VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Editor TYP souborů pro nástroj Mkgmap

TYP File Editor for Mkgmap

Adam Draisaitl

Bakalářská práce

Vedoucí práce: Ing. Jan Janoušek

Ostrava, 2022

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Adam Draisaitl**

Studijní program: B0613A140014 Informatika

Téma: Editor TYP souborů pro nástroj Mkgmap
TYP File Editor for Mkgmap

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit webovou aplikaci pro tvorbu a editaci TYP souborů pro nástroj Mkgmap postavenou na frameworku Angular. Aplikace bude poskytovat grafické uživatelské rozhraní pro kompletní editaci veškerého obsahu TYP souboru. Jádrem aplikace bude jednoduchý grafický editor pro kresbu a editaci bitmap, které jsou v daném TYP souboru obsaženy.

Hlavní body zadání:
1. Seznámení s moderními technologiemi pro tvorbu webových aplikací a frameworkem Angular.
2. Návrh a implementace knihovny pro tvorbu a editaci TYP souborů.
3. Návrh a implementace webového editoru TYP souborů.
4. Uživatelské testování a shrnutí dosažených výsledků.

Seznam doporučené odborné literatury:

[1] FAIN, Yakov a Anton MOISEEV. Angular Development with TypeScript. 2. Manning Publications, 2018. ISBN 9781617295348.
[2] SAVKIN, Victor a Jeff CROSS. Essential Angular. Packt Publishing Ltd, 2017. ISBN 9781788291040.
[3] SAVKIN, Victor a Jeff CROSS. Angular Router. Packt Publishing, 2017. ISBN 9781787287150.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Datum zadání: 01.09.2021

Datum odevzdání: 30.04.2022

doc. Ing. Petr Gajdoš, Ph.D.
vedoucí katedry

prof. Ing. Jan Platoš, Ph.D.
děkan fakulty

Abstrakt

Tato práce si klade za cíl navrhnout a vytvořit webovou aplikaci pro tvorbu a editaci Garmin TYP souborů. V rámci této práce je nejprve čtenář seznámen s moderními technologiemi pro tvorbu webových aplikací. Poté je popsán TYP soubor, konkrétně jeho význam, struktura, obsah a také knihovna potřebná pro načtení a zpracování tohoto souboru. Následně se práce zabývá popisem návrhu a implementace webového editoru pro TYP soubory. Nakonec je provedeno uživatelské testování prostředí vytvořené aplikace, jehož výstupem jsou poznatky, dle kterých bude aplikace vylepšena.

Klíčová slova

Angular; TypeScript; JavaScript; SPA; webová aplikace; Garmin; TYP soubor

Abstract

This thesis aims to design and implement a web application for editing of Garmin TYP files. First of all, the reader is acquainted with modern technologies for creating web applications. Then the TYP file is described, specifically its meaning, structure, content and also the library needed to load and process this file. Subsequently, the thesis deals with the description of the design and implementation of a web editor for TYP files. Lastly, user testing of the application environment is conducted, the output of which is the information according to which the application will be improved.

Keywords

Angular; TypeScript; JavaScript; SPA; web application; Garmin; TYP file

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Janu Janouškovi za odborné vedení, rady a konzultace při tvorbě této bakalářské práce.

Obsah

Seznam použitých zkratek a symbolů

UI	–	User Interface
HTML	–	Hypertext Markup Language
CSS	–	Cascading Style Sheets
JS	–	JavaScript
TS	–	TypeScript
SPA	–	Single-page application
DOM	–	Virtual Document Object
API	–	Application Programming Interface
JSX	–	JavaScript XML
DI	–	Dependency Injection
CLI	–	Command Line Interface
CDK	–	Component Dev Kit
RxJS	–	Reactive Extensions Library for JavaScript
JSON	–	JavaScript Object Notation
POI	–	Point Of Interest
XPM	–	X PixMap
UTF	–	Unicode Transformation Format
PID	–	Product ID
FID	–	Family ID

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní moderní době, kdy každý může mít jednoduše přístupné mapy skrze své oblíbené chytré zařízení, již zaniká důvod vlastnit fyzické kopie map. Tyto digitální mapy také mohou být upraveny dle preferencí uživatele. A právě zde přicházejí takzvané TYP soubory. Jedná se o formát souboru společnosti Garmin, který umožňuje uživateli přepsat předefinované styly jednotlivých prvků mapy pomocí svých vlastních stylů. Díky TYP souborům můžou mapy vypadat a působit mnohem přehledněji a čitelněji než předtím.

K tvorbě a editaci těchto TYP soborů je nutné použít nějaký dostupný grafický editor TYP souborů. V současné době je k dispozici jen malé množství takových editorů a všechny jsou již velmi zastaralé, což je činí uživatelsky nepřívětivými. Hlavním cílem této práce je tedy navrhnout a vytvořit webový grafický editor TYP souborů s použitím nejmodernějších technologií v oblasti webových aplikací. To sebou také obnáší vytvoření knihovny nástrojů pro načtení a zpracování dat TYP souboru.

Samotná práce je logicky rozdělena do čtyř kapitol. První kapitola se zabývá seznámením a popisem moderních technologií pro tvorbu webových aplikací. Součástí této kapitoly je také porovnání nejpopulárnějších front-end frameworků současnosti. Poté následuje kapitola zabývající se detailním popisem TYP souboru, konkrétně jeho významem, obsahem a strukturou. V další části této kapitoly je rozebrána tvorba knihovny pro zpracování TYP souborů a také tvorba souvisejících nástrojů pro práci s binárními soubory. Dále následuje kapitola pojednávající o hlavním cíli této práce, a to o návrhu a tvorbě webového grafického editoru TYP souborů. Aplikace je zde postupně popsána po jednotlivých komponentách, kde u každé komponenty je vysvětlena její hlavní logika. Poslední kapitola se zabývá uživatelským testováním prostředí aplikace formou scénářů, jehož výstupem jsou poznatky, dle kterých bude aplikace vylepšena.

Kapitola 2

Seznámení s moderními technologiemi pro tvorbu webových aplikací

2.1 Webové frameworky a jejich použití

O pojmu framework lze často slyšet v souvislosti s kódem. Jedná se o sadu nástrojů, která je navržena za účelem pomoci vývojáři s realizací a tvorbou projektů. Umožňuje vývojáři se soustředit převážně na high-level funkcionalitu aplikace, jelikož o low-level funkcionalitu se postará samotný framework – zapouzdří ji od vývojáře. Frameworky se obecně rozdělují na tři základní kategorie, a to na front-end, back-end a na UI frameworky.

Termíny front-end a back-end se týkají oddělení zájmů mezi prezentační vrstvou (front-end) a vrstvou přístupu k datům (back-end) určitého softwaru. V modelu klient-server je klient obvykle považován za front-end a server je obvykle považován za back-end.

2.1.1 Front-end framework

Front-end webové aplikace je ta část (vrstva), která je viditelná a se kterou se interaguje. Skládá se z webového designu a z interakce uživatele s webovou aplikací. Jinak řečeno, jedná se o proces převodu dat do grafické podoby, tedy do grafického rozhraní. Z hlediska jazyků se téměř vždy skládá z HTML, CSS a JavaScriptu.

Front-end frameworky jsou ve většině případů napsány v JavaScriptu a obecně slouží k uspořádání funkčnosti a interaktivity webové aplikace. Primární použití front-end frameworků spočívá v tom, že vytvářejí interaktivní nástroje a vyvíjejí responzivní webové stránky. Vytvářejí konzistentní produkty a vylepšují vzhled a dojem z webových aplikací. Poskytují vývojáři spousty předdefinovaných metod a komponent, čímž urychlují vývoj aplikace. Jednou z významných výhod front-endového frameworku pro vývoj webu je to, že je podporován technologií, kterou lze snadno škálovat, učit se a používat.

