VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

KONFIGURÁTOR ROZVODNY VELMI VYSOKÉHO NAPĚTÍ

EXTRA HIGH VOLTAGE SUBSTATION CONFIGURATOR

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. Patrik Pokorný

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Ladislav Dobrovský

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Patrik Pokorný**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Ladislav Dobrovský**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Konfigurátor rozvodny velmi vysokého napětí

Stručná charakteristika problematiky úkolu:

Proces přenosu požadavků zákazníků na konstrukční oddělení používající CAD systémy a naplánování výroby v případě vícero možných továren v rámci spolupracujícího koncernu lze výrazně zlepšit automatizací softwarem spojující přístupy konfigurátoru a editoru schémat (single line diagram) a případně modulu CRM systému.
Vzhledem k unikátnosti každého skládaného řešení rozvodny velmi vysokého napětí a povaze velmi individuálního vztahu se zákazníkem (stát, města) jsou v procesu místa, která zatím stojí pouze na lidech. Tito mohou ve svém rozhodování velmi benefitovat. Pro přidělení výroby továrnám vhledem k dopravě do místa sestavení může dát SW dobrou představu i o finanční náročnosti řešení ještě před sestavením individuální nabídky k výběrovému řízení.
Také možnost zobrazení skládaných celků prostředky počítačové 3D grafiky ovlivňuje představu zákazníka o prostorovém uspořádání a náročnosti řešení pro územní plánování a případný výkup pozemků ještě před finálním návrhem konstrukčního oddělení.

Cíle diplomové práce:

– Rešerše webových technologií pro interaktivní aplikaci s editorem.
– Volba vhodné knihovny pro 3D rendering.
– Návrh databáze s ER diagramem.
– Implementace prototypu webové aplikace s produkty firmy Hitachi Energy včetně 3D modelů.
– Analýza možností optimalizace plánu výroby a transportu voleného řešení.

ABSTRAKT

Předmětem této práce je implementace webové aplikace pro konfiguraci rozvodny velmi vysokého napětí. Aplikace kombinuje zobrazení prostorového modelu a jednopólového schématu konfigurovaného produktu. Tohoto je dosaženo kombinací technologií využívajících standardu HTML5 a JavaScriptu.

ABSTRACT

The subject of this thesis is the implementation of a web application for configuring a extra high voltage substation. The application combines a view of a 3D spatial model and a single-line diagram of the configured product. This is achieved using technologies based on HTML5 standard and JavaScript.

KLÍČOVÁ SLOVA

Konfigurátor produktu, webová aplikace, produktová vizualizace, plynem izolovaná rozvodna

KEYWORDS

Překlad klíčových slov Product configurator, web application, product visualization, gas insulated switchgear

2022

BIBLIOGRAFICKÁ CITACE

POKORNÝ, Patrik. *Konfigurátor rozvodny velmi vysokého napětí.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 2022, 49 s. Diplomová práce. Vedoucí práce: Ing. Ladislav Dobrovský

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků.

V Brně dne 20. 5. 2022 .....

Patrik Pokorný

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat Ing. Ladislavu Dobrovskému za vedení, cenné rady a odbornou pomoc při zpracování této práce. Dále bych chtěl poděkovat Ing. Petru Šmejkalovi za poskytnutí informací a dat k plynem izolovaným rozvodnám společnosti Hitachi Energy.

OBSAH

1 Úvod

Vývoj webových technologií v poslední dekádě přinesl nové možnosti prezentace produktů na internetu. Způsob prezentace se posouvá od strohého popisu s obrázkem k stále detailnější vizualizaci. Přidáním interaktivních prvků je navíc možné zapojit kreativitu zákazníka a umožnit mu přirozeným způsobem seznámit se podrobněji s různými variantami produktu. Zákazník se tak může cítit součástí procesu od samého začátku.

Výzvou v tomto ohledu je vytvořit co nejjednodušší uživatelské rozhraní, při zachování maximální variability produktu. Ukázkám různým přístupům firem v pojetí konfigurátoru je věnován prostor v následující kapitole. Některé z těchto přístupů posloužily jako inspirace a jsou zahrnuty i ve výsledném konfigurátoru. Hlavní součástí aplikace je 3D prostředí, které umožňuje zákazníkovi seznámit se s prostorovou orientací produktu. Nicméně samotný 3D model nemusí v případě zákazníka neseznámeného s portfoliem firmy Hitachi Energy utvořit jasnou představu o produktu. Proto aplikace implementuje také náhled na jednopólové schéma rozvodny, které je univerzální mluvou v tomto energetickém odvětví.

2 Plynem izolované rozvodny

Tyto rozvodny se funkcí neliší od jinak známých vzduchem izolovaných rozvoden pro velmi vysoké napětí, které můžeme klasicky spatřit v blízkosti elektráren. Rozvodny velmi vysokého napětí jsou také součástí každého uzlu distribuční energetické soustavy. Jedna z výhod plynem izolovaných rozvoden spočívá v jejich kompaktnosti, což předurčuje použití tohoto typu rozvoden pro uzly distribuční soustavy umístěné v hustě zastavěných oblastech. Nejrozšířenějším izolačním médiem je v současnosti fluorid sírový, v energetickém průmyslu často nazývaný SF6 (vychází ze vzorce pro chemickou sloučeninu SF_6). Tato látka není pro živé organismy jedovatá, jedná se však o skleníkový plyn s daleko větším dopadem na životní prostředí než oxid uhličitý. Pro časový horizont 20 let má plyn SF6 potenciál globálního oteplování (GWP) 16300x větší než oxid uhličitý [1]. Z těchto důvodů je také ve společnosti Hitachi Energy vyvíjeno a testováno nové ekologické izolační médium, které je již dnes nabízeno pro některé produkty, pod názvem EconiQ.

Vodič v plynem izolované rozvodně je koaxiálně uložen v hliníkovém pouzdru, které je naplněno izolačním plynem. Celá rozvodna se tedy skládá z koaxiálních dílů, které podléhají firemní standardizaci. Ačkoliv byla v poslední době vyvinuta velká snaha o standardizaci kompletních řešení celé stanice, výrobce musí být stále schopen řešit individuální požadavky na změnu rozložení. Tyto požadavky se často týkají speciálních umístění senzorů a modulů pro měření a zaručení provozuschopnosti celé rozvodny v případě poruchy na libovolné součásti, a to i během odstraňování této poruchy. S tím jsou také spojeny požadavky na bezpečnost personálu během řešení nenadálých situací, které se promítají do celkového plánu rozvodny.

3 Vybrané konfigurátory dostupné na webu

Konfigurace výrobků je dnes běžnou součástí internetových obchodů. Možnosti konfigurace se ale často omezují na jednoduché parametry, jejichž změna se projeví pouze na ceně výrobku. Náhled na výsledný produkt ale často chybí. Vzhledem k charakteristice implementovaného produktu bylo obtížné najít obdobnou webovou aplikaci.

3.1 iDesigner

Jedná se o komplexní webovou aplikaci od polských autorů, která obsahuje rozsáhlou knihovnu modelů a textur stavebních komponent a nábytku. Cílem aplikace je umožnit zákazníkovi vytvořit si vlastní model domu, který pak může upravovat a doplňovat objekty z knihovny, a uživateli tak vizualizuje proces návrhu domácnosti. [2]

Tento projekt kombinuje 3D vizualizaci s 2D náhledy půdorysů, takže řeší obdobný problém. K tomuto využívá front-end Vue.js a framework pro 3D rendering Babylon.js. IDesigner posloužil jako inspirace pro volbu nástrojů použitých pro konfigurátor rozvodny. Jeden z autorů tohoto projektu věnoval čas sepsání článku, ve kterém popisuje některá chytrá řešení architektury, které se svým týmem využil při práci na projektu. [3]

3.2 Konfigurátor vozu Škoda

Ačkoliv je konfigurace vozu velice komplexní záležitostí, uživatelské rozhraní konfigurátoru výrobce aut Škoda Auto je velice přehledné a uživatelsky přívětivé. Možností doplňkové výbavy automobilu jsou desítky, a autoři konfigurátoru tak rozdělili výběr doplňkové výbavy do několika kategorií, kterými je uživatel provázen formou formuláře. Tento postup byl uplatněn i v konfigurátoru, který je předmětem této práce. Každá změna, která se odlišuje od výchozí konfigurace, se ihned projeví na zobrazované ceně automobilu. Po přihlášení je možné aktuální konfiguraci uložit. Drobným nedostatkem konfigurátoru je to, že nenabízí ovladatelný 3D náhled na vůz, ale změny se projevují jen aktualizací fotografií vozu, které lze přepínat do různých pohledů. [4]

3.3 Brikl

Firma, která se zabývá e-shopy a jako doplňkový produkt nabízí 3D konfigurátor. Paleta možností konfigurace je sice bohatá, ale omezuje se výhradně na jeden konfigurovaný produkt. Zajímavým prvkem je, že software dokáže vyexportovat výrobní

podklady pro šití a obsahuje i vlastní objednávkový systém. Nutno ale dodat, že export se týká pouze barev a textur nanášených na jednotlivé kusy látky, jejichž tvar a velikost je předem definovaná správcem systému. [5]

3.4 Threekit

Tato společnost se zabývá konfigurátorem produktů s vysoce detailním renderováním 3D modelu. Nabízí spolupráci při tvorbě a implementaci vlastních modelů produktů. Konfigurace však spočívá pouze ve změně barev a materiálu textur 3D modelu, jehož tvar se po dobu konfigurace nemění. Jako 3D framework využívá tento produkt Three.js. [6]

4 Technologie pro interaktivní webovou aplikaci

Obsah webových stránek je definován pomocí jazyka HTML (Hypertext Markup Language), kde samotný obsah (nejčastěji text a multimédia) je uzavřen do značek (neboli tagů), které obsah zařadí do příslušné třídy a na základě těchto tříd prohlížeč rozhodne o způsobu vykreslení daného obsahu. [7]

Styly vykreslování obsahu lze dále upravit pomocí jazyka CSS (Cascading Style Sheets), ve kterém lze specifikovat například velikost a styl fontu, polohu obsahu, barevnost, a který je primárně zodpovědný za vizuální stránku webové aplikace. [7]

S prvními dvěma popsanými technologiemi je možné vytvořit obsah statických webových stránek, který je předem definovaný, načten ze serveru a dále se nemění. V dnešní době již tento druh webových stránek téměř vymizel. I když během prohlížení obsah působí staticky a nemění se, na pozadí často stránky ovlivňuje JavaScript. Příkladem může být například panel se souhlasem zpracování cookies třetích stran.

