://automotivetestingblog.files.wordpress.com/2017/02/openmdm-web-client.png

Přílohy

Vyjádření k bakalářské práci

Hlavním přínosem bakalářské práce "Vývoj webové aplikace pro navigaci v úložišti podle standardu ASAM ODS" je software umožňující efektivní přístup k datům uloženým na ASAM ODS serverech pomocí webového prohlížeče. Současné řešení vyžaduje instalaci na straně klienta a nutnost distribuovat aktualizace ke klientovi, všechny tyto aktivity centralizovaným nasazením odpadají.
Další výhodou je rozdělení software na klientskou a serverovou část, tím došlo k enormnímu zrychlení přístupu k datům, protože náročná část komunikace s ASAM ODS serverem se odehrává v serverové části aplikace, která je nablízku ASAM ODS serverům, zatímco klient je často na vzdáleném pracovišti.
Software má moderní uživatelské rozhraní, které je reaktivní, jak je tomu zvykem u moderních webových aplikací. Načítání dat často probíhá až na základě konkrétního uživatelského požadavku, což opět snížilo objem přenášených dat nebo na pozadí. Celé uživatelské rozhraní je velmi přehledné, umožňuje snadnou navigaci a přidává doposud nedostupnou funkcionalitu (stahování souborů, zobrazení počtu instancí, vylepšené zobrazování grafů atp.).
Takový software nebyl doposud v praxi dostupný a v současnosti pokrývá naše potřeby z hlediska zkušeného uživatele, pro nezkušené uživatele by bylo zapotřebí nabídnout pozměněnou variantu uživatelského rozhraní (skrýt metainformace o aplikačním modelu). Postupy použité v této práci je možné dále rozvíjet a dosáhnout tak ještě větší efektivity práce s daty na ASAM ODS serverech.
V budoucnosti ještě dojde k zapracování zbývajících požadavků, oprav a návrhů vzešlých z pilotního provozu.
Možným zásadním rozšířením by bylo přidání funkcionality pro zápis/mazání nebo vyhledávání Toto však nemělo být předmětem bakalářské práce. Dalším krokem pro rozšíření aplikace by bylo odstranění některých okrajových omezení, které jsou z pohledu našich současných potřeb nepodstatné (podpora sekvenčních datových typů, komplexních čísel nebo binárních dat), tím by se stala aplikace naprosto univerzální.

V Českých Budějovicích, dne 8.12.2020

Ing. Miroslav Duník

pki, [redacted], CZ, M, I, Miroslav.Dunik
2020.12.08
17:14:55 +01'00'