Mezi významné zástupce moderních front-end frameworků by se daly například zařadit frameworky Angular, Vue a React. Všechny tři slouží k tvorbě Single Page aplikací (SPA). Single Page aplikace je webová aplikace nebo webová stránka, která komunikuje s uživatelem dynamickým přepisováním aktuální webové stránky novými daty z webového serveru namísto načítání celé nové stránky.

2.1.2 Back-end framework

Back-end, také nazýván jako strana serveru, reprezentuje část aplikace (vrstvu), která se obvykle skládá ze serveru, jenž poskytuje data na vyžádání, aplikace, která má na starost business logiku a databáze pro uchování a organizaci dat. Zjednodušeně řečeno, hlavním účelem back-endu je poskytování dat front-endu.

Back-endové frameworky jsou na rozdíl od front-endových mnohem rozmanitější. Jsou napsány v různých programovacích jazycích a mají širokou škálu funkcí. Mezi významné zástupce současných back-end frameworků by se daly například zařadit frameworky Spring (Java), Django (Python) nebo Node.js (JavaScript).

2.1.3 UI framework

UI frameworky pomáhají vytvářet stylizované a profesionálně vypadající webové aplikace. Většina z nich obsahuje nějakou formu grid systému pro jednodušší umístění a zarovnání prvků uživatelského rozhraní. Dále poskytují předdefinované a jednotné stylování jednotlivých HTML elementů nebo komponent, což umožňuje webové stránce/aplikaci vypadat přehledně a profesionálně. Zde můžeme zařadit frameworky jako například Bootstrap, Angular Material, Ant Design nebo Semantic UI.

2.1.4 Proč zvolit framework?

Použití vhodného frameworku je pro vývojáře zásadní, jelikož umožňuje šetřit důležitý čas a úsilí při vytváření aplikace. Účelem frameworku je umožnit návrhářům a vývojářům soustředit se na vytváření jedinečné funkcionality pro své projekty nežli ji znovu vynalézat.

Framework šetří čas - Pokud je k vývoji aplikace použit pouze samotný programovací jazyk, je nutné začít úplně od začátku. Musí se vkládat funkce do tříd a proměnných atd. Když je však použit framework, funkce, třídy a proměnné jsou v něm již vloženy. Takže je možné použít framework přímo k provedení konkrétního úkolu, který je vyžadován.

Robustnost - O frameworku lze říci, že je robustní, protože byl vyvinut týmem vývojářů, nikoliv samotným vývojářem. Před uvedením frameworku na trh je provedena celá řada testů, takže je velmi malá šance, že framework nebude fungovat tak, jak je od něj očekáváno. Lze tvrdit, že metody frameworku budou obvykle efektivnější než vlastní kód.

Bezpečnost - Zabezpečení je jedním z nejdůležitějších aspektů při tvorbě aplikace. Obzvláště když se vytváří aplikace od začátku čistě pomocí programovacího jazyka, je nutné zvážit všechny

bezpečnostní díry. Vývoj aplikace pomocí frameworku je však mnohem bezpečnější, protože kód ve frameworku je již pečlivě testován před jeho uvedením na trh.

Škálovatelnost - Framework je škálovatelnější než vlastní kód, který byl vytvořen pro provádění nějaké konkrétní funkce. Je to proto, že frameworky jsou již testovány řadou dalších vývojářů, a proto s největší pravděpodobností budou fungovat lépe než vlastní kód, který byl vytvořen pro provádění stejné sady funkcí.

Pokud se podíváme na jakékoliv významné frameworky, zjistíme, že jsou vyvíjeny v týmech vývojářů s více než deseti členy, kteří jsou experti v dané oblasti. Frameworky jsou stavěny tisíci vývojářů a testovány miliony uživatelů před tím, než jsou oficiálně uvedeny na trh. A toto jsou hlavní důvody, proč je velmi obtížné konkurovat frameworku vlastním kódem.

2.1.5 Jak zvolit framework?

V softwarovém průmyslu je vývojář obklopen spoustou frameworků, ze kterých si lze vybrat. Je ale velmi důležité si umět vybrat pouze ty frameworky, které jsou vhodné pro vývoj dané aplikace. Před použitím frameworku je doporučováno udělat si průzkum. Bez řádného průzkumu se může stát, že je zvolen framework, který bude nutné přizpůsobovat vlastním potřebám, i když existuje jiný framework, jenž dokáže přesně plnit dané potřeby. Mezi hlavní kritéria pro výběr patří:

Popularita - Čím známější a uznávanější framework bude, tím více bude „živý“ a vyvíjející se: nové nápady, rostoucí kvalita a počet rozšíření atd. Popularita má také vliv na počet vývojářů disponujících znalostmi daného frameworku. Čím větší popularita, tím je vyšší šance, že se povede nalézt členy vývojového týmu pro aplikaci používající daný framework.

Vlastnosti/Schopnosti - Základní kritérium při výběru frameworku je, že bude vyhovovat daným potřebám a nebude ho nutno nijak významně upravovat.

Podpora - Dalším kritériem, které by nemělo být přehlíženo, je snadnost nalezení odpovědí na případné otázky a problémy, jež mohou při vývoji nastat. Například získání odpovědí na různých komunitních fórech. Toto kritérium velmi úzce souvisí s popularitou frameworku.

Dokumentace - Je nezbytné vyhodnotit povahu, množství a kvalitu existující dokumentace o frameworku. Dobře zdokumentovaný framework se snáze používá a dá se s ním rychleji seznámit.

2.1.6 Porovnání populárních front-end frameworků

React - React je open-source framework vyvinutý a vytvořený společností Facebook. Podle průzkumu Stack Overflow Developer 2021 je React nejpopulárnějším webovým frameworkem, používán většinou front-end vývojářů. Primárním záměrem Reactu bylo vyřešit problémy s údržbou kódu kvůli neustálému přidávání funkcionality do aplikace. Od ostatních front-end frameworků se React odlišuje díky svému Virtual Document Object Modelu (virtual DOM). Kromě toho je tento framework také uživatelsky přívětivý pro nové vývojáře – lze nalézt dostatečné množství studijních materiálů, tutoriálů, ale také i mnoho rad na komunitních fórech.

Mezi hlavní výhody Reactu patří jeho schopnost rozdělit aplikaci na dílčí části do komponent. Díky tomu je možné opětovně zužitkovat již vytvořený kód, což vývojáři značně urychluje práci, ale také díky tomu dělá kód čitelnějším. Další předností Reactu je již zmíněný virtual DOM. React vytvoří strom vlastních objektů představujících část DOM. Například namísto vytvoření skutečného prvku DIV obsahujícího prvek UL vytvoří objekt React.div, který obsahuje objekt React.ul. React dokáže s těmito objekty manipulovat velmi rychle, aniž by se skutečně „dotýkal“ DOMu nebo procházel DOM API. Poté když vykreslí komponentu, použije tento virtuální DOM, aby zjistil, co potřebuje udělat se skutečným DOM, aby se oba stromy shodovaly. Virtual DOM si lze představit jako nákres nebo nějaký plán, který obsahuje všechny detaily potřebné ke konstrukci DOMu, ale protože nevyžaduje všechny části, které jsou součástí skutečného DOMu, lze jej vytvořit a změnit mnohem snadněji. Jak již bylo zmíněno, React má značnou komunitní podporu. Díky této podpoře existuje široké spektrum různých React knihoven pro dodatečné usnadnění vývoje aplikace.

Za hlavní nevýhodu Reactu by se dal považovat nedostatek současné dokumentace. React se velmi rychle vyvíjí, což má za následek absenci dokumentace k novým vlastnostem. ReactJS používá JSX. JSX je rozšíření syntaxe, které umožňuje smíchat HTML s JavaScriptem. Tento přístup má své výhody, ale někteří členové vývojářské komunity považují JSX za bariéru, zejména noví vývojáři.