4.1 Skriptovací jazyky

Vyvíjený konfigurátor je interaktivní aplikací, která si nevystačí pouze s HTML a CSS. JavaScript však není jediným použitelným skriptovacím jazykem, když přijde na dynamický obsah webových stránek.

4.1.1 JavaScript

Byl poprvé zveřejněn na konci roku 1995 společností Netscape a dnes se dá považovat za programovací standard v oblasti webových stránek.[8] V každoročním průzkumu (2021) společnosti provozující IT fórum Stack Overflow odpovědělo 64,96% z 83 052 respondentů, že v posledním roce používali pro vývoj JavaScript.[9] To dělá z JavaScriptu v tomto průzkumu nejpoužívanější programovací jazyk na světě.

JavaScriptový engine si vytváří každý distributor prohlížeče sám. Vychází při tom ale ze standartu ECMA[10], čímž je zaručena podpora JavaScriptu u všech moderních a aktuálně nejpoužívanějších prohlížečů. Vyjma klientské části je možné použít JavaScript pro psaní kódu na straně serveru. Nejznámějším prostředím pro spouštění JavaScriptu na serveru je Node.js.

Za nevýhodu může být považováno to, že jazyk je dynamicky typovaný, což znamená, že proměnná může odkazovat na hodnotu jakéhokoliv typu. To může vést ke komplikacím při hledání chyb a obtížnější správě kódu, což se během vývoje aplikace projevilo. V současné chvíli je aplikace napsána právě v JavaScriptu. Tento přístup vychází z toho, že byl JavaScript použit v prvotní fázi vývoje při testování

frameworků pro 3D rendering a některé části kódu tak byly použity i jako základ pro vývoj prezentované aplikace.

4.1.2 TypeScript

Byl zveřejněn společností Microsoft v roce 2012. Podobnost názvů s JavaScriptem není náhodná, protože TypeScript lze považovat za nadmnožinu JavaScriptu a doplňuje tento jazyk o některá vylepšení. Mezi tato vylepšení patří například možnost deklarace typu proměnné a ověřování typů při kompilaci. V průzkumu zveřejněnému na stránkách StackOverflow používá TypeScript pro vývoj 30.19% respondentů a řadí se tak na 7. místo. [9]

Kompatibilita TypeScriptu s moderními webovými prohlížeči je zajištěna procesem transpilace do JavaScriptu. U klienta proto aplikace běží v enginu JavaScriptu. Podpora TypeScriptu je implementována i v Node.js pro použití na straně serveru. Drobnou nevýhodou tohoto jazyka je právě nutnost transpilace, která může vývojáře zpomalovat při psaní kódu. Rovněž deklarace typů a návratových hodnot vede nutně ke zvětšování samotného kódu.

4.1.3 Další skriptovací jazyky

Ačkoliv existují i další skriptovací jazyky, které lze použít pro vývoj aplikace (např. Dart, CoffeeScript, Kaffeine aj.), kvůli kompatibilitě jsou transpilovány do JavaScriptu. Liší se především syntaxí, která se například zkracuje pro lepší efektivitu, nebo tyto jazyky rozšiřují JavaScript o další funkce.

4.2 Front-end technologie

Pro vývoj webových aplikací existuje několik frameworků, které ulehčují práci vývojářům tím, že implementují určitý návrhový vzor a přináší předpřipravené bloky, ze kterých lze skládat webovou aplikaci. Vývojář se tak například nemusí obtěžovat

	W3 counter	Statcounter
Chrome	66,0%	66,0%
Firefox	3,2%	8,5%
Safari	16,8%	9,56%
IE a Edge	5,2%	9,2%
Ostatní	8,8%	6,73%

Tab. 1: Nejpoužívanější internetové prohlížeče za prosinec 2021 podle [11] a [12]. Konfigurátor byl testován na Chrome, Firefox a Edge

implementací logiky pro sdílení dat mezi komponentami webové aplikace, nebo řešit přesměrování při navigaci na single-page aplikaci.

4.2.1 Vue.js

Verze 1.0 byla zveřejněna v roce 2015. Projekt odstartoval Evan You, který dříve pracoval na vývoji frameworku Angular. Oproti dále zmíněným frameworkům se Vue odlišuje tím, že za jeho vývojem nestojí žádná korporace, ale je vyvíjen a sponzorován komunitou. To může být vykládáno jako nevýhoda, jelikož projekt závisí na darech od komunity, a proto může být náchylnější na výkyvy v preferencích této komunity. Přesto byl vzhledem k jeho jednoduchosti a rostoucí oblíbenosti zvolen jako framework pro vývoj konfigurátoru.

Práce s Vue je založena na skládání komponent. Vue komponenta má příponu souboru *vue* a je rozdělena do tří částí vymezených tagy, které definují vzhled, logiku a styl dané komponenty. Komponenta *Hello world* s využitím všech částí logiky Vue vypadá následovně:

```
<template>
    <span class="message">{{msg}}</span>
</template>

<script>
    export default {
        setup(){
            let msg = "Hello World!";
            return { msg }
        }
    }
</script>

<style>
    .message{
        font-size: 14px;
    }
</style>
```

Důležitou knihovnou používanou v aplikaci konfigurátoru je Vuex. Jedná se o oficiální rozšíření Vue.js a slouží jako single source of truth (SSOT nebo SPOT) pro celou aplikaci. Spravuje přístup komponent k datům a umožňuje sdílení dat napříč komponentami. V zásadě se jedná o úložiště, které sleduje změny svého obsahu a na tyto reaguje vysláním signálu komponentám, které na těchto datech závisí. Komponenty pak podle změn aktualizují vykreslovaný obsah.

Nejdůležitější proměnnou uloženou v tomto modulu Vue je pole *bays*, do kterého se vkládá reprezentativní objekt každého modulu rozvodny, který se vyskytuje v 3D prostředí. Reprezentativní objekt obsahuje v první řadě data formuláře, podle

kterých je seskládán 3D model. Dále pak obsahuje data, která souvisí s pozicováním vkládaného modulu. Také jsou zde uloženy reference na obrázky, ze kterých se skládá náhled jednopólového schématu a data nutná pro nacenění produktu. Před zpracováním formuláře tento objekt vypadá následovně:

```
{
    info: {
        unique_id: null,
        component_tree: {}
    },
    positioning: {
        posX: 0,
        posY: 0,
        posZ: 0,
        rotX: 0,
        rotY: 0,
        rotZ: 0
    },
    core: {
      bay_type: "diameters",
      phase_distance: "1040",
      mre: "mre11",
      cb: "standard",
      ct_north: "true",
      ct_north_cores: 2,
      ct_south: "true",
      ct_south_cores: 2,
      exits: 1,
      busbar: "",
    },
    exit_1: {
      side: "north",
      configuration: "es ds faes",
      measurement: [],
      offset_straight: 5,
      offset_side: 0,
      termination: "hk",
      termination_side: "straight"
    },
    exit_2: {
      side: "south",
      configuration: "es ds faes",
      measurement: [],
      offset_straight: 5,
      offset_side: 0,
      termination: "hk",
      termination_side: "straight"
    },
```

```
    sld: {
        rendered_slds: []
    }
}
```

4.2.2 React

React je JavaScriptová knihovna vyvíjená společností Meta (dříve Facebook). Jako open-source byl na GitHub přidán v roce 2013. Knihovna je napsána v jazyce JavaScript a pro své uživatelské rozhraní ji používají aplikace jako Facebook nebo Instagram, které se pravidelně umísťují v první desítce nejnavštěvovanějších webů. [13]

Ačkoliv React umožňuje komponenty definovat pomocí JavaScriptu, preferovanou volbou je syntaxe zvaná JSX. Tato syntaxe je rozšířením JavaScriptu, které umožňuje při vývoji kombinovat JavaScript a HTML. Díky tomuto přístupu jsou komponenty definované jako funkce, jejichž návratovou hodnotou je HTML. Pokud má mít komponenta vlastní logiku, dosáhne se toho importem příslušné knihovny, která zajistí reaktivnost prvků komponent.

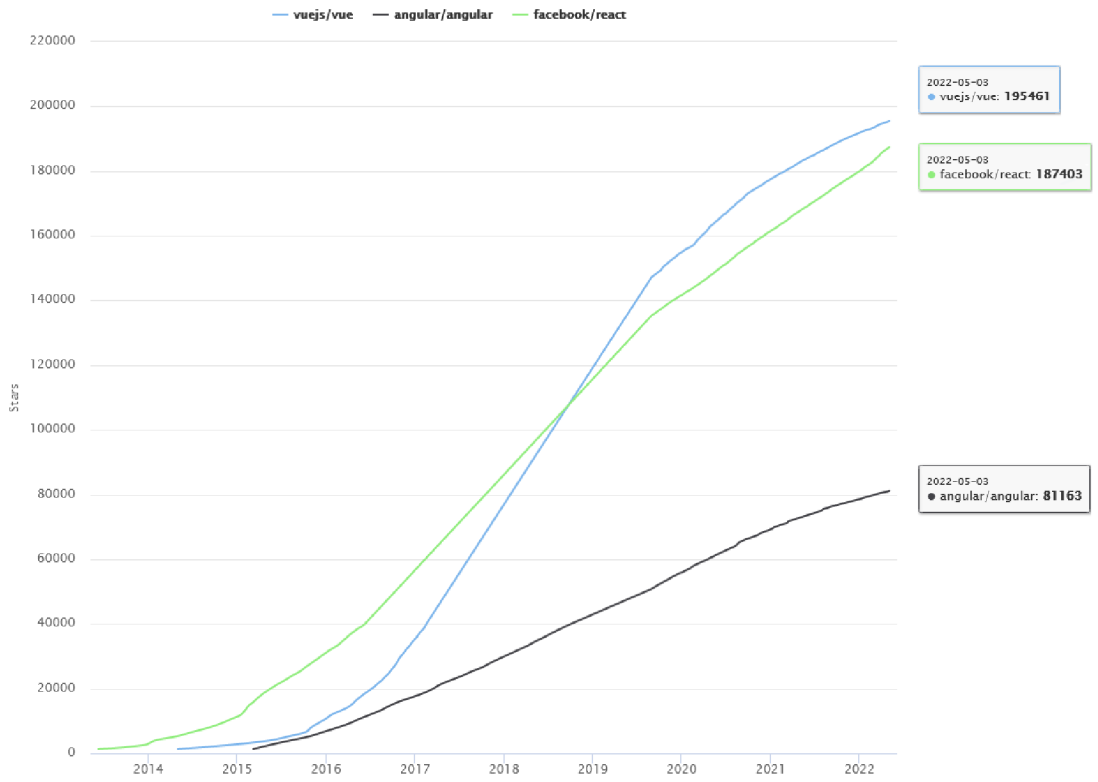
Možnost výběru a rozmanitost knihoven jsou jedny z hlavních rysů, které definují React. Velmi aktivní komunita uživatelů knihovny totiž stojí také za množstvím open-source balíků, které mohou rozšířit funkci aplikace o téměř cokoliv, s čím se při vývoji programátor může setkat.

