Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Konfigurátor vektorových obrázků

2022

Bc. Pavel Přidal

Vedoucí práce:
RNDr. Martin Trnečka, Ph.D.

Studijní program: Aplikovaná informatika,
prezenční forma

Bibliografické údaje

Autor:	Bc. Pavel Přidal
Název práce:	Konfigurátor vektorových obrázků
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2022
Studijní program:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	RNDr. Martin Trnečka, Ph.D.
Počet stran:	37
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Bc. Pavel Přidal
Title:	Vector image configurator
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2022
Study program:	Applied Computer Science, full-time form
Supervisor:	RNDr. Martin Trnečka, Ph.D.
Page count:	37
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem diplomové práce, která je vypisována ve spolupráci s firmou ConfigAir, je navrhnout a implementovat webovou komponentu, realizující editaci SVG obrázků s podporou drag and drop. Důraz bude kladen na znovupoužitelnost, přizpůsobivost a rozšiřitelnost komponenty. Komponenta bude používána klienty s různými požadavky na funkcionalitu. Součástí diplomové práce bude i ukázka použití komponenty formou jednoduchého SVG editoru.

Synopsis

The goal of the thesis, which is written in cooperation with the ConfigAir company, is to design and implement a web component that implements the editing of SVG images with drag and drop support. Emphasis will be placed on reusability, adaptability and extensibility of the component. The component will be used by clients with different functionality requirements. Part of the diploma thesis will also be a demonstration of the use of the component in the form of a simple SVG editor.

Klíčová slova: SVG; editor; vektorový obrázek; komponenta

Keywords: SVG; editor; vector image; component

Děkuji RNDr. Martinu Trnečkovi, Ph.D. za vedení této diplomové práce a také své rodině a přátelům za podporu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Cílem práce bylo vytvořit webový konfigurátor vektorových obrázků pro firmu ConfigAir, pomocí kterého si klienti firmy budou moci vizuálně vytvořit vektorový obrázek ve formátu SVG. Takto vytvořený obrázek bude převeden do firemního vizualizačního jazyka, ze které lze na serveru vytvořit vizualizaci pro webové konfigurátory klientů.

V současné době pro dosažení stejného výsledku je potřeba někdo, kdo zná syntaxi zmíněného vizualizačního jazyka a ovládá jej na dostatečné úrovni, aby zvládl zapsat vygenerování potřebných objektů. Editorem vytvořeným pro tuto diplomovou práci se tento proces velmi zjednoduší – odpadne potřeba znát vizualizační jazyk a bude stačit mít webový prohlížeč, ve kterém si uživatel vizuálně naskládá SVG elementy na požadované souřadnice. Tyto elementy bude moci dodatečně upravit pomocí drag&drop funkcionality.

Podobné aplikace, které jsou volně přístupné nebo i ty, které jsou zpoplatněné, nejsou na zmíněné využití vhodné. Některé obsahují zbytečně moc funkcionalit, některé naopak málo, ale všechny mají společný problém – nelze v nich spolehlivě vytvořit SVG soubor v takovém formátu, který bude vždy kompatibilní se serverovým řešením převodu do vizualizačního jazyka. Také takové aplikace nelze dále rozšiřovat a nebo ohýbat přesně podle potřeby klientů, u kterých se můžou lišit jejich požadavky.

Aplikace k této diplomové práci byla vypracována od základu s tím, aby se daly jednoduše použít jednotlivé komponenty podle potřeby klienta a také jim měnit nastavení i za běhu aplikace. Díky tomu může jednomu klientovi nabídnout základní verzi editoru a pro druhého vytvořit řešení na míru a ani jeden nebude mít ve výsledném balíčku nic navíc.

Práce obsahuje popis použitých technologií, programátorskou příručku a uživatelskou příručku.

2 Použité technologie

Praktickou částí této práce je aplikace vypracovaná v Reactu. Pro tuto technologii jsem se rozhodl z důvodu, že umožňuje jednoduchou práci se stavem a jeho úpravami i přesto, že volí funkcionální přístup. Také proto, že umožňuje zapisovat HTML kód pomocí JSX a jednoduše rozdělit logicky související části do samostatných komponent. Další a určitě ne poslední výhoda je zjednodušená práce s technologiemi Sass a TypeScript, které běžně potřebují nejdříve zkompilovat do CSS a JavaScriptu, ale při použití s Reactem je můžeme importovat a používat i nezkompilované.

2.1 React

React [1] je knihovna pro vytváření uživatelského rozhraní. Slouží k izolování složitého uživatelského rozhraní do malých, nezávislých a znovupoužitelných komponent. Jedná se o open-source knihovnu [2] udržovanou především společností Meta (dříve Facebook).

Základní stavební jednotka Reactu jsou komponenty [3]. Komponenty jsou funkce, které na vstupu přijímají objekt (**props** čili properties) a na výstupu vrací elementy, které se zobrazí uživateli. Speciální případ zmíněných props je atribut **children**, který nemusíme vkládat do klasických HTML atributů, ale můžeme ho vložit jako vnořený HTML (JSX) element.

Další důležitou funkcionalitou jsou stavy a schopnost Reactu vykreslit (renderovat) aplikaci při změně stavu a to pomocí speciálních funkcí, tzv. hooků [4] (např. **useState** nebo **useEffect**), které přibyly do Reactu s verzí 16.8 a umožnily psát kód více funkcionálně než předtím.

```
1  import React, { useState } from 'react';
2
3  function Example() {
4    const [count, setCount] = useState(0);
5
6    return (
7      <div>
8        <p>You clicked {count} times</p>
9        <button onClick={() => setCount(count + 1)}>
10         Click me
11       </button>
12     </div>
13   );
14 }
```

Zdrojový kód 1: React – příklad počítadla, po kliknutí na tlačítko dojde k inkrementaci hodnoty [4].

I přesto, že React zvolil funkcionální přístup místo původního objektově orientovaného, umožňuje uživateli volat funkce funkcionální komponenty podobně jako by to byla třída. To dělá pomocí **forwardRef**[5], což je funkce, která obalí komponentu, ve které je potom možné použít hook **useImperativeHandle** a v něm specifikovat veřejné funkce.

```
export const Zoomable = React.forwardRef(
  (props: Props, ref: React.ForwardedRef<ZoomableRef>) => {
   useImperativeHandle(ref, () => ({
     resetZoom() {...},
     zoomIn() {...},
     zoomOut() {...}
   })
   ...
  }
);
```

Zdrojový kód 2: React – ukázka forwardRef a useImperativeHandle

2.2 JSX a TSX

JSX [6] je rozšíření ECMAScriptu[1] o syntaxi podobnou XML (HTML). JSX soubory jsou za pomocí preprocesorů transformovány do standardního ECMAScriptu (JavaScriptu). Je často využíván v kombinací s knihovnou React [8], jelikož usnadňuje práci UI vývojářům.