Angular - Jakýkoliv seznam předních front-end frameworků by byl neúplný, aniž by byl zmíněn Angular. Angular je framework založený na TypeScriptu. Byl vytvořen společností Google, aby propojil mezeru mezi rostoucími požadavky technologie a konvenčními způsoby vývoje. Na rozdíl od Reactu je Angular exkluzivní se svou vlastností obousměrné vazby dat. To znamená, že existuje skutečná časová synchronizace mezi pohledem a modelem, kdy se jakákoliv změna v modelu okamžitě aplikuje na pohled a naopak. Pro nové vývojáře je k tomuto frameworku spousta tutoriálů a velmi dobrá dokumentace.

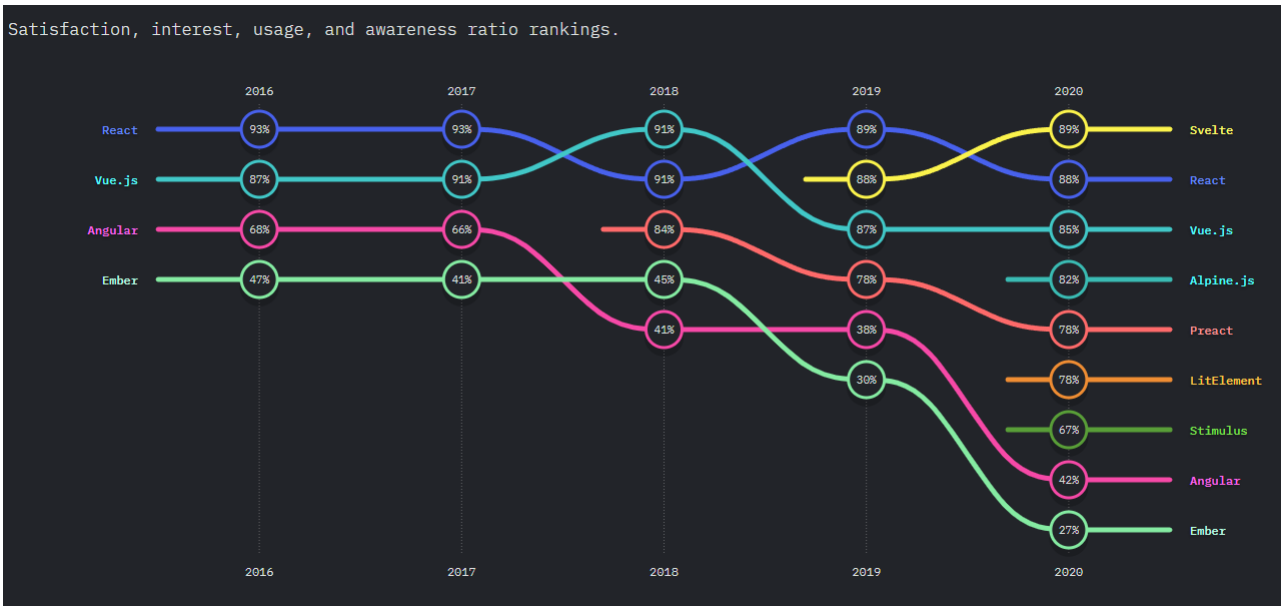
Taktéž jak React, Angular umožňuje rozdělení aplikace na dílčí části do komponent. Což podporuje znovu použitelnost, škálovatelnost a údržbu kódu. Mezi jednu z hlavních výhod Angularu patří již zmíněná obousměrná vazba dat. Když se změní data v modelu, změní se také pohled. Obousměrná datová vazba umožňuje zkrátit dobu vývoje, protože není vyžadováno psaní dalšího kódu pro zajištění nepřetržité synchronizace pohledu a modelu. Další výhodou Angularu je Dependency injection (DI). Třídy mohou zdědit externí logiku, aniž by věděly, jak ji vytvořit. Spotřebitelé těchto tříd také nemusí nic vědět. DI šetří třídy i spotřebitele od nutnosti vědět více, než je nutné. Přesto je kód stejně modulární jako dříve díky mechanismům podporujícím DI v Angularu. Díky DI jsou komponenty více znovupoužitelné, jednodušší na správu a na testování. Angular je napsán pomocí jazyka TypeScript, což je v podstatě nástavba JavaScriptu, která jej rozšiřuje o statické typování a další atributy z objektově orientovaného programování. Plně se kompiluje do JavaScriptu a pomáhá odhalit a odstranit běžné chyby při psaní kódu. Díky TypeScriptu je kód čitelnější. Angular byl vytvořen velkou a významnou společností Google, což představuje záruku dlouhodobé podpory. I samotný Google oznámil dlouhodobou podporu pro tuto technologii. Komunita frameworku Angular je taktéž velmi rozsáhlá. Jsou k dispozici mnohé knihovny pro usnadnění vývoje.

Za jeden z nedostatků Angularu by se dala považovat nedostatečná dokumentace Angular CLI (Command Line Interface). CLI je pro Angular vývojáře velmi důležitý nástroj, ale v oficiální dokumentaci k ní není dostatek informací, což nutí vývojáře prozkoumávat vlákna na Githubu či jinde dohledávat potřebné informace. Pro začínající vývojáře by se mohlo jednat o trochu složitější framework na naučení, jelikož Angular má svůj vlastní rozsáhlý ekosystém. Také trvá na použití TypeScriptu, což představuje další novou technologii, kterou se musí začínající vývojář naučit.

Vue - Vue je jedním z nejjednodušších front-end frameworků dnešní doby. Byl navržen pro škálovatelnost spolu se snadnou integrací s jinými knihovnami zaměřenými na zobrazovací vrstvu. Má malou velikost a představuje dvě hlavní výhody – virtuální DOM a možnost tvorby komponent. Využívá také obousměrnou datovou vazbu. Ačkoli je Vue framework navržen tak, aby se vypořádal se složitostí a zlepšil výkon aplikací, není obecně populární pro tvorbu větších projektů.

Jednou z hlavních výhod tohoto frameworku je virtual DOM, což přináší pozitivní vliv na výkonnost aplikace. Vue také podporuje obousměrnou datovou vazbu, která slouží pro zajištění nepřetržité synchronizace pohledu a modelu. Dále Vue umožňuje tvorbu komponent, čímž podporuje znovu použitelnost a škálovatelnost. Vue je mnohem méně komplexní framework v porovnání s Angularem či Reactem, takže představuje dobrou volbu pro začínající vývojáře. A to také díky své dobře zpracované a srozumitelné dokumentaci.

Vue je stále velmi mladý framework. Jeho komunita a velikost vývojového týmu je stále nesrovnatelná s vyspělejším Angularem čí Reactem, za kterými stojí velké společnosti. To je také jeden z důvodů, proč je Vue používán převážně v menších projektech.



Obrázek 2.1: Vývoj popularity front - end frameworků [1]

2.2 Moderní webové technologie použité pro implementaci TYP Editoru

Tato kapitola má za cíl seznámit čtenáře s moderními technologiemi, jež se běžně používají při tvorbě webových aplikací/stránek. Zde budou konkrétně popsány ty technologie, které byly užity při tvorbě webové aplikace TYP Editoru. Veškeré zmíněné technologie jsou zdarma k užití.

2.2.1 Angular Material

Angular Material je UI framework vyvinutý společností Google. Jeho cílem je pomoci vývojářům navrhnout responzivní a rychlé UI aplikace strukturovaným a přehledným způsobem s pomocí komponent. Jeho komponenty umožňují snáze vytvářet atraktivní, konzistentní a funkční webové stránky či webové aplikace. Je navržen dle předloh Material designu, který udává vizuální, pohybový a interakční design.