Tento modulární přístup sdílí s Vue, ale přínos komunity Vue zatím není tak znatelný jako u Reactu. Oproti tomu Angular se snaží implementovat většinu potřebných balíků sám. [14]

4.2.3 Angular

Tento framework byl poprvé zveřejněn v roce 2016 a za jeho vývojem stojí Google. Stejně jako u předchozích frameworků se jedná o open-source. Ačkoliv by se mohlo z uvedených dat zdát, že se jedná o nejmladší z uvedených frameworků, není to úplně pravda. Angular vychází ze svého předchůdce AngularJS, který byl poprvé uveřejněn už v roce 2010. Vývojáři v Googlu se však rozhodli, že celý projekt vystaví znovu od základu. V lednu roku 2022 Google oficiálně ukončil podporu AngularJS a doporučil uživatelům migrovat své aplikace do Angularu (někdy též uváděného jako Angular 2+). [15]

Pro vývoj aplikací v tomto frameworku se primárně používá TypeScript. Použití JavaScriptu sice není vyloučeno (vychází ze vztahu mezi TypeScriptem a JavaScriptem), ale oficiální dokumentace uvádí výhradně příklady napsané v TypeScriptu.



Obr. 1: Vývoj oblíbenosti uvedených frameworků na základě GitHub stars. Vygenerováno pomocí [16].

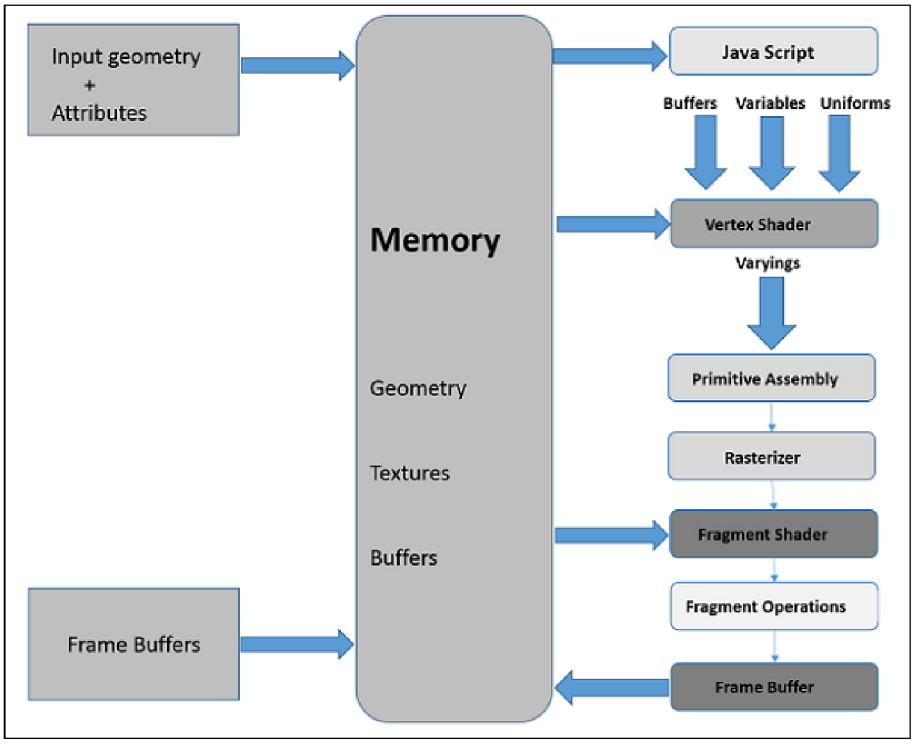
4.3 WebGL

Hlavní součástí konfigurátoru je vykreslování prostorového modelu rozvodny. O zobrazování trojrozměrných modelů nebo i 2D grafiky se na webu stará knihovna WebGL (zkratka pro Web Graphics Library), která je součástí standardu HTML5 implementovaného v moderních webových prohlížečích. Volba jiného 3D enginu by vedla k nutnosti instalace přídavných pluginů na straně uživatele, což by výrazně snižovalo uživatelskou přívětivost aplikace a neplnilo tak požadavek na její dostupnost. V minulosti byl pro vývoj interaktivních webových rozhraní používán například Adobe Flash Player nebo Microsoft Silverlight, jejichž používání bylo podmíněno právě instalací vhodných pluginů do prohlížeče. Dnes už funkcionalitu těchto pluginů nahradily nástroje implementované ve standardu HTML5. [17]

WebGL je nízkoúrovňová API napsána v JavaScriptu. Vychází z OpenGL, jedná se tedy o hardwarově akcelerovaný rendering – pro výpočty je využíván i grafický procesor klienta.

Ve WebGL je každý vykreslovaný objekt reprezentován body (vrcholy), které svojí polohou určují tvar objektu a nastaveným materiálem ovlivňují i jeho barev-

nost. Jednou z prvních operací prováděných na GPU se anglicky říká *vertex shading*. Během této operace jsou body modelu propojeny hranami tak, že vznikne trojúhelníková síť kopírující tvar objektu. Poté dojde k rasterizaci, čímž je umožněno použití maticových operací, které podle natočení normály každého z trojúhelníku vůči paprskům zdroje světla a také podle gradientu barev mezi vrcholy každého trojúhelníku přiřadí každému pixelu barvu a jas. Této operaci se říká *fragment shading* a může zahrnovat i další operace pro vylepšení obrazu, jako třeba vyhlazování hran. Popsaným způsobem vznikne jeden snímek, který je nahrán do bufferu, jehož obsah je poté postupně vykreslován do HTML elementu *canvas*.



Obr. 2: Proces vykreslování objektů ve WebGL. Převzato z [18].

4.3.1 Frameworky pro WebGL

Ačkoliv je možné k WebGL API přistupovat přímo v prohlížeči, existuje několik frameworků, které můžou práci s WebGL výrazně ulehčit.

Babylon.js

Framework vzniknul jako open-source projekt zaměstanců Microsoftu pod vedením Davida Catuhe. [19] Na GitHubu byl uveřejněn v roce 2013. Zdrojový kód je psaný v TypeScriptu, ale JavaScript je podporován na úrovni dokumentace i příkladů. Autor frameworku uvádí, že z velkých firem používají Babylon.js korporáty jako Adobe nebo Microsoft. [20]

Velmi užitečnou součástí projektu je Babylon.js Playground, což je webové rozhraní s textovým editorem a zobrazovací částí, jehož funkce je podobná se službami jako JSFiddle nebo CodeSandbox. Jde o online vývojové prostředí, na které často odkazuje dokumentace s příklady, a které umožňuje rychle a jednoduše testovat framework, případně replikovat problém a sdílet na oficiálním fóru s velmi aktivní a nápomocnou komunitou. Vývojové prostředí se dá přepínat mezi TypeScriptem a JavaScriptem a všechny oficiální příklady jsou zde uvedené v obou programovacích jazycích.

Dokumentace je velmi detailní a perfektně přehledná s množstvím odkazů na příklady v Babylon.js Playground. Z důvodu přehlednosti dokumentace, množství příkladů a oficiálních tutoriálů byl pro vývoj zvolen právě tento framework. Babylon.js je navíc vzhledem k omezenějším možnostem nastavení renderingu vhodnější pro rychlý vývoj, protože se u vývojáře nepředpokládají hlubší znalosti z oblasti 3D renderingu.

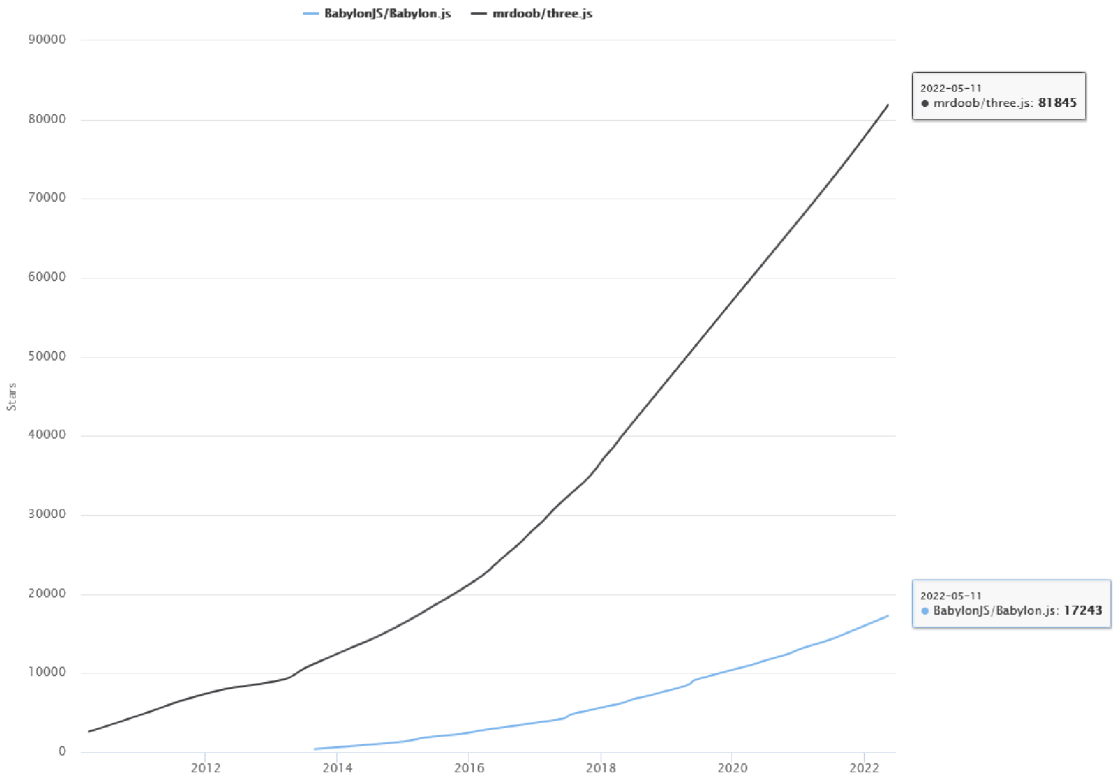
Three.js

Starší z frameworků byl poprvé zveřejněn na GitHub v roce 2010 Ricardem Cabello. Stejně jako Babylon.js využívá tento framework WebGL a HTML5. Zdrojový kód je napsán v JavaScriptu. Ačkoliv lze Three.js doplnit o TypeScript doinstalováním balíku s definicemi typů, nejedná se o jeho oficiální podporu.