JSX je ve své podstatě syntaktický cukr nad funkcí React.createElement() [9], užitečný obzvláště v případech, kdy jsou elementy do sebe zanořeny.

```
<MyButton color="blue" shadowSize={2}>
   Click me
</MyButton>

/*
se zkompiluje do:

React.createElement(
   MyButton,
   {color: "blue", shadowSize=2},
   'Click me'
);
*/
```

Zdrojový kód 3: JSX/TSX – příklad

[1]ECMAScript [7] je standard programovacího jazyka. Jeho implementací je např. jazyk JavaScript.

TSX [10] je rozšíření o definice typů používané v TypeScriptu. TSX se kompiluje do JSX.

2.3 TypeScript

TypeScript [11] je nádstavba (rozšíření) jazyka JavaScript[2] o typy:

- Základní – number, boolean, string, object, null, undefined, bigint, symbol
- Další – Array, Enum, any, never, void a uživatelsky definované typy (pomocí type alias a interface)

TypeScript se používá pro zlepšení statické analýzy kódu a díky tomu odhalení chyb dříve, než je program vůbec spuštěn. Kód se kompiluje do JavaScriptu.

```
1  function add (a: number, b: number): number{
2     return a + b;
3  }
4
5  /*
6  vygenerovaný JavaScript:
7  function add(a, b) {
8     return a + b;
9  }
10 */
```

Zdrojový kód 4: TypeScript – příklad funkce

Typy můžeme definovat sami nebo se odvodí (tzv. type inference [13]), takže je není potřeba explicitně vyjádřit. K takovému odvození dochází v různých situacích, například pokud proměnnou inicializujeme na hodnotu nějakého čísla nebo nastavujeme výchozí hodnotu parametru.

```
1  let counter = 0;
2
3  /*
4  je ekvivalentní k:
5
6  let counter: number = 0;
7  */
```

Zdrojový kód 5: TypeScript – Type inference

[2] JavaScript [12] – programovací jazyk sloužící především pro vývoj webových stránek, s možností reagovat pomocí event handlerů na události vyvolané uživatelem.

2.4 CSS a Sass

CSS [14] (Cascading Style Sheets) slouží k definici vzhledu webové stránky (od barev a rozmístění elementů až po složité animace a efekty).

Sass [15] je rozšíření jazyka CSS. Umožňuje psát přehlednější styly pomocí konstrukcí známých z jiných programovacích jazyků, např. vnořování selektorů, proměnné, funkce či mixiny. Vzniklý kód se poté zkompiluje do CSS.

```
1  @mixin theme(\$theme: DarkGray){
2     background-color: \$theme;
3        color: #fff;
4  }
5  .info{
6     @include theme;
7  }
8
9  .alert{
10    @include theme(\$theme: DarkRed);
11 }
12
13 .success{
14    @include theme(\$theme: DarkGreen);
15 }
16
17 /*
18 vygenerované CSS:
19 .info {
20    background-color: DarkGray;
21    color: #fff;
22 }
23
24 .alert {
25    background-color: DarkRed;
26    color: #fff;
27 }
28
29 .success {
30    background-color: DarkGreen;
31    color: #fff;
32 }
33 */
```

Zdrojový kód 6: Sass – ukázka

2.5 Nx

Poslední, ale neméně důležitou použitou technologií je build systém Nx[16], který je velmi vhodný pro použití v monorepo[3] projektech. Jeho filozofií je mít co nej-

[3] zjednodušeně řečeno – projekt, který obsahuje více vzájemně nezávislých podprojektů

jednodušší jádro, rozšiřitelné přes doplňky (pluginy). Takový projekt lze založit příkazem **npx create-nx-workspace@latest --preset=core**. Ten ale nabízí opravdu jen základní funkcionalitu a pro React je lepší využít příkazu **npx create-nx-workspace@latest**, který uživatele provede založením a vyzve ho k zadání názvu workspace, použité technologie, jména aplikace a výchozího formátu stylového souboru.

Tímto způsobem se dá jednoduše založit připravený workspace pro složitější monorepa s nachystanými soubory package.json (například pro použití s Reactem). Struktura vytvořené aplikace a její další použití je popsáno v kapitole Struktura.

Nx také urychluje vývoj skrz Visual Studio Code plugin [4]– Nx Console. V ní se dá mimo jiné vyvolat příkaz pro vytvoření základu nové React knihovny nebo komponenty.

Další výhodou jsou automatické inkrementální buildy jednotlivých knihoven a aplikací – to znamená, že lze mít spuštěnou aplikaci a provádět změny kdekoli v kódu, které se následně okamžitě projeví ve spuštěné aplikaci (pokud v upraveném kódu nebyly nalezeny syntaktické chyby). To je velká výhoda oproti běžnému monorepo projektu, který v základu tuto funkcionalitu nemá a je tedy potřeba při každé změně v jiné než spuštěné aplikaci provést nový build, aby se změny projevily. Díky tomu lze projekt rozdělit na jednotlivé ucelené části (knihovny) a jednoduše s nimi pracovat.

[4]existují pluginy i pro jiné populární editory, například pro Webstorm

3 Programátorská příručka

V této části následuje popis jednotlivých komponent a připravené demo aplikace, ve kterém lze pozorovat využití komponent a jejich spojení ve funkční celek.

3.1 Struktura

Struktura projektu se z velké části odvíjí od předvytvořeného prostředí pomocí Nx. Tato část práce obsahuje popis nejdůležitějších částí jako jsou složky **/libs** a **/apps**. Konfigurační a podobné soubory jako například project.json, tsconfig.json, které obsahuje každá knihovna a jsou zpravidla předem vytvořeny pomocí Nx, budou vynechány.

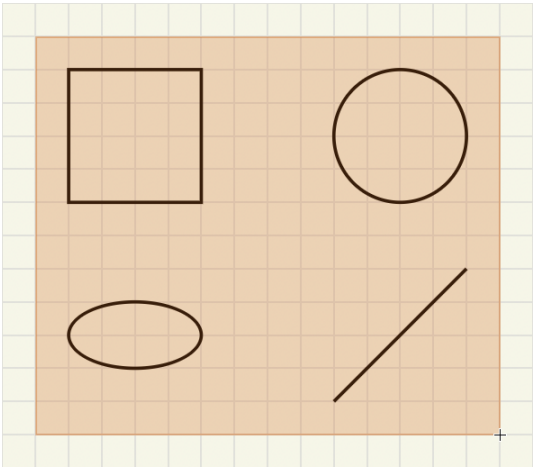
Každá knihovna obsahuje složku **/src**, v ní vnořenou složku **/lib** a soubor **index.ts**, který slouží pro export veřejných funkcionalit, čímž získáme určité zapouzdření v rámci TypeScriptu. Například můžeme exportovat komponentu bez jejich soukromých funkcí, které jsou určený pro použití pouze v této komponentě a nechceme je zpřístupnit zvenčí.

3.2 Vytvořené knihovny

Všechny knihovny jsou vytvořeny příkazem **nx generate @nrwl/react:library**.