Mezi hlavní vlastnosti Angular Materialu patří jeho **responsivní design**. Díky responsivnímu UI designu aplikace vypadá přehledně a čitelně na širokém množství různě velkých zařízení – od mobilních telefonů až po stolní počítače. Dále poskytuje spousty **UI komponent**. Mezi nimi lze nalézt jak základní UI prvky, jako různé druhy tlačítek a vstupů, tak i specializované prvky jako například záložky, dialog, karty, expanzní panel, menu a mnohé další. Jednotlivé komponenty jsou kvalitně navrženy, mají své individuální vstupní parametry, skrze které lze upravit jejich vzhled či chování. Vzhled komponent lze dále modifikovat a přepsat pomocí vlastního CSS. Krom komponent poskytuje také **Component Dev Kit (CDK)**, což je knihovna předdefinovaných chování v Angular Material. CDK umožňuje používat funkce, které nezávisí na Angular Material a jeho stylu. Jedná se tedy o univerzální nástroje pro vytváření komponent, které jsou plně nezávislé. Patří zde například funkce Drag & Drop nebo virtuální scroll. Za další užitečnou vlastnost Angular Material by se dala považovat jeho schopnost tvorby předefinovaných Angular komponent dle šablony, tzv. **Angular Material Schematics**. V podstatě se jedná o sadu příkazů, díky nimž lze vygenerovat nové komponenty, jež jsou běžně používány jako například navigační panel, tabulka, dashboard, formulář a jiné. Další silnou stránkou Angular Material je jeho **dokumentace**. Dokumentace je kvalitně a přehledně zpracována. Lze v ní nalézt seznam všech jeho komponent a dalších možností. Jednotlivé komponenty jsou detailně popsány, konkrétně jejich API. U detailu každé komponenty nechybí ani demo ukázka možného využití dané komponenty. Za zmínku také stojí to, že Angular Material poskytuje **dlouhodobou podporu**.

2.2.2 Angular Material Extensions

Angular Material Extensions je UI knihovna vytvořená za účelem rozšíření Angular Material frameworku. Taktéž jak Angular Material dodržuje designovou předlohu Material designu. Jedná se tedy skutečně o takové rozšíření Angular Materialu ve formě nových komponent. Nelze jej použít

samostatně, pro použití vyžaduje instalaci Angular Materialu. Poskytuje užitečné komponenty jako například: **Color Picker** – komponenta sloužící pro výběr barvy z palety barev, **Data Grid** – komponenta tabulky s rozšířenými možnostmi oproti standardní tabulce v Angular Material (loading bar, výběr konkrétních buněk/řádků/sloupců...), **Date Time Picker** – komponenta umožňující výběr nejen data, ale i času.

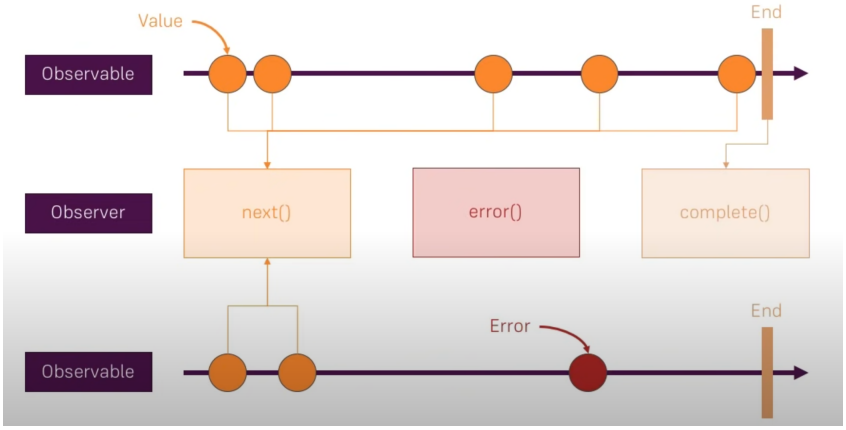
Angular Material Extension je velmi mladá knihovna, která se stále teprve vyvíjí. To může mít za následek velké změny při přechodu na novější verzi, což může vyústit ve větší zásah do kódu. Jinak je ale tato knihovna dobře zdokumentovaná včetně názorných ukázek použití jednotlivých komponent.

2.2.3 RxJS (Reactive Extensions for JavaScript)

RxJS je JavaScriptová knihovna, která používá tzv „observables“ pro reaktivní programování, tedy pro programování orientované kolem datových toků a šíření změn. Lze jej použít a kombinovat s jinými JavaScriptovými knihovnami či frameworky a dobře se integruje i do frameworku Angular. Hlavním cílem RxJS je zjednodušení tvorby asynchronního kódu, kódu založeného na zpětném volání (callback-based code) nebo kódu založeného na událostech (event-based code).

RxJS má některé základní funkce, které pomáhají zvládnout asynchronní implementaci. První z nich je **Observable**. Observable představuje wrapper nad proudem dat (hodnot), jenž umožňuje publikovat události. Observables mají dvě metody: subscribe (přihlášení) a unsubscribe (odhlášení). Reakci na události z observable lze spustit přihlášením se k odběru dané observable. Observable je v podstatě funkce, která může v průběhu času vracet proud hodnot observerovi (pozorovatelovi), a to buď synchronně, nebo asynchronně. Vrácené hodnoty se mohou pohybovat od nuly až do nekonečného rozsahu hodnot. Instance observable prochází během svého životního cyklu těmito čtyřmi fázemi: Creation, Subscription, Execution, Destruction. K tomu, aby observables fungovaly, musí existovat **observers** (pozorovatelé). Jak již bylo zmíněno, observables jsou wrappery nad proudem hodnot. Pokud dojde ke změně hodnot nebo pokud přibude nějaká nová hodnota, observer vykoná určité instrukce jako reakci na změnu v proudu hodnot. Observable je připojený k observerovi skrze **subscription** (předplatné) - pomocí metody subscribe se observer připojuje k observable, aby provedl blok kódu. O správu subscription se stará plánovač. Observer má metody next(), error(), a complete(), které jsou volány v případě interakce s observable. RxJS dále nabízí **operátory**. Operátory jsou funkce, které staví na základě prvků observables a umožňují provádět určité akce s událostmi vyvolanými pozorovanými objekty. Jedná se tedy o jakousi sofistikovanou manipulaci s kolekcemi. Operátory umožňují snadné sestavení složitého asynchronního kódu deklarativním způsobem. Operátory se dělí na dva druhy, a to na Pipeable Operators a na Creation Operators. **Pipeable Operator** je v podstatě čistá funkce, která bere jednu observable jako vstup a generuje další observable jako výstup. Přihlášením se k odběru výstupní Observable dojde také k přihlášení se k odběru vstupní Observable. Na rozdíl od pipeable operátorů, **Creation Operators** jsou funkce,

které lze použít k vytvoření Observable s určitým společným předdefinovaným chováním nebo spojením jiných Observable. Například: of(1, 2, 3) vytvoří observable, jenž bude emitovat hodnoty 1,2 a 3 přímo po sobě. Mezi příklady RxJs operátorů patří například map(), filter(), concat() a flatMap(). Další klíčovou vlastností RxJS je **Subject**. RxJS Subject je speciální typ Observable, který umožňuje multicasting hodnot mnoha observerům. Zatímco prosté Observable jsou unicast (zdrojová observable emitující hodnoty může být sledována - subscribed - pouze jedním observerem), Subjects jsou multicast. Z toho plyne, že běžný observable by měl být používán, když je potřeba pouze jeden observer nebo pokud nezáleží na tom, že observer, který přijde jako první, skončí jako první, dokud druhý nezíská své hodnoty. Naproti tomu Subject je vhodné použít, pokud je potřeba více observerů a záleží na tom, aby všichni observers dostávali své nové hodnoty současně.