Syntaxe framewroků Three.js a Babylon.js se na první pohled podobají. Three.js ale jde v některých věcech více do hloubky. Například dává vývojáři větší kontrolu nad volbou a nastavením shaderů. Umožňuje také větší zásahy do procesu renderingu WebGL. Toto bude pravděpodobně také jeden z hlavních důvodů, proč je Three.js daleko oblíbenější než Babylon.js. Množství možností nastavení sice umožňuje lépe optimalizovat procesy aplikace, ale po vývojáři se také vyžadují hlubší znalosti z oblasti 3D renderingu. Dokumentace Three.js sice obsahuje vše potřebné, ale odkazy na příklady u daných témat chybí. Příklady lze potom najít v oficiálním repozitáři na GitHubu, jde ale pouze o neokomentované úryvky kódu.

4.4 Back-end technologie

Pro vývoj serverové části aplikace byla zvolena platforma Docker. Ta umožňuje spouštět služby jako izolované kontejnery a díky virtualizaci je možné tyto služby nasadit na libovolné platformě. Kontejnery jsou založeny na některé z odlehčených distribucí linuxového systému, na kterých jsou nainstalovány potřebné aplikace s nastavenou výchozí konfigurací. Moderní služby často závisí na několika různých aplikacích. To znamená při vývoji také čas nutný na jejich správnou konfiguraci. Pomocí Dockeru lze ale distribuovat kompletní službu zabalenou jako kontejner, a tak



Obr. 3: Vývoj oblíbensti uvedených frameworků na základě GitHub stars. Vygenerováno pomocí [16].

umožňuje vývojáři věnovat více času na konfiguraci samotné služby místo závislých aplikací.

4.4.1 Strapi

Strapi je systém pro správu obsahu, anglicky označovaný zkratkou CMS (content management system). Mezi nejznámější zástupce těchto systémů patří například WordPress nebo Drupal. Strapi pak představuje specifickou odnož těchto systémů, označovanou jako headless. Narozdíl od tradičních CMS, kde serverová část vrací klientovi již zpracované HTML, headless CMS se nezabývá vykreslováním, ale pouze skrze API zprostředkovává strukturovaný obsah, zpravidla ve formátu JSON. Prezentační vrstva je pak zcela pod kontrolou klientské aplikace, v našem případě napsané v jazyce JavaScript. Výhoda tohoto přístupu spočívá v možnosti jednoduchého sdílení dat mezi různými platformami, což v budoucnu ulehčí vývoj dalších nástrojů a přispívá k možné škálovatelnosti aplikace. Dále byla tato technologie zvolena z důvodu rychlosti implementace. Přichází totiž s předpřipravenou databází, REST API a jednoduchým grafickým rozhraním pro správu obsahu. Zvolena byla

verze pro Docker s databází SQLite. Z dalších databázových systémů byly na výběr PostgreSQL, MySQL nebo MariaDB.

Struktura databáze

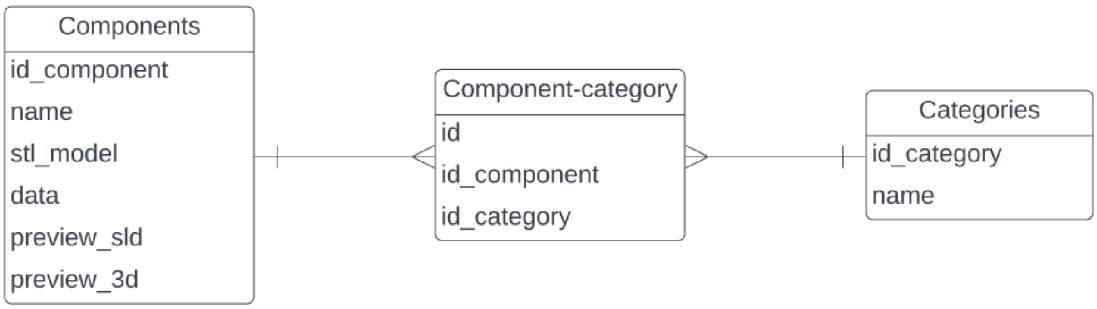
Ačkoliv je možné uvnitř kontejneru přistupovat přímo k tabulkám databáze i upravovat záznamy v ní, během vývoje nenastala situace, kdy by bylo nutné do struktury tímto způsobem zasahovat. Pro používání tohoto CMS tak není vůbec nutná znalost jazyka SQL. Správa databáze se provádí přes uživatelsky přívětivé webové rozhraní. Strapi umožňuje vytvořit datový typ kolekce, která obsahuje jednotlivá pole dalších různých datových typů. Kolekce se tedy dá chápat jako tabulka.

Kolekce *Components*

Tato kolekce slouží k uchování všech komponent, které jsou nutné k poskládání celé rozvodny. Kolekci definují položky *name*, *stl_model*, *data*, *preview_sld* a *preview_3d*. Položka *name* slouží k jednoznačné identifikaci dané komponenty a pro celou kolekci je unikátní. Položka *stl_model* uchovává odkaz na stažení příslušného 3D modelu komponenty. Položka *data* obsahuje JSON strukturu, která obsahuje další podrobnosti a taky uchovává polohy bodů, kde mají navazovat další součásti. *Preview_3d* je volitelná položka a umožňuje uložit náhled na komponentu ve standardních obrázkových formátech. *Preview_sld* umožňuje indexovat více souborů a tyto soubory jsou obrázky ve formátu *svg*, které reprezentují danou komponentu v náhledu na jednopólové schéma rozvodny.

Kolekce *Categories*

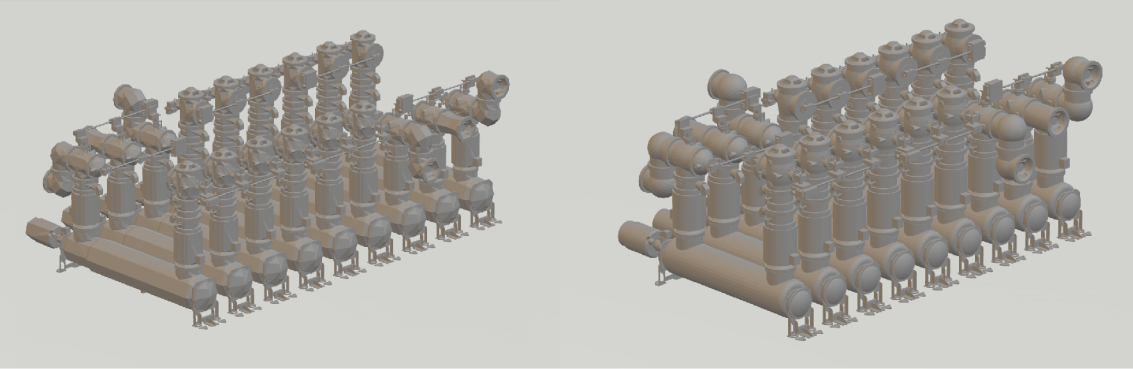
V této kolekci jsou uvedeny kategorie komponent. Jedna komponenta může být součástí několika kategorií. Tato struktura kolekce je důležitá pro výběr správných komponent při skládání sestavy rozvodny, jak bude uvedeno dále.



Obr. 4: E-R diagram implementované databáze.

4.5 Vytváření 3D modelů pro konfigurátor

Modely jednotlivých komponent bylo možné exportovat z 3D softwaru, který je standardně využíván pro skládání rozvoden ve firmě Hitachi Energy. Tento software je založen na platformě Creo Parametric a umožňuje export komponent do formátu *stl*. Modely v tomto formátu jsou reprezentovány pomocí sítě trojúhelníků, takže detaily modelu při špatném nastavení parametrů exportu zanikají. U zmíněného softwaru jsou ale parametry nastavení sítě nešikovně spjaty s velikostí exportovaného modelu. Čím je model větší, tím méně detailní *stl* model je možné vyexportovat, a to i v případě, kdy se snažíme exportovat jedinou komponentu sestavy. Dosažené výsledky detailů exportovaných modelů tak neodpovídaly požadované kvalitě. Z tohoto důvodu byly exportovány pouze jednotlivé komponenty a dílčí sestavy, používané v konfigurátoru, byly poté skládány v open-source modelovacím programu Blender.



Obr. 5: Porovnání struktury modelů při exportu ze standardně užívaného softwaru (vlevo) a ze softwaru Blender (vpravo).

Některé komponenty navíc byly vyexportovány se špatnou pozicí počátečního bodu modelu a v Blenderu je možné polohu tohoto počátečního bodu (*origin*) upravit. Poloha tohoto bodu je důležitá z pohledu skládání modelů do sestav, jelikož od tohoto počátku jsou odměřovány polohy bodů, kam jsou vkládány navazující součásti.

Babylon.js umožňuje importovat celou scénu ve standardizovaném formátu *gltf*, případně pak v proprietárním formátu *babylon*, pro jehož export scény existuje plugin do programu Blender. Dále pak umožňuje import jednotlivých 3D objektů ve formátu *obj* nebo *stl*.