3.2.1 Activable Svg

Komponenta ActivableSvg generuje HTML element <svg>, do kterého vkládá attributy, které má předané zvenčí, včetně children elementů. Dále tomuto <svg> elementu přidává **onMouseDown**, **onMouseMove** a **onMouseUp** event handlery. Ty se starají o to, aby se element stal při kliknutí aktivním a také umožňují vybrání více elementů zároveň pomocí kliknutí a tažení myši v editoru (obrázek: 1).



Obrázek 1: Vybrání více elementů zároveň

Knihovna přijímá atributy, které jsou spojením atributů elementu **<svg>** a dalších, které slouží pro funkcionalitu starající se o vybírání aktivních elementů. Ty mohou být dále použity komponentami implementující komponentu ActivableSvg.

```
type Props = React.SVGProps<SVGSVGElement> & {t
  activeElements: SVGGraphicsElement[];
  setActiveElements: React.Dispatch<React.SetStateAction<
      SVGGraphicsElement[]>>;
  tool: Tool;
  setTool: React.Dispatch<React.SetStateAction<Tool>>;
  getMousePoint?: GetMousePoint;
  zoom: number;
};
```

Zdrojový kód 7: Activable Svg – Props

Atributy **activeElements** a **setActiveElements** jsou aktuálně vybrané SVG elementy a funkce pro jejich nastavení, u obojího je očekávané, že komponenta využívající ActivableSvg tyto atributy vytvoří pomocí React hooku useState. To samé platí pro atributy **tool** a **setTool**, které se starají o předávání informací o tom, jaký nástroj je zrovna aktivní. Nástroje jsou více popsány v kapitole Enums.

Funkce **getMousePoint** je definovaná v knihovně Zoomable a je potřeba pro získání bodu SVG pod kurzorem. Atribut **zoom** je hodnota aktuálního přiblížení v editoru.

Komponenta také využívá SCSS pro pohodlnější vybírání elementů **<rect>**, **<circle>** a **<ellipse>** pomocí následujícího kódu:

```
.zoomable svg {
  & rect,
  & circle,
  & ellipse {
    pointer-events: bounding-box;
  }
}
```

Zdrojový kód 8: Activable Svg – elements pointer-events

Ukázka použití je v komponentě SvgEditor:

```
<ActivableSvg
  xmlns="http://www.w3.org/2000/svg"
  width={svgSize?.width}
  height={svgSize?.height}
  style={{
    backgroundImage: backgroundImage,
    backgroundColor: options.backgroundColor,
  }}
  activeElements={activeElements}
  ...
>
```

Zdrojový kód 9: Activable Svg – ukázka použití

3.2.2 Draggable

Knihovna draggable vytváří obal elementu předaného jako children a přidává mu funkcionalitu **drag**.

```
interface Props {
  dragImageStyle?: React.CSSProperties;
  dragImageRef?: React.RefObject<HTMLDivElement>;
  onDragStart?(event: React.DragEvent): void;
  onDrag?(event: React.DragEvent): void;
  children: React.ReactElement;
}
```

Zdrojový kód 10: Draggable – Props

Atribut **dragImageStyle** se stará o vzhled dragImage (obrázek, který se zobrazí pod kurzorem při táhnutí elementu). **DragImageRef** slouží k předání reference pro jednoduché ovládání nebo úpravy dragImage komponentami, které využívají Draggable komponenty.

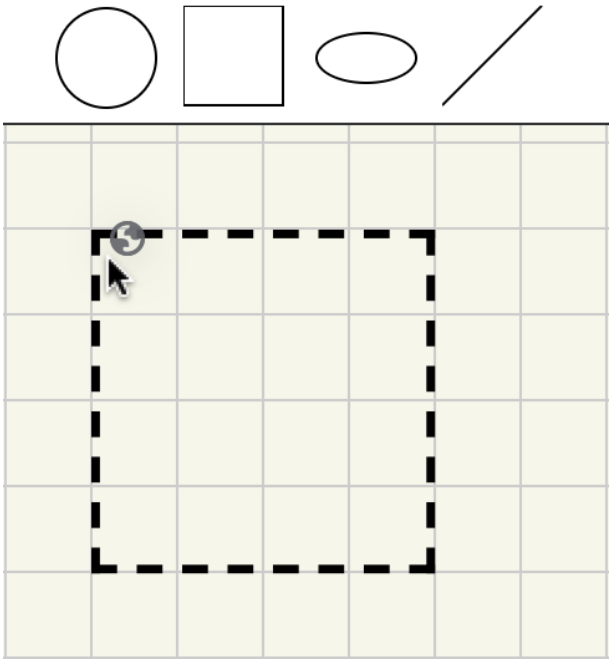
Event handlery **onDragStart** a **onDrag** umožňují zvenčí definovat co se má stát při začátku táhnutí a následném táhnutí elementu.

Ukázka použití je v demo aplikaci v souboru **/src/app/components/DraggableSvg**:

```
<Draggable
  dragImageStyle={dragImageStyle}
  dragImageRef={dragImageRef}
  onDragStart={onDragStart}
  onDrag={onDrag}
>
    {children}
</Draggable>
```

Zdrojový kód 11: Draggable – ukázka použití

Na následujícím obrázku 2 je vidět, že při přetáhnutí **DraggableSvg** elementu do editoru lze dragImage upravit (nejen) podle faktoru přiblížení – v tomto případě 350%. To lze jednoduše provést díky dříve zmíněným atributům komponenty Draggable.



Obrázek 2: Přetáhnutí elementu zvenčí do editoru

3.2.3 Droppable

Knihovna Droppable umožňuje svému child elementu naslouchat eventům **onDrop**, **onDragEnter** a **onDragOver**.

```
interface Props {
  droppableRef?: React.RefObject<HTMLDivElement>;
  onDrop(event: React.DragEvent): void;
  onDragEnter?(event: React.DragEvent): void;
  onDragOver?(event: React.DragEvent): void;
  children: React.ReactElement;
  style?: React.CSSProperties;
}
```

Zdrojový kód 12: Droppable – Props

Funkce vyvolané těmito eventy si specifikuje rodičovská komponenta. Droppable dále přijímá atributy droppableRef a style pro jednoduché získání reference a úpravy vzhledu zvenčí. Využití lze najít v komponentě SvgEditor, kde komponenta Droppable tvoří celkový obal všech dalších komponent a HTML elementů. Pro zjednodušení jsou v příkladu vynechány předávané atributy:

```
<Droppable>
  <Zoomable>
    <ActivableSvg>
      ...
    </ActivableSvg>
  </Zoomable>
</Droppable>
```

Zdrojový kód 13: Droppable – ukázka použití

3.2.4 Element Menu

Knihovna element-menu se skládá ze dvou komponent – **ElementMenu** a **Attributes**.

ElementMenu má 3 atributy:

```
interface Props {
  elements: SVGGraphicsElement[];
  updatedFromOutside: number;
  setUpdated: React.Dispatch<React.SetStateAction<number>>;
}
```

Zdrojový kód 14: Droppable – Props

Z předaných **elements** vytvoří pole objektů obsahující jméno atributu a jeho výchozí hodnotu pomocí rozřazení v souboru **element-attributes.ts**. Zmíněné pole je následně použito v komponentě **Attributes**. **UpdatedFromOutside**

a **setUpdated** se starají o správnou synchronizaci mezi hodnotami atributů elementu a hodnotami zobrazenými v menu.