Obrázek 2.2: Komunikace Observer - Observable [2]

Mezi hlavní výhody knihovny RxJS patří její flexibilita. RxJS může být použito v kombinaci s jakýmikoliv jinými JS frameworky či knihovnami. Reaktivní programování a RxJS nejsou snadné technologie na naučení se, ale RxJS naštěstí poskytuje velmi dobrou a podrobnou dokumentaci vysvětlující vše potřebné. RxJS je taky soběstačné, nemá žádné závislosti na třetích stranách. Jako každý jiný nástroj má RxJS také své nevýhody. Jednou z nevýhod je obtížnější debugging. Debugging kódu obsahujícího observables není nejjednodušší. Za další nevýhodu by se dala považovat neměnnost dat. Potřeba neměnnosti dat v RxJS sice není přesně požadavkem, ale reaktivní programování funguje nejlépe v kombinaci s funkcionálním programováním.

2.2.4 NG2 – Charts

Modul ng2-charts je open-source JavaScriptová knihovna vytvořená výhradně pro Angular 2+ a je dostupná prostřednictvím npm. Pomáhá vytvářet visuálně poutavé, přehledné a responsivní grafy v Angularu s pomocí Chart.js. Chart.Js je dobře známá JavaScriptová knihovna a používá se k reprezentaci dat pomocí HTML5 canvasu. Umožňuje vytvářet dynamické i statické grafy a přichází s plnou podporou animací pro různé grafy. Přebírá data ve formě JSON, tudíž je snadné ji používat s jakýmkoli programovacím jazykem. Ng2-charts je v podstatě wrapper pro knihovnu Chart.js umožňující integraci jejích grafů do Angularu s využitím Angular directives.

Ng2-charts umožňuje vytvořit celkem 10 typů grafů, a to: Line Chart, Bar Chart, Doughnut Chart, Radar Chart, Pie Chart, Polar Area Chart, Bubble Chart, Scatter Chart, Dynamic Chart a Financial Chart. Vlastnosti těchto grafů lze měnit a nastavovat pomocí atributů. Hlavním z těchto atributů je atribut **type**. Pomocí něj lze nastavit konkrétní typ grafu (line, bar, pie...). Dalším atributem jsou **data**. Skrze data je grafu předána datová struktura, která má být vykreslena. Jsou podporovány různé flexibilní formáty dat jako například Primitive[] nebo Object[]. V závislosti na typu použitého grafu mohou být alternativně použity vlastnosti **labels** a **datasets** pro specifikaci individuálních nastavení (ChartData<TType, TData, TLabel>). Atribut labels představuje popisky k datasetu. Labels se přiřazují v takovém pořadí, v jakém jsou v datasetu. Samostatný atribut datasets funguje úplně stejně jako atribut dataset uvnitř atributu data (ChartDataset<TType, TData>[]). Dalším atributem grafu je **legend**. Tento atribut slouží ke specifikaci popisku grafu. Dále lze nastavit dodatečně nastavení grafu skrze atribut options. Pomocí options je možné nastavit cokoliv dodatečného, co umožňuje knihovna Chart.js, jako například specifikovat vlastní barevnou paletu nebo měřítko (scale). Krom atributů Ng2-charts poskytuje také detekci událostí (events). Událost **chartClick** je spuštěna, když dojde ke kliknutí na graf. Vrátí informace o aktivních bodech a popiscích (labels). Událost **chartHover** je spuštěna, když dojde k pohybu myší (najetí myší) na graf. Taktéž vrátí informace o aktivních bodech a popiscích. Zde je ukázka vytvoření koláčového grafu (pie chart):

```
<div class="chart-wrapper">
    <canvas baseChart
    [data]="pieChartData"
    [labels]="pieChartLabels"
    [chartType]="pieChartType"
    [options]="pieChartOptions"
    [legend]="pieChartLegend">
   (chartClick)="onChartClick(($event)"
  </canvas>
</div>
```

Výpis kódu 2.1: Ukázka koláčového grafu v HTML

2.2.5 Angular Flex Layout

Angular flex-layout je samostatná knihovna vyvinutá týmem Angular pro tvorbu komplexních rozvržení UI elementů webové aplikace/stránky. Angular flex-layout poskytuje sofistikované responsivní API s použitím CSS Flexbox, CSS Grid, a mediaQuery, které usnadňuje rozmístění různých komponent či jiných prvků. Toto responzivní API umožňuje vývojářům specifikovat různé typy rozvržení, velikosti, viditelnosti, velikosti viewportu (viditelná oblast webové stránky) a zobrazovací zařízení. Tato knihovna je dostupná pouze pro framework Angular a v současné době podporuje verzi Angularu 4.1 a vyšší.

Angular flex-layout využívá HTML značek (HTML markup) ke specifikaci požadované konfigurace rozvržení. Ke komunikaci s Flexbox kontejnery se používají tyto direktivy: fxFlex, fxLayout, fxLayoutGap a fxFlexOrder. Direktiva **fxFlex** je zodpovědná za změnu velikosti prvků (flex-item) podél hlavní osy rozvržení a měla by být použita na podřízené prvky uvnitř fxLayout kontejneru. Akceptuje tři typy parametrů: flex-grow, flex-shrink a flex-basis. **Flex grow** udává, o kolik poroste flexbox prvek (item) vzhledem k ostatním flexbox prvkům uvnitř stejného flex kontejneru, pokud je dostatek místa. **Flex shrink** určuje, jak moc by se měl prvek flexbox zmenšit vzhledem ke zbytku flexbox položek ve stejném flex kontejneru, když není dostatek volného místa. V CSS vlastnost **flex-basis** určuje výchozí velikost prvku flexboxu, než je změněna jinými vlastnostmi flexboxu, jako je flex-grow a shrink. Ale v Angular flex-layout, jakmile je zadána počáteční velikost, flex prvek se nezvětší ani nezmenší, i když se nastaví vlastnosti flex-grow a shrink.

```
<div fxFlex="<flex-grow> <flex-shrink> <flex-basic>"></div>
```

Výpis kódu 2.2: Použití direktivy fxFlex

Jako další je direktiva fxLayout. **FxLayout** definuje „tok“ podřízených prvků podél hlavní osy nebo křížové osy uvnitř flex kontejneru. V závislosti na rozložení lze předat čtyři různé hodnoty atributu fxLayout: row, column, row-reverse a column-reverse. Dále také akceptuje další parametry jako wrap a inline. **FxLayout row** zobrazí podřízené flex prvky podél hlavní osy, tj. vodorovné osy, uvnitř flex kontejneru. **FxLayout column** zobrazí podřízené flex prvky podél svislé osy uvnitř flex kontejneru. Hodnoty column-reverse a row-reverse fungují úplně stejně, pouze s tím rozdílem, že uspořádají prvky z opačného směru. Další možnou nastavitelnou hodnotou je **fxLayout wrap**. Ten slouží k řešení problému, když se uvnitř flexboxu nachází více položek, například za sebou, ale nejsou všechny viditelné z důvodu nedostatku místa. FxLayout wrap se dělí na **row wrap** (zarovnání do více řádků) a na **column wrap** (zarovnání do více sloupců).

```
<mat-card fxLayout="row wrap" fxLayoutGap="30px">
    <mat-card class="child-1">1. One</mat-card>
    <mat-card class="child-2">1. Two</mat-card>
    <mat-card class="child-3">1. Three</mat-card>
    <mat-card class="child-3">1. Four</mat-card>
    <mat-card class="child-3">1. Five</mat-card>
    <mat-card class="child-3">1. Six</mat-card>
</mat-card>
```

Výpis kódu 2.3: Použití direktivy fxLayout

Další direktiva je **fxLayoutGap**. FxLayoutGap se používá k určení mezery mezi podřízenými flex prvky uvnitř flex kontejneru. Poslední direktivou je **fxFlexOrder**. FxFlexOrder definuje poziční upořádání prvků uvnitř flex kontejneru.

```
<mat-card fxLayout="column">
    <mat-card class="child-1" fxFlexOrder="3">1. Children</mat-card>
    <mat-card class="child-2" fxFlexOrder="2">2. Children</mat-card>
    <mat-card class="child-3" fxFlexOrder="1">3. Children</mat-card>
</mat-card>
```