Při importu celé scény je možné vypnout zobrazení vkládaných modelů. Tento přístup by znamenal stažení celé knihovny s potřebnými modely při spuštění aplikace. Výhodou by ovšem byla možnost vkládání objektů do aktivní scény v podobě instancí. Využití instancí vede k optimalizaci výpočetní náročnosti. Instance totiž

sdílejí např. textury a materiály s primárním modelem. Procesoru tak na jejich vykreslení stačí jedno volání grafického procesoru (GPU) [21]. Tento proces by ale znamenal zpomalení startu aplikace z důvodu stahování knihovny, navíc scéna by obsahovala i modely, které uživatel nevyužije. Editace importované scény je také náročnější na správu v případě editace komponent. Rovněž by zůstávala nutnost databáze komponent, jelikož ta kromě modelů obsahuje i další metadata nutná pro pozicování komponent a vyhodnocení cen.

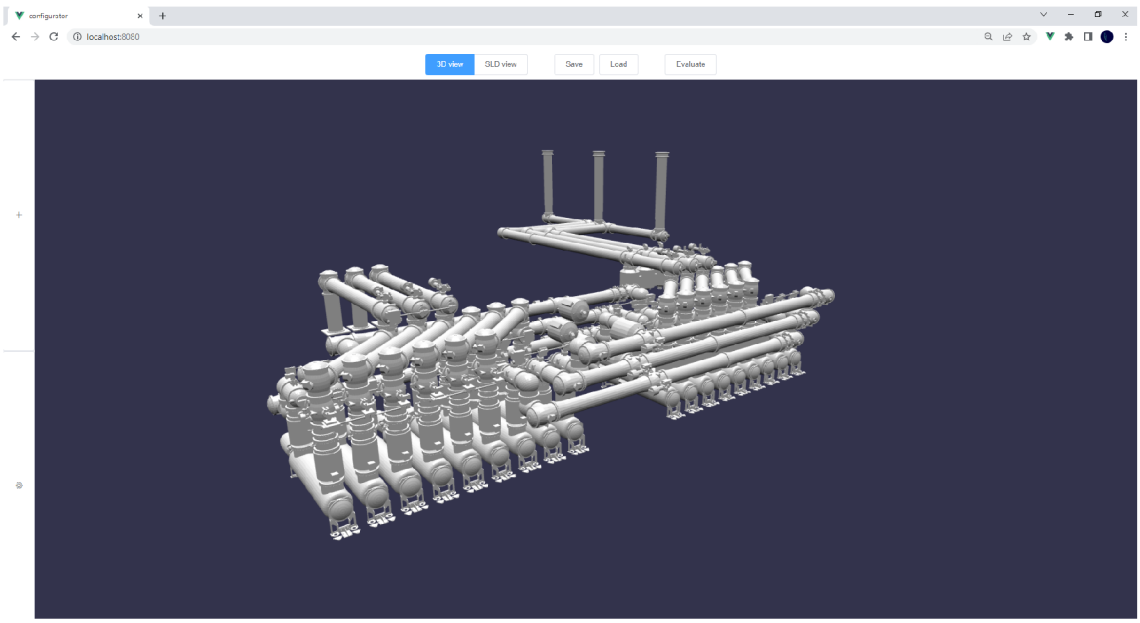
Proto byl nakonec zvolen přístup importu samostatných komponent (ve formátu *stl*) ve chvíli, kdy jsou potřeba. Během testování navíc nebyl pozorovaný zásadní pokles FPS a to i v případě rozsáhlých sestav.

5 Konfigurátor rozvodny

Aplikace aktuálně umožňuje konfigurovat jeden layout rozvodny z produktové řady pro napěťovou hladinu 550 kV. Součástí je pak zobrazení 3D modelu a náhled na jednopólové schéma. Další součástí aplikace je modul pro generování seznamu komponent a jejich nacenění. Celý projekt je možné kdykoliv uložit na disk klienta.

5.1 Uživatelské rozhraní

Aplikace je rozčleněna do tří částí. V horní liště je přepínač mezi pohledem na 3D model rozvodny a náhledem na její jednopólové schéma. Dále zde nalezneme tlačítka pro uložení rozvodny a načtení rozvodny ze souboru. Poslední tlačítko v horní liště otevře okno se seznamem komponent a jejich naceněním. Boční lišta obsahuje dvě tlačítka. Kliknutí na tlačítko se symbolem + otevře dialogové okno s formulářem, ve kterém má uživatel možnost nastavit parametry rozvodny a vložit tzv. *bay* (základní stavební modul rozvodny). Druhé tlačítko se symbolem pro nastavení otevře *Bay configurator*, ve kterém se po vložení modulu zobrazí informace o poloze a natočení a také je zde možnost editace vloženého modulu. Zbytek okna vyplňuje část s pohledem na 3D model rozvodny, případně s náhledem na jednopólové schéma.



Obr. 6: Uživatelské rozhraní s 3D náhledem na dva vložené moduly.

5.2 Přidání modulu do scény

Uživatel přidá model do scény vyplněním formuláře, který je rozdělen do dvou částí. První se týká primárně základních komponent, které se neprojevují v samotném modelu. Jde o MRE (maintenece, repair & extension), které souvisí se zajištěním bezpečnosti a provozuschopnosti rozvodny při technických zásazích v případě poruchy nebo údržby. Toto je zajištěno záměnou komponent, které rozdělují vnitřní prostor rozvodny na několik plynotěsných segmentů. Tato změna se projeví pouze při naceňování rozvodny. Změna odpojovače se týká také pouze komponent umístěných uvnitř modelu.

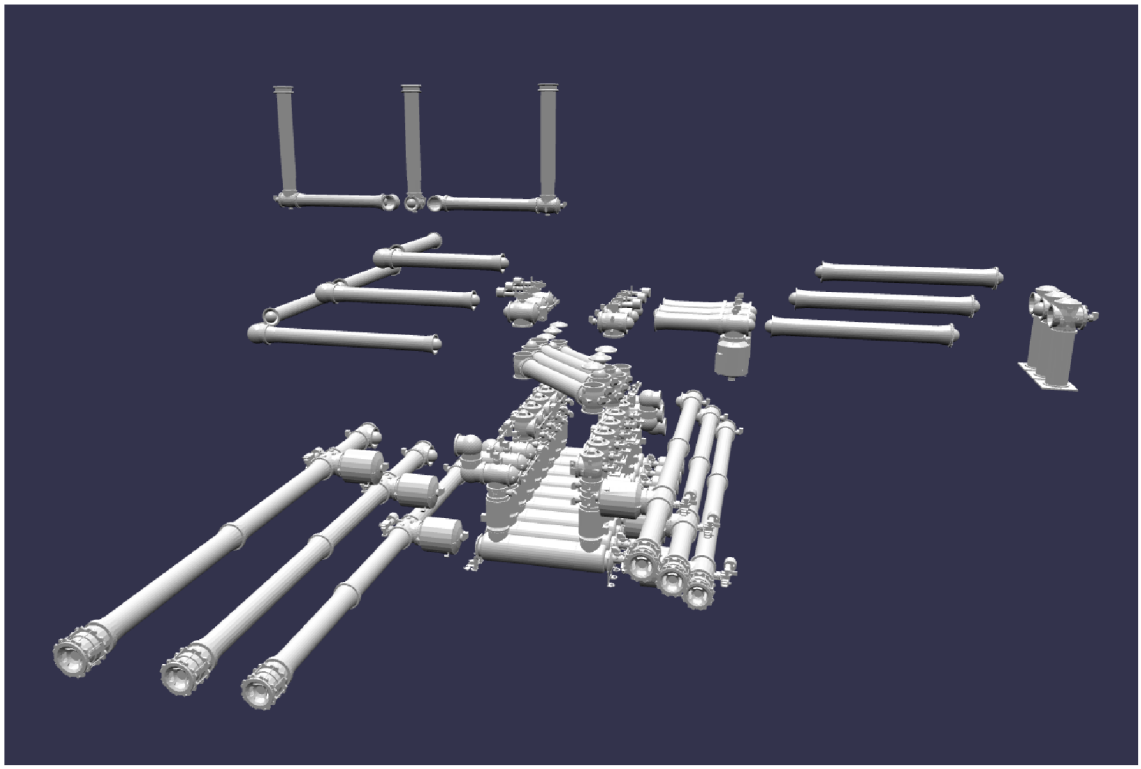
Změna modelu proběhne pouze v případě, že se uživatel rozhodne nezařadit do okruhu zařízení pro měření proudu. V takovém případě je toto zařízení v modelu nahrazeno prostým zapouzdřeným vodičem. Pokud se rozhodne toto zařízení použít, může navolit počet měřicích cívek. Tato změna se projeví rovněž v odpovídající části jednopólového schématu.

Na druhé stránce formuláře uživatel volí *exity*, tato součást slouží k napojení celé rozvodny na okolní sítě VVN. Na jeden modul je možné přivést maximálně dvě přípojky. Často ale zákazník obě přípojky nevyužije, nebo nevyužije vůbec žádnou a počítá s rozšířením přípojných míst až v budoucnosti. Toto je ve formuláři zohledněno. Přípojka může obsahovat součást sloužící k odpojení dané větve od rozvodny, nebo zařízení pro měření napětí v síti. Oboje vede ke změně součástí v modelu i ve schématu. Kromě toho si uživatel může navolit vzdálenost zakončení od rozvodny včetně zahnutí celé přípojky do strany a také stranu rozvodny, na kterou bude přípojka napojena. Implementovány jsou dvě možnosti zakončení, a to buď s napojením na kabelovou síť, nebo s napojením na síť vzdušného venkovního vedení. Potvrzením formuláře započne skládání rozvodny, které se viditelně projevuje ve 3D náhledu.

5.3 Sestavení modulu z komponent

Počet komponent jednoho modulu je příliš velký na to, aby se do prostředí vkládaly jako samostatné 3D objekty. V závislosti na konfiguraci modulu se jedná o přibližně 500 komponent na jeden modul. Pokud by byl zvolen tento přístup, zvedla by se výrazně náročnost na vykreslování těchto komponent, což by mělo vliv na plynulost celé aplikace. Každá komponenta musí být také před vložením stažena ze serveru, takže by se zvedl také počet volání na server a množství přenesených dat.

Místo tohoto přístupu byl zvolen postup, kdy je hierarchie na sebe navazujících součástí zjednodušena, a tak se do scény vkládají některé části sestavy jako jeden 3D model.



Obr. 7: Pohled na jeden modul rozvodny. Sestava je rozdělena do jednotlivých vkládaných modelů.