Komponenta **Attributes** podle předaných atributů vygeneruje seznam atributů, kde jsou rozdělené podle typu – číselné, procentuální nebo barvy. To je důležité pro správné nastavování a zobrazování hodnot, kdy v případě číselného atributu je vygenerován číselný HTML element <input>, v případě procentuálního přidává suffix, aby bylo uživateli jasné, že se jedná o procenta. V případě barev pak vygeneruje input s typem *color*, u kterého se prohlížeč postará o možnost vybrání barevné hodnoty.

3.2.5 Enums

V knihovně enums jsou definované výčtové typy používané více knihovnami, za zmínku stojí především výčet **Tool**, který definuje jednotlivé nástroje, které se nastavují v editoru například při drag&drop funkcionalitě nebo při změně velikosti elementu pomocí kurzoru. Podle nastaveného nástroje se mění i vzhled kurzoru.

```
1  export enum Tool {
2    NONE = 'NONE',
3    PAN = 'MOVE',
4    ZOOM = 'ZOOM',
5    NW_RESIZE = 'NW-RESIZE',
6    SW_RESIZE = 'SW-RESIZE',
7    ...
8    PATH_MOVE_POINT = 'PATH_MOVE_POINT',
9    SELECTING_ELEMENTS = 'SELECTING_ELEMENTS',
10 }
```

Zdrojový kód 15: Enums – Tool

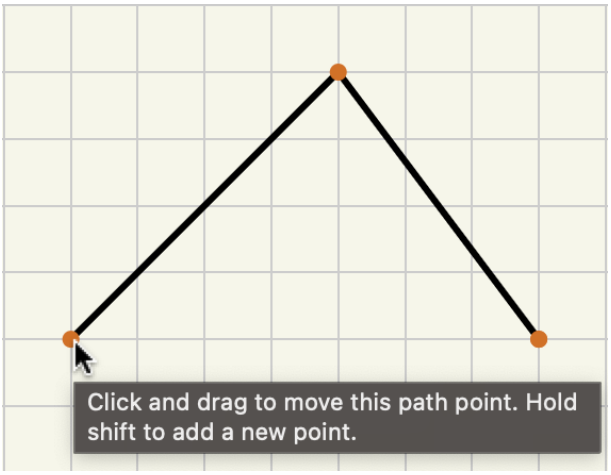
Také obsahuje výčet **SvgAttribute**, ve kterém jsou definované nastavitelné atributy SVG elementů.

3.2.6 SVG Editor

Knihovna svg-editor se stará o spojení logiky mezi jednotlivými komponentami jako jsou **Droppable**, **Zoomable**, **ActivableSvg** a přidává k tomu svoje vlastní **GuideLines** . Tím umožňuje vytvořit editor, který reaguje na event onDrop, má funkci pan a zoom a také v něm lze zvolit aktivní elementy (a díky tomu například měnit jejich atributy). Editor také přidává elementům možnost lepit se (snap on) na nastavenou mřížku (grid) editoru, případně na elementy na stejných souřadnicích x nebo y. Tím umožňuje centrovat nově přidávané a nebo přesouvané elementy vzhledem k ostatním, které se v editoru již nachází. Elementy, na které je SVG Editor připraven jsou SVG elementy <rect>, <circle>, <ellipse> a <path>.

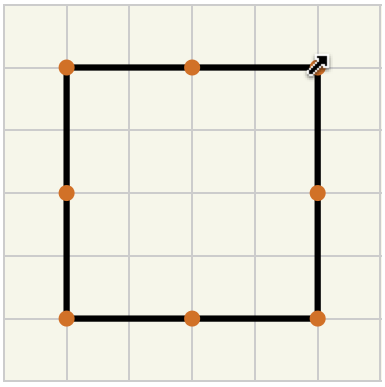
Komponenta **ActiveElements** přidává aktivním elementům získaných z **ActivableSvg** možnost měnit jejich pozici v editoru. To umožňuje přidáním příslušných event handlerů pro HTML document a v nich vypočítáním nové pozice vybraného elementu podle rozdílu souřadnic při tažení kurzoru. Tato funkcionalita funguje i v případě více vybraných elementů najednou, kdy přesune všechny zároveň. Také přidává funkci smazání elementu (elementů) pomocí klávesy **delete** na klávesnici. Nakonec jsou v ní použity komponenty **PathControls** a **ActiveElementResize**.

PathControls je komponenta, která přidává vybranému SVG elementu <path> funkce umožňující přesun jednotlivých bodů nebo přidání dalšího. Přesunout lze jakýkoli bod vybraného <path> elementu pomocí kliknutí a tažení kurzoru. Přidávat lze pouze z prvního a posledního stisknutím a držením klávesy **shift** a následným kliknutím a tažením kurzoru. Na to také upozorňuje tooltip, který se zobrazí při najetí na příslušný bod (obrázek: 3). Element <path> lze také přesouvat celý najednou při drag&drop mimo zobrazené body.



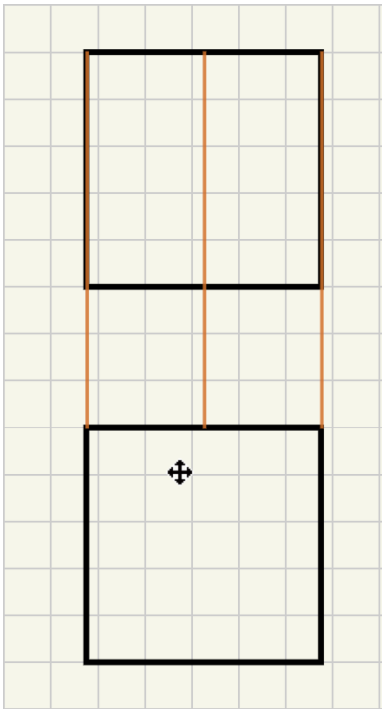
Obrázek 3: Tooltip na krajním bodu elementu <path>

ActiveElementResize pracuje s elementy, které nejsou <path> a přidává jim možnost změny velikosti pomocí drag&drop. Tato funkcionalita je umožněna pomocí kliknutí na příslušný roh elementu a následné táhnutí kurzoru na požadované místo (obrázek: 4). Změna velikosti je řízená podle příslušného atributu zvoleného elementu, to znamená, že pro element <rect> lze měnit jeho atributy width a height, pro <ellipse> je to rx a ry, obojí podle toho, který roh je použit pro změnu příslušné velikosti. U kruhu lze tímto způsobem měnit pouze atribut r, to znamená jeho poloměr (radius).