Výpis kódu 2.4: Použití direktivy fxFlexOrder

Za jednu z hlavních výhod knihovny Angular flex-layout by se dalo považovat to, že je napsána čistě v **TypeScriptu**. Jedná se tedy o čistě TypeScriptový UI layout engine na rozdíl od jiných implementací knihoven pro rozvržení používající jen CSS nebo JS + CSS stylesheets jako například AngularJS Material Layouts. V normálním CSS flexboxu nebo CSS gridu je nutné napsat složitý CSS kód pomocí mediaQueries, abychom vytvořili responzivní rozvržení, což bývá také často těžko pochopitelné. Ale s pomocí Angular flex-layout můžeme přímo definovat nastavení rozložení flexboxu uvnitř HTML šablony pomocí direktiv flexboxu. Kód je takto čitelnější a jednodušší na údržbu. Další výhodou Angular flex-layoutu je jeho **nezávislost** na Angular Materialu, může tedy být použit v kombinaci i s jinými UI frameworky/knihovnami. Jak již bylo zmíněno v úvodním odstavci, za flex-layoutem stojí vývojový tým samotného Angular frameworku, což představuje určitou záruku ve formě kvality, podpory a spolehlivosti zpracování.

2.2.6 Web storage API

Web storage je technlogie, která umožňuje aplikaci ukládat data lokálně v prohlížeči uživatele. Web storage přišlo s HTML5 jako náhrada za Cookies. Web storage je bezpečnější než Cookies a umožňuje ukládat velké množství dat lokálně, aniž by to ovlivnilo výkon webové aplikace. Web storage poskytuje dva typy objektů k ukládání dat, a to LocalStorage a SessionStorage. **LocalStorage**

ukládá data bez jakéhokoliv data expirace, naopak **SessionStorage** ukládá data pouze po dobu jednoho sezení (session) – se zavřením záložky v prohlížeči jsou data ztracena.

Oba tyto web storage objekty umožňují ukládání dat v podobě párů klíč – hodnota, kde hodnota i klíč musí být datového typu string. Mají i stejné metody a vlastnosti pro manipulaci s daty:

- setItem(klíč, hodnota) – uložení páru klíč – hodnota
- getItem(klíč) – získání hodnoty na základě klíče
- removeItem(klíč) – odstranění hodnoty na základě klíče
- clear() – smaže všechna data
- key(index) – získání klíče na dané pozici skrze index
- length – počet uložených položek

Web storage API je velmi jednoduché a přívětivé API pro lokální ukládání méně komplexních dat v rámci prohlížeče. Nejedná se tedy v žádném případě o náhradu za serverovou databázi, hodí se spíše pro ukládání dat, která jsou relevantní pro jednoho konkrétního uživatele, například obsah košíku v e-shopu.

2.2.7 HTML5 Canvas

Canvas je prvek značkovacího jazyka HTML, který byl přidán v rámci specifikace HTML5. Značí se pomocí HTML tagu **<canvas>**. Slouží k dynamickému vykreslování 2D a 3D bitmap a grafických primitiv. Canvas je ale pouze „kontejner“ pro grafiku, k samotnému vykreslení obsahu je zapotřebí skriptovací jazyk jako například JavaScript. Prvek canvas má dva atributy, a to **výšku** (height) a **šířku** (width). Ve výchozím nastavení nemá canvas žádný obsah ani okraj.

K práci s canvas prvkem se používá **Canvas API**. Canvas API poskytuje prostředky (metody) pro kreslení grafiky pomocí JavaScriptu a canvas prvku. Mimo jiné lze Canvas API použít pro animace, herní grafiku, vizualizaci dat, manipulaci s fotografiemi a zpracování videa v reálném čase. Canvas API se převážně zaměřuje na 2D grafiku. WebGL API, které také používá canvas prvek, kreslí hardwarově akcelerovanou 2D a 3D grafiku. Canvas lze vytvořit následujícím způsobem:

```
<canvas id = "myCanvas" width ="200" height ="150"> </canvas>
```

Výpis kódu 2.5: Použití prvku Canvas v HTML

Canvasu bylo takto přiřazeno ID pro použití v JavaScript kódu a také šířka a výška. Nyní lze do canvasu vykreslit grafický obsah:

```
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');
ctx.fillStyle = 'green';
ctx.fillRect(10, 10, 150, 100);
```

Výpis kódu 2.6: Ukázka vykreslení obsahu do Canvasu

Metoda **document.getElementById()** získává referenci na HTML element <canvas> dle zadaného ID. Dále metoda **canvas.getContext()** získá kontext tohoto prvku – to, kde bude vykreslen grafický obsah. Samotné kreslení se provádí pomocí rozhraní **CanvasRenderingContext2D**. Vlastnost fillStyle nastaví barvu výplně obsahu. Dále metoda **fillRect()** umístí obdélník do canvasu dle x,y souřadnic levého horního rohu a nastaví mu šířku a výšku.

Kapitola 3

Návrh a implementace knihovny pro tvorbu a editaci TYP souborů

V této kapitole bude rozebrán návrh knihovny pro práci s TYP souborem, ale i TYP soubor samotný. Konkrétně jeho obsah a význam jeho obsahu pro mapy Garmin.

3.1 TYP soubor

Původně měly všechny Garmin mapy jednotný a neměnný styl. To se ale změnilo s příchodem TYP souborů, které umožňují uživateli tyto předefinované styly jednotlivých prvků mapy přepsat pomocí svých vlastních stylů. Díky TYP souborům mohou mapy vypadat a působit mnohem přehledněji a čitelněji než předtím. To byl velký krok vpřed, jelikož bylo možné od sebe rozlišit například různé druhy cest, lesů a dalších míst, běžně se vyskytujících v mapách. S použitím souboru TYP se tedy může uživatel rozhodnout, které prvky se zobrazí a jak. Například cyklisté mohou mít jinak stylizovanou mapu (s důrazem na cyklostezky) než chodci.

TYP soubor v sobě obsahuje celkem tři základní prvky: body zájmu (POIs – points of interest), cesty jako například dálnice nebo cyklostezka (Polylines) a polygony (lesy, pole, rybníky...). Každý prvek má své jedinečné identifikační číslo, konkrétně **typ** a **podtyp**, díky kterému jej lze jednoznačně identifikovat a určit, co reprezentuje. Tato čísla nejsou náhodně vytvořena, odpovídají oficiálnímu seznamu Garmin. Všechny tyto tři prvky umožňují mít jak svou **denní verzi**, tak i **noční verzi**. Při přepnutí mapy do nočního režimu lze takto zachovat jednotný a přehledný styl map. Dále tyto prvky mohou obsahovat text ve formě nějakého popisku ke každému prvku. Tento popisek je přidružen k **jazyku**, ve kterém byl napsán. Těchto popisků může prvek obsahovat více, avšak pouze jeden ke každému jazyku. Všem prvkům lze dále nastavit **velikost a barva fontu**. Celkem lze nastavit dvě barvy fontu, jednu pro denní reprezentaci a druhou pro noční.