Každý modul rozvodny je na straně klienta reprezentován objektem, který v sobě kromě dat formuláře obsahuje také další informace, jako je pozice modulu ve scéně, nebo unikátní identifikátor *root nodu*, na který jsou hierarchicky skládány modely. Tento unikátní identifikátor je získán přímo z frameworku Babylon.js a dále je využíván pro jednoznačnou identifikaci modulu v aplikaci. Pozicování navazujících komponent je součástí objektu stahovaného spolu s modelem. Polohy a orientace míst, kde navazují další modely, jsou relativní vůči počátku modelu, na který je součást vkládána. Po nahrání nového modelu do scény jsou všechny tyto informace z databáze připojeny k 3D modelu do objektu *Component*. V tomto objektu je také vytvořena funkce pro vkládání dalších komponent:

```
class Component {
    constructor(mesh, json_data){
        Object.assign(this, json_data);
        this.mesh = mesh;
    }
    assembleComponent(newmesh, parentFlangeId){
        this.mesh.addChild(newmesh);
        let parent_flange = this.partFlanges.find(fl => {
            return fl.flangeID === parentFlangeId
        })
```

```
11          newmesh.position = new Vector3(
12              parent_flange.posVector[0],
13              parent_flange.posVector[1],
14              parent_flange.posVector[2]).scale(MODEL_SCALE);
15          newmesh.rotation = new Vector3(
16              parent_flange.rotVector[0] * Math.PI,
17              parent_flange.rotVector[1] * Math.PI,
18              parent_flange.rotVector[2] * Math.PI);
19      }
20  }
```

Vyplnění formuláře zavolá funkci, které jsou data z formuláře předána jako JSON objekt. Tato funkce následně vybírá podle dat z formuláře správné 3D modely a přidává je postupně do scény. Jelikož ta samá funkce slouží také k editaci existujících modulů rozvodny, musí nejdříve rozhodnout, zda se jedná o nový modul, nebo půjde o editaci již existujícího modulu. Tato informace lze zjistit z předaného objektu, kde by v případě editace existujícího modulu byl vyplněn unikátní identifikátor. Pokud není vyplněn, je v prostředí Babylon.js vytvořen *root node* a jeho unikátní identifikátor je přidán do JSON objektu, který modul reprezentuje.

Skládání modulu a přidávání těchto komponent do scény lze rozdělit na 5 hlavních částí. Části s připojením na okolní vysokonapěťové sítě jsou navíc ještě dále konfigurovatelné:

1. Diametr
2. Bridge
3. Exit 1
   (a) Zařízení pro odpojení větve od sítě
   (b) Měření napětí
   (c) Délka a zabočení větve do strany
   (d) Zakončení přípojky
4. Exit 2
   (a) Zařízení pro odpojení větve od sítě
   (b) Měření napětí
   (c) Délka a zabočení větve do strany
   (d) Zakončení přípojky
5. Přípojnice *(bus bar)*

Diametr je nejkomplexnější vkládaný modul. Jeho součástí jsou výkonové vypínače, nejrůznější měřicí zařízení a jeho součástí je také napojení na ostatní moduly rozvodny pomocí přípojnice. Počet jednotlivých 3D modelů, ze kterých se diametr skládá, převyšuje 300. Reálně se však vkládá do scény jako jeden model a uživatel konfigurací vybere jednu ze 4 možných podob diametru.

Následuje přidání komponenty *bridge*, která spojuje další výkonové vypínače diametru a na kterou jsou poté osazovány *exity*, tedy větve s napojením na okolní sítě VVN. Nevyužitá přípojná místa jsou případně zaslepena příslušnou komponentou.

Pokud modul má být připojen k okolním sítím VVN, začne se na příslušném místě komponenty *bridge* skládat přípojná větev. V první řadě je na *bridge* vložen modul odpojovače větve, případně pouze jejího uzemnění. Na tento modul pak může navazovat část pro měření napětí na větvi. Samotná větev se skládá z obyčejných koaxiálních vodičů, které mají stanovenou maximální výrobní délku, a tak jsou vodiče v případě potřeby spojovány. Délka větve je plně uživatelsky definovatelná v rozsahu od 0,8m do 50m délky kolmé na rozvodnu a větev může také zabočit kolmo do obou stran v tom samém rozsahu.

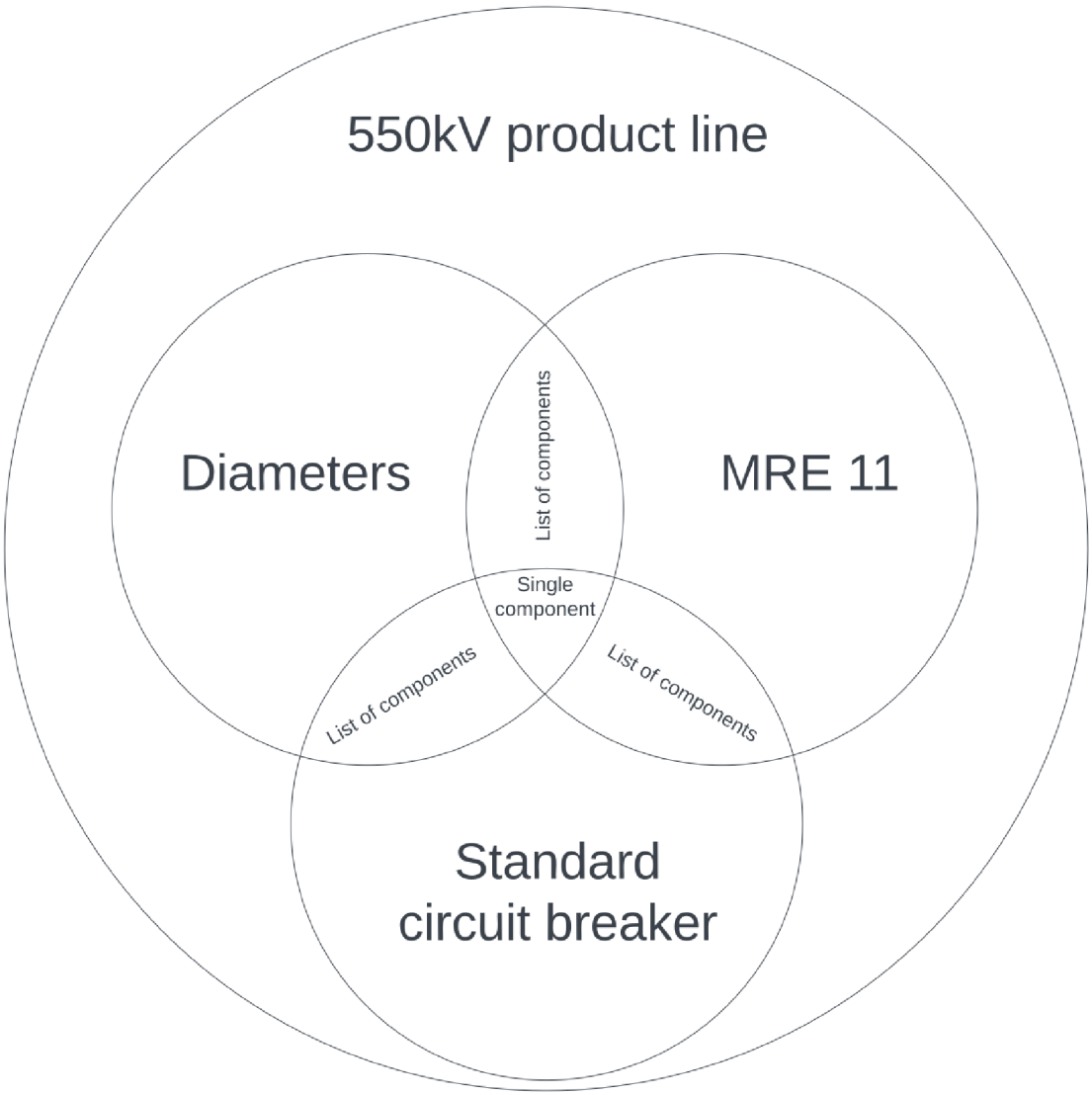
Vzhledem k této variabilitě není možné vytvořit model pro každý vodič zvlášť, proto je tato komponenta vytvořena až na straně klienta. Babylon.js umožňuje upravit měřítko kteréhokoliv objektu ve scéně podél jedné ze zvolených os kartézského souřadného systému modelu. Součást je tedy v databázi rozdělena na 3 části. Prostřední část vodiče může být zvětšována či zmenšována podle podélné osy, aniž by se na výsledné součásti projevila viditelná deformace. Poté, co je prostřední část zvětšena na potřebnou délku, jsou k této součásti připojeny příruby a modely jsou sloučeny do jednoho.

Pokud je délka větve větší než maximální výrobní délka vodiče, jsou tyto pospojovány a na poslední součást je připojeno jedno ze dvou zakončení (kabelová přípojka nebo napojení na vzdušné vedení VVN). Stejným postupem je na *bridge* napojen případný *exit 2*.

Pokud je do scény přidán druhý modul, musí být propojen s prvním modulem pomocí přípojnice *(bus bar)*. Ta je vždy součástí předchozího modulu. První dva vložené moduly jsou vždy propojeny přípojnicí se segmentem měření napětí. Další moduly jsou pak už pospojovány pomocí obyčejných vodičů, stejných jako ze kterých se skládají případné větve rozvodny.

Uživatel může vyplněním formuláře vytvořit až 28 různých variant vkládaného modulu, které se viditelně projeví ve scéně. Do budoucna se počítá s implementací dalších produktových řad a dalších layoutů, a tak zde hrozí riziko kombinatorické exploze.

Výběr správné komponenty pro vložení do scény lze znázornit jako průnik množin, kde každou položku v kolekci *Categories* lze chápat jako množinu a jednotlivé komponenty reprezentují prvky těchto množin. Při správné kategorizaci komponent pak průnik množin kategorií vytvoří množinu pouze s jedním prvkem – výslednou komponentou. Správné zařazení komponent do kategorií je stěžejní pro správné fungování konfigurátoru a musí jej tak provádět člověk, který je seznámen s vnitřní logikou aplikace.