Obrázek 4: Změna velikosti elementu pomoci kurzoru

GuideLines jsou pomocné čáry, které slouží k usnadnění zarovnání elementu vzhledem k ostatním elementům v editoru. Zobrazí se při přesouvání existujícího elementu a nebo při přidávání nového (obrázek: 5). Čáry značí společné souřadnice s jiným elementem na začátku, prostředku a konci v obou osách (X a Y). To společně s dříve popsanou funkci **snap** umožňuje napozicovat přidávané nebo upravované elementy podle těch už existujících. Tohle funguje, i když se změní velikost mřížky (gridu), v tom případě mají přednost již existující elementy před gridem, jak je vidět na následujícím obrázku – přemisťovaný element má tendenci se lepit na souřadnice již přítomného elementu před gridem (na který lze element také přesunout při dostatečné preciznosti u přesouvání).



Obrázek 5: Guide lines při přesouvání elementu

3.2.7 Types

Knihovna types definuje TypeScriptové typy pro nastavitelné možnosti knihoven **svg-editor** a **zoomable** a také definici pro **ref**, pomocí kterého lze v komponentě vyvolat nějakou funkci.

```
export interface SvgEditorOptions {
  backgroundColor?: string;
  grid?: {
    gap?: number;
    style?: GridLinesStyle;
    color?: string;
    snap?: boolean;
    hide?: boolean;
  };
  elements?: {
    snap?: boolean;
    snapRadius?: number;
  };
  visible?: boolean;
  zoomOptions?: ZoomOptions;
}
```

Zdrojový kód 16: Types – SvgEditorOptions

```
export interface ZoomOptions {
  allowZoom?: boolean;
  allowPan?: boolean;
  step?: number;
  minZoom?: number;
  maxZoom?: number;
  onZoomChange?: OnZoomChange;
  onPanChange?: OnPanChange;
  onToolChange?: OnToolChange;
}
```

Zdrojový kód 17: Types – ZoomOptions

Atribut **step** ovládá faktor přiblížení, o který je element transformován při eventu onWheel nebo při kliknutí na tlačítko, které má funkci přiblížení nebo oddálení. **OnZoomChange**, **onPanChange** a **onToolChange** je možné použít pro získání informace o změně vnějšími komponentami.

Tyto možnosti jsou následně sloučené s výchozím nastavením v příslušných knihovnách.

```
export interface SvgEditorRef {
  zoomableRef: ZoomableRef | null;
  createNewEditor: (width: number, height: number) => void;
  changeEditorSize: (width: number, height: number) => void;
  import: (file?: File) => void;
  export: (extension: 'svg' | 'png') => void;
}
```

Zdrojový kód 18: Types – SvgEditorRef

```
export interface ZoomableRef {
  getChild(): HTMLElement | undefined;
  getMousePoint: GetMousePoint;
  center(): void;
  translate(x: number, y: number): void;
  resetZoom(): void;
  resetView(): void;
  zoomIn(): void;
  zoomOut(): void;
  zoomTo(scale: number): void;
}
```

Zdrojový kód 19: Types – ZoomableRef

3.2.8 Utils

V knihovně utils nalezneme definované funkce používané napříč knihovnami, za zmínku stojí především funkce **strokeWidthByZoom**, která vypočítá tloušťku elementu podle faktoru přiblížení. Používaná je například pro změnu velikosti tloušťky gridu, který by bez úpravy byl vizuálně moc tlustý při větším přiblížení v editoru.

3.2.9 Zoomable

Zoomable je knihovna, která vytvoří obal v podobě HTML elementu a umožní funkci přiblížení nebo oddálení (zoom) a přemístění (pan) svému child elementu. To dělá pomocí event handlerů **onMouseDown**, **onMouseMove**, **onMouseUp** a **onWheel** na zmíněném obalu. Zvenčí je ovladatelná pomocí **options** a **ref** popsaných v kapitole Types.

Zmíněné funkcionality jsou prováděny pomocí CSS atributu **transform** a vypočítáním příslušné maticové transformace. U funkcionality pan také nastavuje nástroj Tool.PAN, u kterého lze odchytit změnu zvenčí a tím získat informaci o změně.

4 Vytvořená demo aplikace

Demo aplikace ve složce **/apps/demo/** ukazuje reálné využití v praxi, kdy editor podobný této aplikaci bude k dispozici klientům firmy ConfigAir a ti v něm budou mít možnost vytvořit SVG obrázek, který bude následně převeden do vizualizačního jazyka, ze kterého se generují vizualizace. Díky tomu pro ně odpadne potřeba znát přímo tento jazyk a orientovat se v jeho syntaxi. Bude jim stačit pouze umět ovládat aplikaci, která je bude schopná vizuálně navést za požadovaným výsledkem.

4.1 App

Komponenta **App** především využívá komponenty **SvgEditor**, ze které získává informaci o vybraných (aktivních) elementech, také jí předává proměnnou **editorOptions**, kde je nastavená barva pozadí editoru, velikost mřížky a její barva. V případě, že není zvolen žádný element, lze tyto možnosti nastavit z grafického rozhraní editoru, jak bude ukázáno v kapitole Možnosti editoru.

Pro změnu atributů zvolených aktivních elementů využívá dříve popsané ElementMenu.

Pro ostatní funkcionality jsou vytvořeny další komponenty **Modal**, **ModalContact**, **ModalNew**, **Draggables**, **DraggableSvg**, **EditorOptions**, **Loading** a **MenuList**.

4.1.1 Modal

Modal je samostatná komponenta, která představuje modální okno.

```
1 interface Props {
2   name: string;
3   title?: string;
4   closeModal: () => void;
5   children: React.ReactElement;
6 }
```

Zdrojový kód 20: Modal – Props

Při otevření celé okno prohlížeče překryje, aby se nedal ovládat obsah pod modálním oknem, když je otevřené. Z předaných atributů vykreslí **title** do své hlavičky společně se zavíracím tlačítkem. Zavření okna lze vyvolat i zmáčknutím tlačítka **escape** na klávesnici. Pod hlavičku vykreslí předané **children** elementy. Atribut **name** je použit jako jméno HTML třídy elementu pro jednodušší selektor v CSS souboru. Na obrázku 6 můžeme vidět modální okno bez dodatečných úprav ve stylech.

Obrázek 6: Modal

Komponenta Modal je využita pro vytvoření komponent ModalContact a ModalNew, které slouží pro zobrazení kontaktních informací a pro vytvoření nového SVG v editoru s možností specifikovat jeho šířku a výšku.

4.1.2 Draggables

Draggables využívá komponenty Draggable pro vytvoření své komponenty **DraggableSvg**. Ta se stará o změnu dragImage tak, aby správně zobrazoval velikost přidávaného elementu. Dále je využita k vygenerování elementů circle, rect, ellipse a path do HTML, na ty uživatel může kliknout a přetáhnout je do editoru, do kterého se při puštění kurzoru přidají.

Vizuální stránka bude ukázána v kapitole Přidání elementu.

4.1.3 EditorOptions

Komponenta EditorOptions je zobrazena v případě, že nejsou vybrány žádné aktivní elementy a dává uživateli možnost nastavit velikost mřížky a jestli ji chce zobrazit, také její barvu, barvu pozadí editoru a jeho průhlednost. Pro všechny zmíněné možnosti je vytvořen HTML element <input> s příslušným event handlerem a typy.