3.1.1 Polygon

Prvním z těchto tří prvků jsou **polygony**. Polygony v mapách reprezentují velké plochy, pod nimiž si lze například představit lesy, pole, rybníky, moře, parkoviště a mnohé další. Tyto plochy budou v mapě zaplněny opakovaným vykreslením daného polygonu. Velikost jednoho polygonu je pevně daná. Každý polygon je čtverec o délce jedné strany **32 pixelů**. Polygony se dají rozdělit na dva hlavní druhy, a to na **bitmapové** a **bez-bitmapové**. Bez-bitmapové polygony mohou obsahovat pouze dvě barvy – jednu pro denní a druhou pro noční verzi polygonu. Zatímco bitmapové polygony umožňují obsah až čtyř barev – dvě denní a dvě noční. Obsah dvou barev umožňuje bitmapovým polygonům mít v sobě nadefinovaný vzor formou bitmapy. Tento vzor (bitmapa) je sdílený mezi denním a nočním polygonem, liší se tedy pouze barvou. Další vlastnost, kterou mají polygony, je **draworder**. Draworder reprezentuje úroveň polygonu, do které bude vykreslen. Čím vyšší je tato úroveň, tím výše bude polygon vykreslen. Nekorektní hodnota draworder může vyústit ve špatné vykreslení polygonu. Může se tak stát, že polygon nebude v mapě vůbec viditelný, jelikož bude překrytý jiným polygonem. Mezi jednotlivými úrovněmi (hodnotami draworder) nesmí být mezery. Polygon s nejvyšší úrovní vykreslení je ten poslední bez mezery. Polygon je jediný prvek, který má tuto vlastnost odlišného pořadí vykreslení. Bez specifikace draworder nelze polygon vykreslit.

3.1.2 Polyline

Dalším prvkem je **polyline**. Polyline slouží v mapách k reprezentaci všech cest, od cyklostezky až po dálnice. Délka jedné polyline je neměnná, vždy je **32 pixelů** dlouhá, zatímco šířka může být změněna. Taktéž jak polygony i polylines se dělí na **bitmapové** a **bez-bitmapové**. Bez-bitmapové polylines můžou mít až čtyři barvy – dvě denní a dvě noční. Dále tyto bez-bitmapové polylines mají definovanou **šířku čáry** a **šířku ohraničení**. Jednobarevné bez-bitmapové polylines mají definovanou šířku čáry a šířku ohraničení mají nulovou, naopak dvoubarevné polylines mají definovanou jak šířku čáry, tak i šířku ohraničení. Tímto typem polyline jsou definovány převážně hlavní silnice. Bitmapové polylines umožňují taktéž obsah až čtyř barev – dvě denní a dvě noční, ale nejsou již definovány skrze šířku čáry a ohraničení. Místo toho jsou definovány pouze přes **šířku bitmapy** a jejich vzhled je dán bitmapou samotnou. Bitmapa je taktéž jak u polygonů sdílená mezi noční a denní polyline. V případě bitmapových polylines je zvykem, že je pouze jedna barva viditelná, druhá barva je plně transparentní. Tímto typem polylines jsou obvykle definovány vedlejší cesty jako například turistické stezky.

3.1.3 POI

Posledním hlavním prvkem jsou **POIs** (body zájmu). POIs neboli místa zájmu obvykle reprezentují v mapách místa jako například čerpací stanice, restaurace, kavárny, hotely a mnohé další. Na rozdíl od polylines nebo polygonů mají POIs pouze bitmapu, neexistuje žádný typ POI bez bitmapy. Dále také nemají žádný limit na počet barev. Jejich velikost je plně volitelná, šířku i výšku lze nastavit

nezávisle. Další vlastnotí POIs je nezávislá bitmapa. Denní POI se může plně lišit od noční POI ve všem kromě velikosti.

3.1.4 Active routing

Garmin mapy mají schopnost zvýraznit oblíbené trasy v závislosti na aktivitě uživatele – cyklistika, horská cyklistika, horolezectví, turistika, pěší chůze. Tato schopnost se nazývá **Active routing**. Styl tohoto zvýraznění tras lze taktéž nastavit skrze TYP soubor. Vlastnosti zvýraznění plně odpovídají vlastnostem prvku polyline. Jedná se v podstatě o polyline, jen s jiným využitím, a to pro reprezentaci oblíbených tras.

3.1.5 Extra POIs

TYP soubor může také obsahovat **extra POIs**. Ty slouží k zobrazení dodatečných služeb, budov, hotelů a dalších. Díky extra POIs lze přiřadit jednomu konkrétnímu typu POI více symbolů (ikon). Například čerpací stanice nemusí mít jeden a ten samý konkrétní symbol, ale mohou být rozděleny podle značky na Shell, Benzina, OMW..., kde každá značka bude mít svůj vlastní symbol. Aby došlo ke korektnímu přiřazení extra poi k danému místu, je nutno při vytvoření specifikovat ID, jež dané místo reprezentuje. Například čerpací stanice Shell má ID = 10. Extra POIs sdílí veškeré vlastnosti se základním prvkem POI. Každý extra POI odkazuje na rodičovskou ikonu, která je také vykreslena pro dané umístění. Jelikož jsou vykresleny obě ikony (jak rodičovská, tak extra), je proto nutné se ujistit, že rodičovská ikona je plně průhledná.

3.1.6 Hlavička

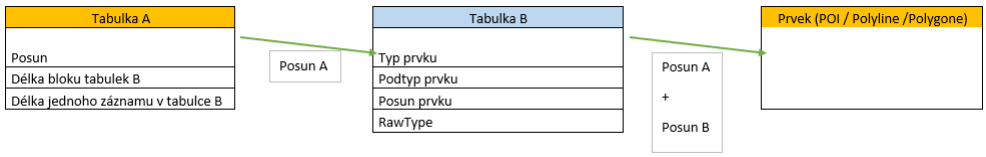
Na začátku každého TYP souboru se nachází hlavička souboru, která obsahuje všechny potřebné informace pro načtení dat ze souboru. Hlavní informací je **délka hlavičky**. Čím větší délka, tím více informací hlavička i samotný soubor bude obsahovat. Je celkem pět možností délky hlavičky:

Velikost	Informace
5B	Základní obsah souboru
6E	Ikony pro Extra POIs
9C	Popisky pro Extra POIs
A4	Indexování Extra POIs
AE	Active Routing

Tabulka 3.1: Obsah hlavičky souboru podle délky

Kromě délky hlavička obsahuje informaci o tom, **kdy byl soubor vytvořen** – datum ve formátu rok, měsíc, den, hodiny, minuty, sekundy. Dále hlavička obsahuje informaci o **kódování textu (codepage)**. Tato informace je klíčová k provedení korektního čtení textu ze souboru bez ztráty

speciálních znaků specifických pro dané kódování. V hlavičce je dále údaj **family ID** a **product code**. Tyto dva údaje slouží ke správnému přiřazení TYP souboru k mapě. Následně se v hlavičce nachází informace o ukazateli na umístění bloku prvků POI, polygone a polyline a také délka tohoto bloku. Hlavička souboru dále obsahuje informace potřebné k načtení jednotlivých prvků (POI, Polygon, Polyline). Tyto informace jsou reprezentovány pomocí tří tabulek (pro každý prvek jedna). Tato jedna tabulka obsahuje tři údaje: posun, délku bloku tabulek B a délku jednoho záznamu v tabulce B. Díky těmto informacím je poté možné načíst další tabulky nutné pro načtení konkrétního prvku. Na každý prvek je jedna tabulka obsahující tyto údaje: typ prvku, podtyp prvku, posun prvku a rawType. Jedná se tedy o indexovací tabulky pro dané prvky. Tuto datovou strukturu popisuje následující schéma:



Obrázek 3.1: Datová struktura

Dále hlavička obsahuje informace pro načtení draworder úrovně pro každý polygon prvek. Tyto informace jsou reprezentovány obdobnou datovou strukturou jak v obrázku X (posun, délka bloku a délka jednoho záznamu). Nakonec se v hlavičce nachází informace potřebné k načtení extra POIs a Active routing.

3.2 Návrh a implementace knihovny pro práci s TYP souborem

Účelem této podkapitoly je seznámení čtenáře s knihovnou navrženou pro práci s TYP souborem. Bude zde rozebrána nejen postupná tvorba této knihovny a jejích částí, ale i tvorba souvisejících nástrojů pro čtení, zápis a zpracování binárních dat.