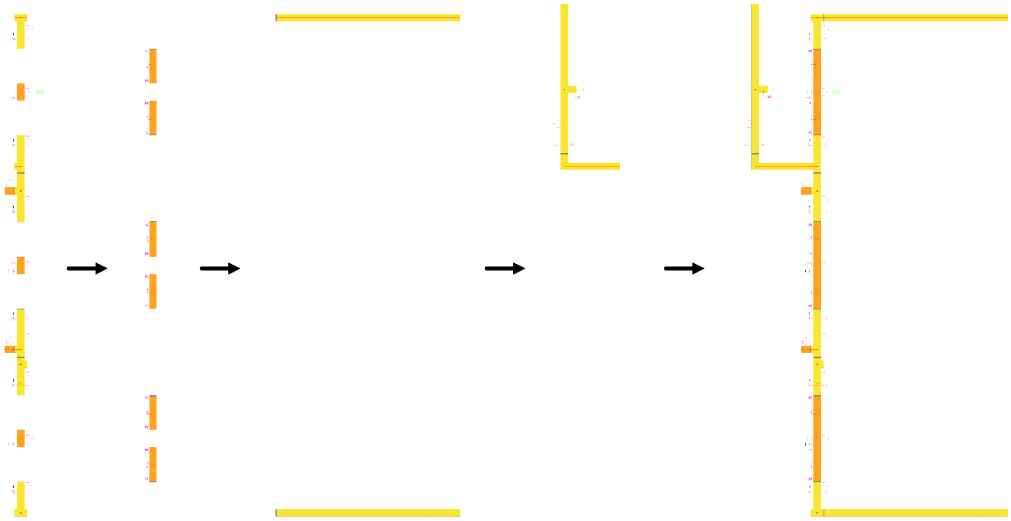


Obr. 8: Znázornění výběru komponenty pomocí průniku množin.

5.4 Vytváření jednopólového schématu

Jednopólové schéma (single-line diagram) je zjednodušená výkresová dokumentace, která zobrazuje jednu fázi rozvodny. Jsou v něm zaznačeny důležité komponenty a jejich vzájemná návaznost. Pohled na toto schéma lze přepnout tlačítkem v horní liště. Každému vloženému modulu odpovídá jedna karta, do které se vkládají schématické značky jednotlivých komponent.

Jako obrazový formát těchto značek byla zvolena vektorová grafika s příponou souboru *svg*. Výhod má tento formát hned několik. První spočívá především v jednoduchosti implementace do webového prostředí. V jazyce HTML5 totiž existuje pro tento formát vlastní tag a navíc lze tyto obrázky dále seskupovat a tyto skupiny



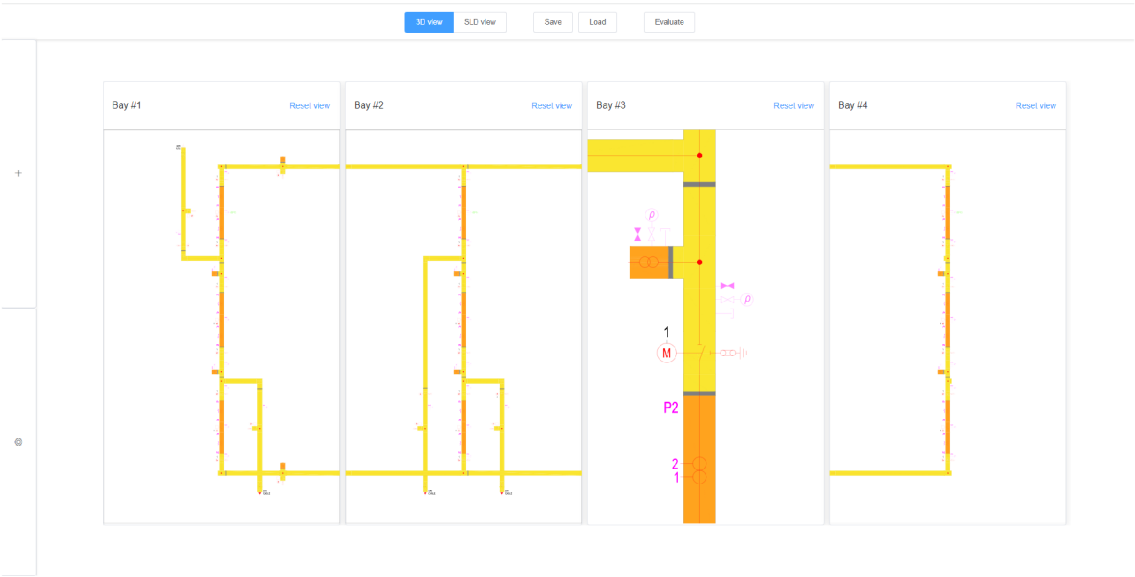
Obr. 9: Proces vrstvení SVG obrázků při vytváření schématu

poté libovolně škálovat a dosáhnout tak efektu zoomu. Jelikož se jedná o vektorovou grafiku, neztrácí obrázek při zvětšování kvalitu detailů. Další výhodou je velikost obrázku ve formátu *svg*, která je oproti rastrovým formátům často výrazně menší. Ačkoliv toto tvrzení není pravidlem, v případě obrázků použitých v konfigurátoru toto skutečně platilo.

Použité značky byly vytvořeny pomocí firemní knihovny bloků pro AutoCAD. Ačkoliv se jedná o software, který pracuje zejména s vektorovou grafikou, nelze z tohoto programu exportovat *svg* přímo. Nejdříve bylo nutné proto exportovat obrázky z AutoCADu do formátu *pdf* a poté v programu Inkscape převedeny do finálního formátu *svg*.

Během vkládání 3D modelů a sestavování modulu se do pole v JSON objektu, reprezentující daný modul, souběžně přidávají reference na příslušný obrázek jednopólového schématu. Vykreslování je zajištěno pomocí Vue.js. To sleduje proměnnou *bays* uloženou v centrálním úložišti Vuex. Pokud se v proměnné objeví objekt modulu, je pro něj automaticky vytvořena karta s komponentou připravenou na vykreslování *svg*. Tato komponenta automaticky vykreslí veškerý obsah pole s referencemi na obrázky. Všechny vykreslované obrázky jsou umístěny do počátku elementu *<svg>*. Pozicování je předpřipraveno a vyexportováno už z AutoCADu. Obrázky komponent jsou poté vrstveny na sebe.

Jakákoliv změna modulu je pevně navázána na objekt uložený v úložišti Vuex. Podstatou tohoto úložiště je hlídat změny a informovat o nich každou komponentu,



Obr. 10: Ukázka rozhraní v režimu náhledu single-line diagramu s vloženými čtyřmi moduly. U třetího modulu je zvětšen detail diagramu.

která na datech z úložiště závisí. Komponenty pak na změny reagují tím, že se aktualizují. Tak je zajištěna synchronizace mezi 3D a 2D grafickým prostředím.

5.5 Ukládání a načítání rozvodny

Jelikož všechna data potřebná k poskládání modulu jsou uložena v reprezentativním objektu, stačí uložit jen tyto objekty. Po kliknutí na tlačítko *Save* dojde uložení rozvodny ve formátu JSON do výchozí složky pro stahování. Prakticky jde pouze o export pole *bays* z centrálního úložiště do JSON souboru. Během exportu tohoto pole jsou z reprezentativních objektů modulů vymazány unikátní identifikátory, čímž je zajištěno, že při načítání souboru bude aplikace s daty pracovat stejně, jako by vytvářela nové moduly. Tento přístup také přináší možnost nahrát uložené moduly k již rozpracované rozvodně.

Pro načtení uložené rozvodny klikne uživatel na tlačítko *Load*. Po rozkliknutí tohoto tlačítka se objeví dialogové okno, ve kterém uživatel vyhledá soubor na disku a po potvrzení se rozvodna podle načtených dat znovu poskládá. Postupným načtením lze také sestavit rozvodnu z více různých uložených souborů. Načtení ze souboru totiž nevymaže aktuální stav aplikace.

5.6 Nacenění výsledného produktu

Komponenty pro implementovanou produktovou řadu vyrábí celkem 5 závodů. Aktuálně to jsou závody v těchto zemích: Švýcarsko (CH), Čína (CN), Česko (CZ), Indie (IN) a Bulharsko (BG). Tyto závody obsluhují regiony Evropa, Indie a Jihovýchodní Asie. Zároveň platí, že inventář komponent, které jednotlivé závody dokážou vyrobit, je omezený.

5.6.1 Vytváření stromu komponent

Jak už bylo zmíněno, do 3D náhledu jsou vkládány celé sestavy jako jeden model. Seznam komponent bylo tedy nutné uložit do separátního pole v databázi Strapi. V tomto poli je celý vkládaný model reprezentován jako strom navazujících komponent. Společně s přibývajícími součástmi se rozrůstá i strom komponent, který se rovněž ukládá do reprezentativního objektu. Přístup se stromem součástí v budoucnu umožní vyhodnocovat optimální sourcing komponent z pohledu celých navazujících celků. V tuto chvíli je ale celá rozvodna naceněna z pohledu jednotlivých komponent.

Protože byly modely sestav tvořeny v programu Blender, bylo možné vyexportovat strom součástí přímo z Blenderu pomocí rekurzivního algoritmu napsaného v jazyce Python 3:

```
# Rekurzivni algoritmus pro vytvoreni stromu
# navazujicich soucasti v programu Blender

import bpy
import json

root_ = bpy.data.objects['root-node']

root_dict = {
    "name": root_.name,
    "meta": {},
    "children": []
}

def findChildren(parent, parent_dict):
    if len(parent.children) > 0:
        for child in parent.children:
            child_dict = {
                "name": child.name,
                "meta": {},
                "children": []
            }
            parent_dict["children"].append(child_dict)
            findChildren(child, child_dict)

```

```
findChildren(root_, root_dict)

with open("component-tree.json","w") as f:
    json.dump(root_dict,f)
```

5.6.2 API pro nacenění rozvodny

Logika nacenění rozvodny není součástí webové aplikace. Nacenění probíhá na serveru, pro který byl napsán skript v jazyce Python 3 s využitím knihoven NumPy, Pandas a FastAPI.