Vizuální stránka bude ukázána v kapitole Možnosti editoru.

4.1.4 Loading

Loading je velice jednoduchá komponenta, kterou tvoří jen načítací kolečko a je zobrazena pro zamezení problikávání grafických elementů při načítání editoru.

4.1.5 MenuList

MenuList tvoří lištu ovládacích prvků, ta obsahuje kategorie, které po kliknutí zobrazí seznam ovládacích tlačítek, kterým lze přiřadit různé funkcionality. Me-

nuList si pamatuje index otevřené kategorie. Pokud je již nějaká otevřená a kurzorem přejedeme na jinou, zobrazí se místo té původní.

Každá z těchto kategorií je upravitelná a lze do ní jednoduše přidat další ovládací prvky, případně i celou novou kategorii. Stačí pouze přidat další prvek seznamu podobný tomuto:

```
<li className="editor__menu-list-item">
  <Menu title="Help" index={3} {...commonProps}>
    <ul>
      <li onClick={() => setShowModalContact(true)}>Contact</li>
    </ul>
  </Menu>
</li>
```

Zdrojový kód 21: MenuList – přidání další kategorie

Atribut index slouží pro správné fungování zmíněné funkcionality při přejetí kurzorem. CommonProps jsou pak atributy, které jsou společné pro všechny komponenty Menu v MenuListu. Vnořený element <ul> tvoří seznam jednotlivých ovládacích tlačítek, kterým je přiřazená nějaká funkce, v ukázaném příkladu je to zobrazení modálního okna ModalContact.

V případě, že chceme vedle ovládacího tlačítka zobrazit také klávesovou zkratku a nebo obecně způsob, jak vyvolat jeho akci, stačí prvek seznamu vytvořit následujícím způsobem:

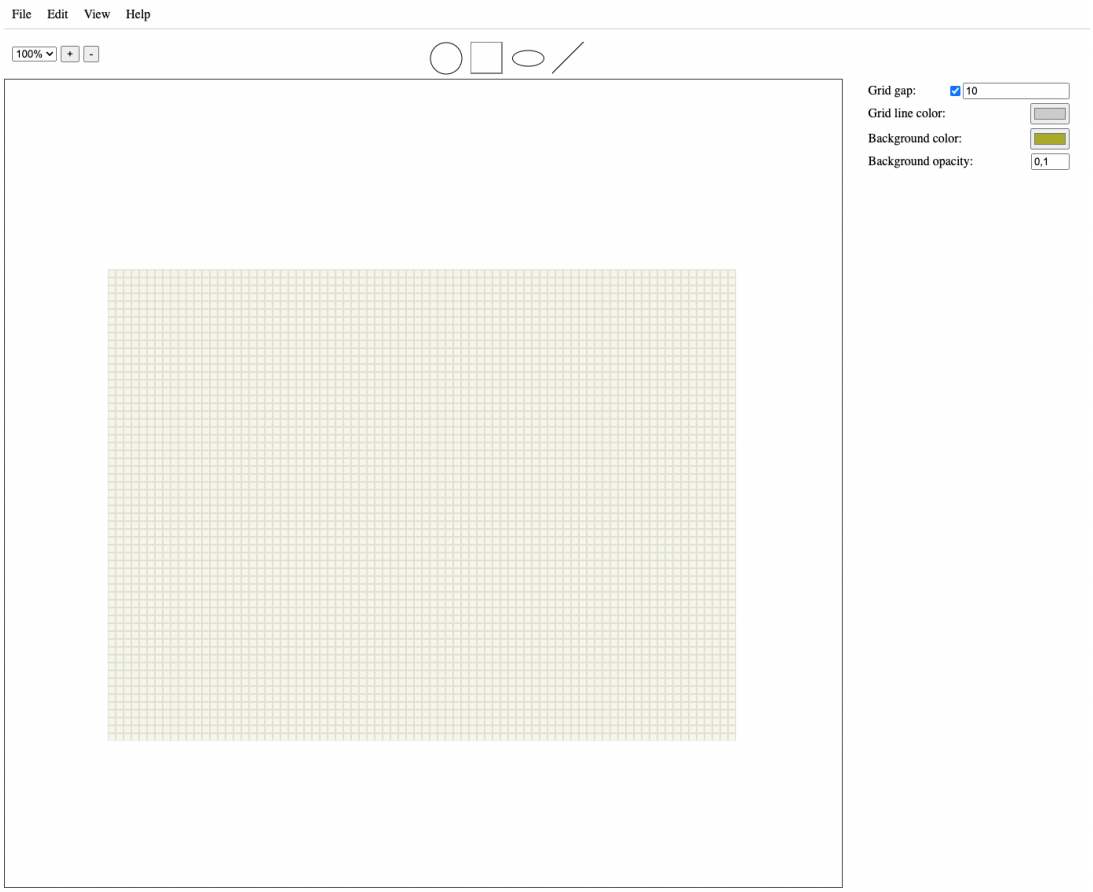
```
<li onClick={() => svgEditorRef.current?.selectAllElements()}>
  <span>Select All</span>
  <span>CTRL+A</span>
</li>
```

Zdrojový kód 22: MenuList – klávesová zkratka

Vizuální stránka bude ukázána v kapitole Menu.

5 Uživatelská příručka

Při načtení aplikace můžeme vidět editor (obrázek 7), do kterého lze vkládat SVG elementy a různě je upravovat. Nabízí také možnosti importu, exportu a různé ovládací prvky v menu liště v horní části aplikace.



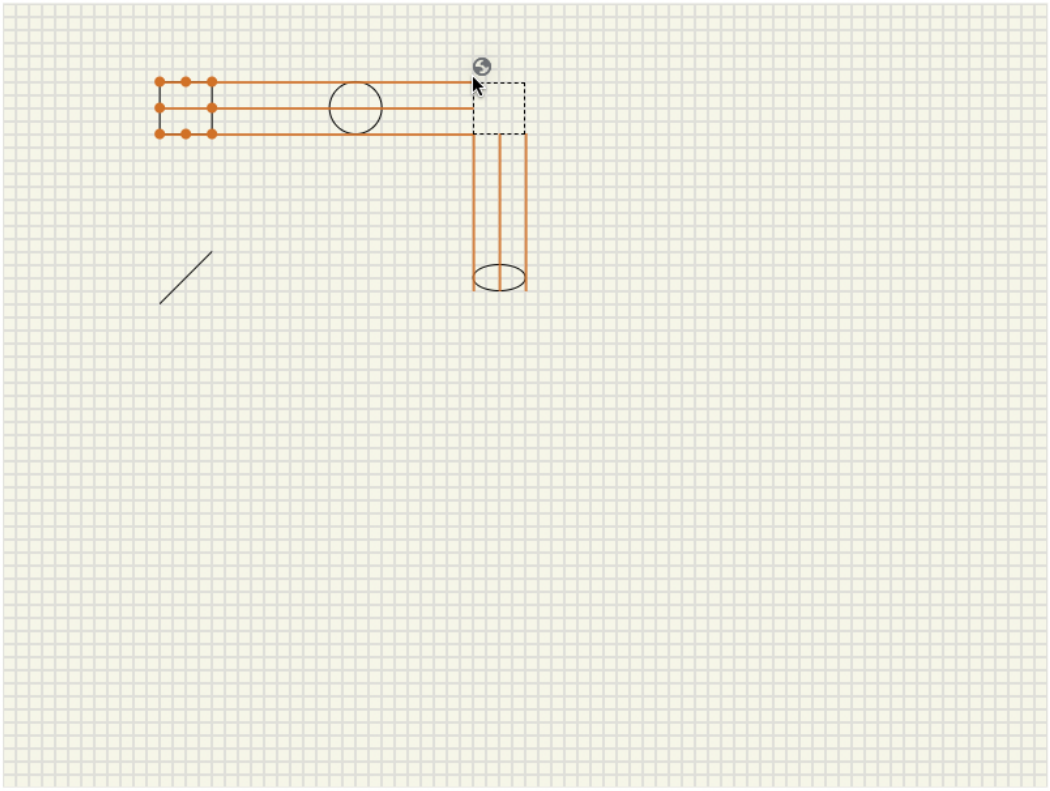
Obrázek 7: Aplikace

5.1 Editor

Nejdůležitější částí aplikace je editor SVG elementů (obrázek: 8). V něm se uživatel může pohybovat pomocí držení klávesy **control** + kliknutí a tažení kurzoru. Pomocí kolečka myši a nebo scroll funkci na touchpadu se lze v editoru přibližovat nebo oddalovat na místo, kde se právě nachází kurzor.

Uživatel do něj také může vkládat nové elementy nebo upravovat (přesouvat, měnit velikost nebo atributy) již vložené pomocí tažení. S obojím mu vizuálně pomáhá mřížka editoru, na kterou se elementy při přidávání lepí a také vodicí

čáry, které zobrazují, jestli (a kde) je tažený element zarovnaný s jinými. Těmto elementům můžeme měnit jejich velikost taháním za hrany nebo rohy.



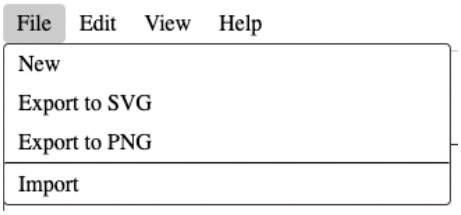
Obrázek 8: Editor

5.2 Menu

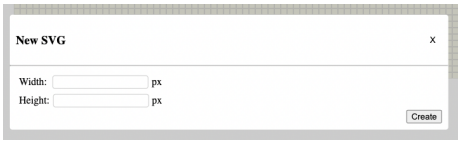
Menu obsahuje 4 kategorie – File, Edit, View a Help.

5.2.1 File

V kategorii File (obrázek: 9) můžeme vytvořit nové SVG se specifikovanými rozměry (obrázek: 10), po kliknutí bude tímto SVG nahrazeno současné v editoru.



Obrázek 9: Menu – File

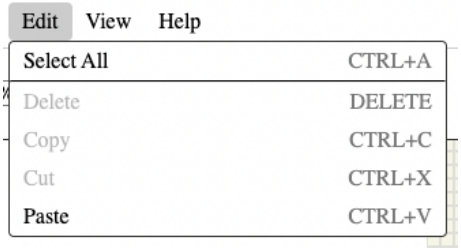


Obrázek 10: Modal – New

Dále lze v této kategorii najít funkce pro export do formátu SVG nebo PNG a také importovat SVG obrázek. V případě importování obrázku vytvořeného mimo editor se takový obrázek přetransformuje do elementu <image> a ten je následně vložen do editoru. Pokud vložíme SVG, které bylo vytvořeno v editoru, je do něj přímo vloženo a můžeme upravovat jeho jednotlivé elementy.

5.2.2 Edit

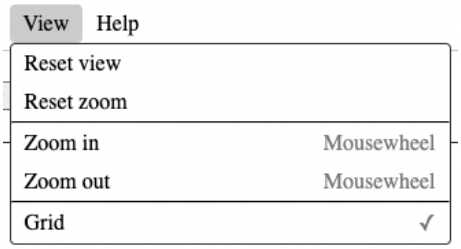
Kategorie Edit obsahuje ovládací prvky pro (aktivní) elementy. Máme zde funkce pro výběr všech elementů v editoru, smazání, kopírování nebo vyjmutí vybraných do schránky a také vložení elementů do editoru ze schránky. Pokud není vybrán žádný element, funkce smazání, kopírování a vyjmutí jsou vypnuty, jak lze vidět na obrázku 11. Ke každé funkci je také nápověda v podobě klávesové zkratky, pomocí které lze příslušnou funkci vyvolat i mimo menu.



Obrázek 11: Menu – Edit

5.2.3 View

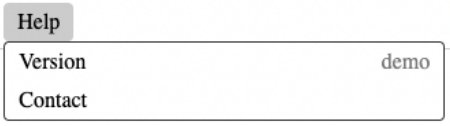
V kategorii View (obrázek: 12) jsou funkce pro obnovení celkového pohledu (to je obnovení faktoru přiblížení i posunu v editoru) nebo pouze faktoru přiblížení. Také zde lze měnit faktor přiblížení po skocích, které jsou ve výchozím nastavení 10% z aktuálního přiblížení. Poslední funkcí je vypnutí nebo zapnutí mřížky v editoru.



Obrázek 12: Menu – View

5.2.4 Help

Kategorie Help (obrázek: 13) obsahuje informace o verzi editoru a také možnost zobrazit si kontaktní informace ve vyskakujícím modálním okně (obrázek: 14).



Obrázek 13: Menu – Help

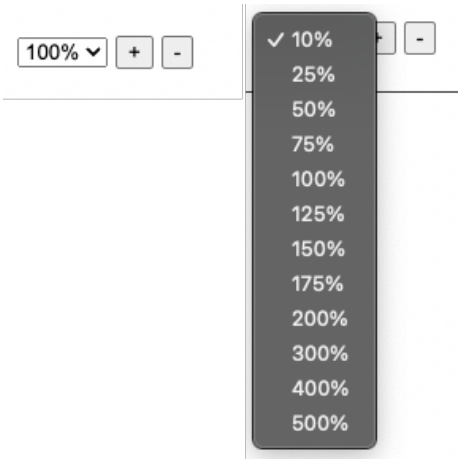
Obrázek 14: Menu – Help – modální okno Contact

5.3 Nástrojová lišta

V nástrojové liště, která se nachází nad editorem, nalezneme 2 funkcionality – změnu přiblížení a přidání elementu pomocí drag&drop.

5.3.1 Změna přiblížení

Faktor přiblížení editoru můžeme nastavit pomocí kliknutí na element ukazující aktuální faktor přiblížení a následné vybrání požadované hodnoty (obrázek: 15).



Obrázek 15: Nástrojová lišta – změna přiblížení

5.3.2 Přidání elementu

Na každý ze 4 nachystaných elementů (obrázek: 16) lze kliknout a přetáhnout ho do editoru. Při této akci je tažený element (vizuálně) transformován podle faktoru přiblížení v editoru.