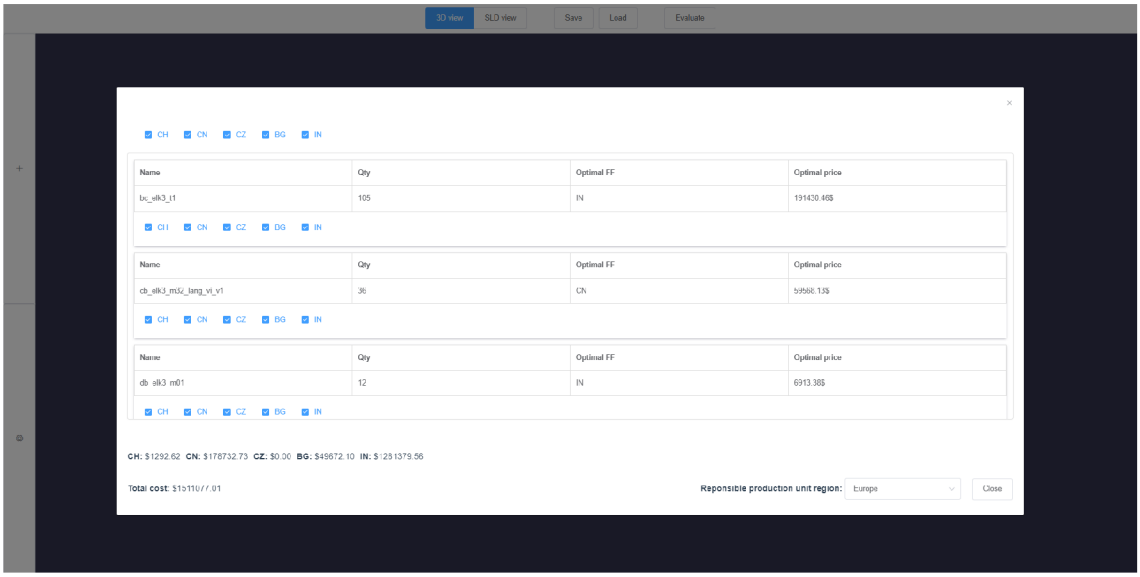
Po otevření okna určeného pro nacenění komponent je pomocí dotazovací metody POST na server odeslán požadavek, jehož součástí je stejný JSON objekt, jaký je generován při ukládání projektu. Součástí tohoto objektu je také strom navazujících komponent jednotlivých modulů. Z těchto stromů je poté vytvořen seznam komponent. Ceník komponent pro jednotlivé závody je uložen na serveru ve formátu *csv*. U komponent, které daný výrobní závod nedokáže vyrobit jsou nastaveny hodnoty těchto komponent na nekonečno. Seznam komponent je ohodnocen podle tohoto ceníku a poté vynásoben koeficientem, který reflektuje cenu za přepravu komponent z výrobního závodu do závodu, kde probíhá kompletace. Vynásobením jednotkové ceny počtem komponent je získána výsledná cena.

Klientovi je poté odeslán zpět ohodnocený seznam komponent ve formátu JSON. Tento ohodnocený seznam obsahuje ceny jednotlivých komponent pro každý výrobní závod. Výběr optimálních (nejlevnějších) výrobních závodů pak už probíhá u klienta, a to podle zadaných omezení.

5.6.3 Uživatelské rozhraní pro zjištění výsledné ceny

Okno s výslednou cenou uživatel vyvolá kliknutím na tlačítko *Evaluate* v horní liště aplikace. V tomto okně se zobrazí vygenerovaný seznam všech komponent, ze kterých se skládá rozvodna. Každá komponenta je v tomto seznamu reprezentována tabulkou, která obsahuje jméno komponenty, celkový počet komponent, celkovou cenu komponenty a zkratku nejlevnějšího dodavatelského závodu. Zákazník může během poptávky vyloučit některé dodavatelské závody a to jak pro konkrétní komponenty, tak pro celý projekt. Z tohoto důvodu je u každé komponenty pětice zaškrtávacích polí se zkratkou země výrobního závodu. Ve výchozím stavu jsou zahrnuty všechny výrobní závody. Případné změny ve vybraných výrobních závodech se projeví ihned automaticky.

Dále okno obsahuje náhled na rozložení cen mezi jednotlivými výrobními závody a je zde zobrazena také celková cena. Pomocí rozbalovací nabídky je možné vybrat také region zodpovědné výrobní jednotky (jednotka ve které probíhá kompletace). Ten by měl být zvolen s ohledem na region, ve kterém má být rozvodna postavena.



Obr. 11: Ukázka rozhraní pro nacenění výsledného produktu.

6 Závěr

S využitím frameworků Vue.js a Babylon.js bylo možné vytvořit webovou aplikaci konfigurátoru rozvodny velmi vysokého napětí, která kombinuje 3D vizualizaci s náhledem na jednopólové schéma výsledného produktu. Aplikace na straně klienta byla napsána v jazyce JavaScript. Serverová část využívá rozhraní Docker, ve kterém jsou spuštěny služby Strapi a FastAPI. Služba Strapi slouží jako databáze pro uložení 3D modelů komponent a dat s němi spojenými. FastAPI zajišťuje běh skriptu napsaného v jazyce Python 3, který pomocí rekurzivního algoritmu vytváří seznam součástí a zároveň zajišťuje jejich nacenění. Aplikace na straně klienta poté zobrazuje celkovou cenu rozvodny a rozdělení této ceny mezi výrobní závody, a to na základě uživatelem definovaných omezení.

Při vývoji konfigurátoru bylo na zřeteli budoucí rozšíření konfigurátoru o další produktové řady. Snadná implementace dalších produktů je zajištěna pomocí selekce imporotvaných modelů na základě průniku množin. Tento systém klade vyšší nároky na správnou kategorizaci komponent v databázi a proto jej musí spravovat osoba seznámena s vnitřní logikou aplikace.

Aplikace i se serverovou částí je díky využití standardů HTML5 a kontejnerizaci služeb nezávislá na platformě operačního systému. Podmínkou této nezávislosti je v případě klientské části využití moderního prohlížeče a v případě serverové části možnost spuštění služby Docker. Všechny frameworky a služby použity pro vývoj aplikace je možné využívat bezplatně, a to i v rámci korporátního prostředí, ve kterém aplikace vznikala.

7 Seznam použité literatury

[1] NECHMI, Houssem Eddine, Abderrahmane BEROUAL, Alain GIRODET a Paul VINSON. Fluoronitriles/CO2 gas mixture as promising substitute to SF6 for insulation in high voltage applications. *IEEE* [online]. 2016, 23(5), 2587 - 2593 [cit. 2022-10-05]. ISSN 1558-4135. Dostupné z: https://doi.org/10.1109/TDEI.2016.7736816.

[2] IDesigner - Kreator Wnętrz Online, In: *Extradom* [online]. 2018 [cit. 2022-05-10], Dostupné z: https://www.idesigner.com.pl/.

[3] JUNG, Krzysztof. Building a 3D iDesigner with Vue.js: Designing a Reactive Entity System. In: *Monterail* [online]. 2018 [cit. 2022-05-10], Dostupné z: https://www.monterail.com/blog/3d-editor-vue.js-reactive-entity-system.

[4] Konfigurátor vozu, In: *ŠKODA AUTO a.s.* [online]. 2019 [cit. 2022-05-10], Dostupné z: https://cc.skoda-auto.com/cze/cs-CZ/.

[5] Brikl - MicroStore and eCommerce software, In: *Brikl BV* [online]. 2020 [cit. 2022-05-10], Dostupné z: https://www.brikl.com/embellish-and-visualize.

[6] 3D Product Configurator & Augmented Reality For Commerce, In: *Threekit* [online]. 2019 [cit. 2022-05-10], Dostupné z: https://www.threekit.com/solutions/3d-product-configurator.

[7] HTML & CSS - W3C. In: *W3C* [online]. 2016 [cit. 2022-05-10], Dostupné z: https://www.w3.org/standards/webdesign/htmlcss.

[8] WIRFS–BROCK, Allen, Brendan ERICH. JavaScript: the first 20 years. *Proceedings of the ACM on Programming Languages* [online]. 2020, 77(4), 1 – 189 [cit. 2022-10-05]. Dostupné z: https://doi.org/10.1145/3386327.

[9] Stack Overflow Developer Survey 2021 In: *Stack Overflow* [online]. 2021 [cit. 2022-05-10], Dostupné z: https://insights.stackoverflow.com/survey/2021/#most-popular-technologies-language.

[10] ECMAScript 2021 language specification In: *ECMA* [online]. 2021 [cit. 2022-05-10], Dostupné z: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/.

[11] Global Web Stats - December 2021 In: *W3Counter* [online]. 2021 [cit. 2022-05-10], Dostupné z: https://www.w3counter.com/globalstats.php?year=2021&month=12.

[12] Desktop Browser Market Share Worldwide In: *StatCounter* [online]. 2021 [cit. 2022-05-10], Dostupné z: https://gs.statcounter.com/browser-market-share/desktop/worldwide/2021.

[13] Most Visited Websites In: *Similarweb* [online]. 2022 [cit. 2022-05-10], Dostupné z: https://www.similarweb.com/top-websites/.

[14] SHAUMIK, Daityari. Angular vs React vs Vue: Which Framework to Choose. In: *Codeinwp* [online]. 2021 [cit. 2022-05-10], Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.

[15] AngularJS - Version Support Status In: *AngularJS* [online]. 2022 [cit. 2022-05-10], Dostupné z: https://code.angularjs.org/snapshot/docs/misc/version-support-status.

[16] NOWAK, Przemek. GitHub stars history and charts In: *PrzemekNowak* [online]. 2022 [cit. 2022-05-10], Dostupné z: https://stars.przemeknowak.com/.

[17] Adobe bidding Flash farewell in 2020 In: *Phys.org* [online]. 2017 [cit. 2022-05-10], Dostupné z: https://phys.org/news/2017-07-adobe-farewell.html.

[18] WebGL - Graphics Pipeline In: *Tutorialspoint* [online]. 2021 [cit. 2022-05-10], Dostupné z: https://www.tutorialspoint.com/webgl/webgl_graphics_pipeline.htm.

[19] CATUHE, David. Where it comes from... In: *Medium* [online]. 2019 [cit. 2022-05-10], Dostupné z: https://babylonjs.medium.com/where-it-comes-from-78004604f5d1.

[20] CATUHE, David. Three.js or Babylon.js? And why? In: *Reddit* [online]. 2019 [cit. 2022-05-10], Dostupné z: https://www.reddit.com/r/javascript/comments/a7zbfp/threejs_or_babylonjs_and_why/ec718g1/.

[21] JUKIĆ, Tonči. Draw calls in a nutshell. In: *Medium* [online]. 2015 [cit. 2022-05-10], Dostupné z: https://toncijukic.medium.com/draw-calls-in-a-nutshell-597330a85381.

8 Seznam obrázků a tabulek

Seznam obrázků

Seznam tabulek