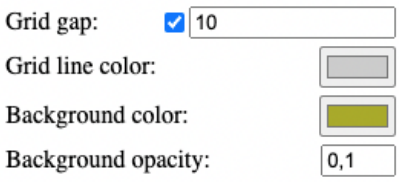
Obrázek 16: Nástrojová lišta – přidání elementu

5.4 Úprava editoru a elementů

V pravé části aplikace je možné upravovat možnosti editoru nebo elementů v případě, že jsou nějaké vybrány.

5.4.1 Možnosti editoru

Možnosti editoru (obrázek: 17) umožňují číselně nastavit velikost mřížky nebo ji vypnout, také pomocí výběru nastavit buď barvu mřížky a nebo barvu pozadí editoru a také její průhlednost. Všechny změny v těchto možnostech se projeví v reálném čase.



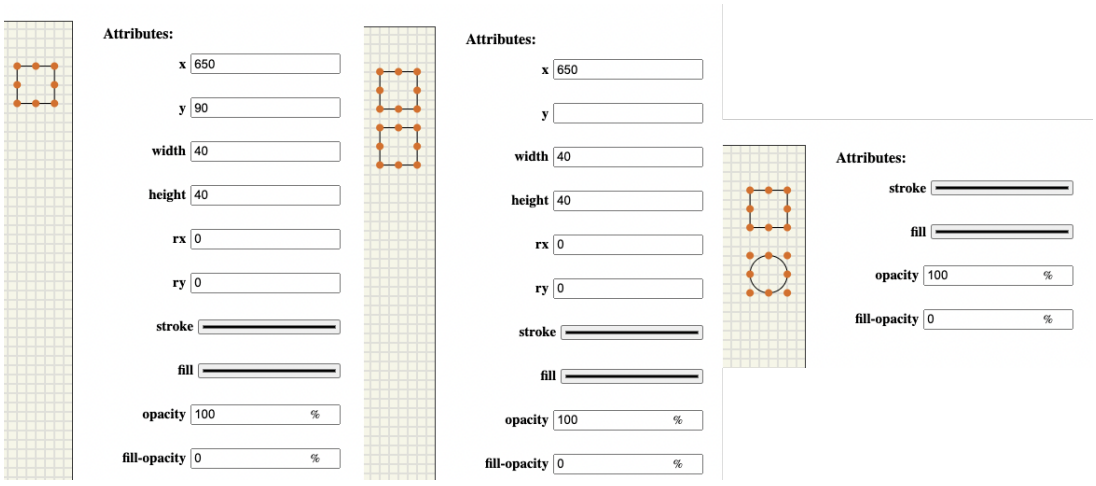
Obrázek 17: Možnosti editoru

5.4.2 Možnosti aktivního elementu

Při vybrání jednoho elementu je zobrazen seznam atributů a jejich hodnot daného elementu (obrázek: 18 – vlevo). Možnosti vložení hodnoty se liší podle typu hodnoty.

Pokud uživatel vybere více elementů stejného typu (obrázek: 18 – uprostřed), je zobrazený seznam stejný jako v prvním případě, jen s rozdílem v zobrazení hodnot – pokud je hodnota stejná pro všechny vybrané elementy, je zobrazena, pokud je různá, je zobrazen prázdný input. V případě, že uživatel nějakou hodnotu změní, je nastavena pro všechny vybrané elementy.

Když uživatel vybere více elementů různého typu (obrázek: 18 – vpravo), pak zobrazený seznam obsahuje průnik jejich atributů, to znamená pouze ty, které mají společné. V případě jejich změny opět platí to, že jsou nastaveny všem vybraným najednou.



Obrázek 18: Porovnání možností aktivních elementů

Závěr

Navrhl a implementoval jsem webový konfigurátor vektorových obrázků, který bude využit klienty firmy ConfigAir. Konfigurátor je upravitelný a rozšiřitelný podle požadavků klienta.

Pro účely práce byla také vytvořena demo aplikace, ve které si uživatel může vyzkoušet základní funkcionality konfigurátoru jako je přidávání a úprava různých SVG elementů (například změnou jejich atributů), pohyb v editoru, import a export obrázků do formátu SVG nebo PNG.

Conclusions

I designed and implemented a web-based vector image configurator to be used by ConfigAir clients. The configurator is editable and expandable according to the client's requirements.

For the purposes of the work, a demo application was created in which the user can test the basic functionality of the configurator, such as adding and editing various SVG elements (for example, by changing their attributes), moving in the editor, importing and exporting images to SVG or PNG format.

A Spuštění aplikace

Pro spuštění aplikace je potřeba mít nainstalovaný Node.js, následně v adresáři **/src** stačí spustit příkaz **npm i** a po nainstalování balíčků příkaz **npm start**. Tím se aplikace spustí na výchozí adrese localhost:4200.

B Obsah přiloženého datového média

`doc/`
: Text práce ve formátu PDF a zdrojové kódy LaTeX.

`src/`
: Zdrojové kódy konfigurátoru.

`readme.txt`
: Instrukce pro instalaci a spuštění programu.

Literatura

[1] *React docs*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://reactjs.org/⟩.

[2] Meta. *Meta Open Source – React* [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://opensource.fb.com/projects/react/⟩.

[3] *React docs – Components and Props*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://reactjs.org/docs/components-and-props.html⟩.

[4] *React docs – Hooks*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://reactjs.org/docs/hooks-overview.html⟩.

[5] *React docs – forwarding refs*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://reactjs.org/docs/forwarding-refs.html⟩.

[6] *JSX docs*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://facebook.github.io/jsx/⟩.

[7] *ECMAScript docs*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://tc39.es/ecma262/⟩.

[8] *React docs – introducing JSX*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://reactjs.org/docs/introducing-jsx.html⟩.

[9] *React docs – JSX in depth*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://reactjs.org/docs/jsx-in-depth.html⟩.

[10] *TypeScript docs – JSX*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://www.typescriptlang.org/docs/handbook/jsx.html⟩.

[11] *TypeScript docs*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://www.typescriptlang.org/⟩.

[12] Mozilla Foundation. *About JavaScript* [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript⟩.

[13] *TypeScript docs – type inference*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://www.typescriptlang.org/docs/handbook/type-inference.html⟩.

[14] Mozilla Foundation. *CSS: Cascading Style Sheets* [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://developer.mozilla.org/en-US/docs/Web/CSS⟩.

[15] *Sass docs*. [online]. [cit. 2022-7-20]. Dostupný z: ⟨https://sass-lang.com/documentation⟩.

[16] *Nx docs*. [online]. 2021 [cit. 2022-7-20]. Dostupný z: ⟨https://nx.dev/getting-started/intro⟩.

[17] Larsen, J. *React Hooks in Action: with Suspense and Concurrent Mode*. 2021.

[18] Flanagan, D. *JavaScript: The Definitive Guide, 7.th Edition*. 2020.

[19] Garreau, J.; Faurot, W. *Redux in Action*. 2018.