3.2.1 Tvorba vlastních nástrojů pro zpracovaní binárních souborů

TYP soubor je binární soubor. Je ho tedy nutné číst a zpracovat postupně po bytech nebo po blocích bytů. JavaScript k práci s daty po bytech poskytuje objekt s nízkoúrovňovým rozhraním zvaný **DataView**. Tento objekt poskytuje metody jak pro čtení, tak i pro zápis dat do objektu **ArrayBuffer**. Objekt ArrayBuffer se používá k reprezentaci obecných, nezpracovaných binárních dat s pevnou délkou. Jedná se tedy o pole bytů (bytové pole). Toto bytové pole je povinným parametrem pro vytvoření objektu DataView.

Jak již bylo zmíněno, DataView poskytuje metody pro čtení a zápis dat do objektu ArrayBuffer. Tyto metody se dělí podle toho, jak velký blok dat jsou schopny přečíst nebo zapsat: 8bitové,

16bitové, 32bitové a 64bitové. Dále se tyto metody dělí na dvě skupiny: signed (znaménkové) a unsigned (bezznaménkové).

Tyto metody umí číst/zapisovat data jak v pořadí **Big Endian**, tak i **Little Endian**. Endianita popisuje, v jakém pořadí jsou data organizována. Dělí se na Little Endian a Big Endian. Little Endian řadí byty od nejméně významného bytu (LSB), až po nejvíce významný byte (MSB). Zato Big Endian řadí byty v přesně opačném pořadí, tedy od nejvíce významného až po nejméně významný. Je tedy nutno zvolit správnou endianitu vůči zpracovávaným datům – metody DataView ve výchozím nastavení používají Big Endian, ale TYP soubor používa Little Endian.

Nedostatkem těchto metod je nutnost specifikace posunu, od kterého má daná metoda číst data. Nelze se tedy automatizovaně posouvat při čtení/zápisu dat ze/do ArrayBufferu opětovným voláním potřebné metody. Dále neexistuje žádná metoda pro načtení/zápis bloku dat o volitelné délce (například načtení řetězce o definované délce) nebo metoda pro načtení bloku dat s ukončovacím znakem. Na základě těchto nedostatků byla vytvořena **vlastní třída BinReaderWriter** pro čtení a zápis binárních dat. Pro vytvoření instance této třídy je nutné dosazení objektu DataView s daty do konstruktoru. Tato třída obsahuje celkem tři privátní proměnné:

Proměnná	Datový typ	Význam
buffLen	number	délka ArrayBufferu
position	number	pozice kurzoru v ArrayBufferu
buffer	DataView	objekt DataView obsahující ArrayBuffer s daty

Tabulka 3.2: Popis proměnných třídy BinReaderWritter

Dále tato třída obsahuje všechny základní metody pro čtení a zápis jako DataView, akorát již nevyžadují specifikaci posunu (offsetu), od kterého mají číst/zapisovat. S každým voláním metody pro čtení/zápis dochází automaticky k posunu kurzoru v souboru. Tyto metody také nyní pracují s daty pouze v pořadí Little Endian (TYP soubor používá Little Endian). Třída BinReaderWriter navíc implementuje následující metody:

Metoda	Návratová hodnota
seek(offsetOrigin: number)	void
readBytes(count: number)	Array<number>
writeBytes(bytes: Array<number>)	void
readString(len: number, codepage: number)	string
readStringWithUndefinedLen(maxLen: number = 0, codepage: number)	string
writeString(text: string)	void
read3U()	number

Tabulka 3.3: Vybrané metody třídy BinReaderWritter

Metoda **seek** slouží k nastavení kurzoru v ArrayBufferu na hodnotu předanou v parametru **offsetOrigin**. Data poté budou čtena/zapisována od této pozice. Metoda **readBytes** slouží ke čtení libovolně velkého bloku dat. Velikost bloku dat je předána přes parametr **count**, který určuje, kolik bytů bude přečteno. Přečtená data vrací jako pole bytů. Metoda **writeBytes** slouží k zápisu libovolně velkého bloku dat. Tato data jsou předána k zapsání jako pole bytů skrze parametr **bytes**. Jako další je metoda **readString**. Ta slouží k přečtení řetězce znaků o pevné délce z ArrayBufferu. Délka řetězce je předána přes parametr **len**. Načtený blok dat je poté převeden na znaky podle parametru **codePage**, který specifikuje, jaké konkrétní kódování aplikovat (například 1251 nebo 65001 (UTF-8)). Přečtený řetězec poté vrací jako string. Metoda **readStringWithUndefinedLen** slouží stejnému účelu jako metoda readString, pouze čte řetězec o předem neznámé délce. Data čte tak dlouho, dokud nenarazí na ukončovací znak nebo dokud nedojde na konec ArrayBufferu. Počet čtených znaků se dá omezit přes parametr **maxLen**. Obě metody, readString i readStringWithUndefinedLenght, využívají k převodu bytů na znaky JavaScript rozhraní zvané **TexDecoder**. Dále je metoda **writeString**, která slouží k zápisu řetězce znaků do ArrayBufferu. Jako vstup bere metoda řetězec znaků přes parametr **text**, který převede na pole bytů. Tyto byty následně uloží do Arrray-Bufferu. K převodu znaků na byty používá metoda writeString JavaScipt rozhraní **TextEncoder**. Jako poslední je metoda **read3U**. Ta umožňuje načtení bloku dat o velikost 24 bitů, načtenou hodnotu poté vrací jako number.

3.2.2 Knihovna pro zpracování TYP souboru

Knihovna pro práci s TYP souborem byla vytvořena na základě již existující implementace pro jazyk C#. Konkrétně se jedná o knihovnu zvanou **GarminCore**, která poskytuje nástroje pro čtení/zápis nejen samotného TYP souboru, ale i mnohých dalších Garmin souborů. Část této knihovny, konkrétně část zabývající se TYP souborem, byla přepsána z jazyku C# do jazyku JavaScript. Bylo nutno provést jisté modifikace knihovny, jelikož JavaScript neposkytuje stejné metody, nástroje nebo datové struktury jako C#. Například vytvoření své vlastní třídy pro reprezentaci **Bitmapy** nebo **Barvy**, které mají obdobné základní metody jako jejich ekvivalenty v jazyce C#.

Knihovna pro TYP soubor by se dala rozdělit celkem na tři části podle svých metod, a to na **čtení** souboru, **zápis** souboru a **editaci** dat souboru. Tyto tři části budou popsány v následujících podkapitolách.

Čtení souboru - celý proces načtení souboru začíná u hlavičky souboru. Jak již bylo zmíněno, hlavička obsahuje informace potřebné k načtení celého souboru. Je tedy nutno nejprve správně načíst data hlavičky souboru. Obsah hlavičky a jeho význam byly již popsány v předcházející kapitole zabývající se samotným TYP souborem, proto zde není nutno zacházet do detailního popisu, k čemu jednotlivé atributy hlavičky slouží.

Jakmile jsou načtena data hlavičky souboru, přichází na řadu načtení ikonek polyline. Z hlavičky souboru víme, kolik prvků polyline soubor obsahuje. Nyní je potřeba zjistit posun jednotlivých prvků

polyline a také jejich typ a subtyp. Tato data jsou reprezentována formou bloku N tabulek, kde N je počet prvků polyline. Informaci o začátku tohoto bloku tabulek obsahuje hlavička. Nastaví se tedy kurzor na danou adresu v ArrayBufferu a začnou se číst jednotlivé tabulky. Po dokončení načtení těchto tabulek je nyní možné číst jednotlivé prvky polyline. Adresu každého prvku polyline lze spočítat sečtením posunu polyline z hlavičky a posunu z patřičné tabulky, která byla právě načtena. Na tuto adresu se nastaví kurzor a dojde k načtení dané polyline. Tento celý proces je opakován, dokud nebudou načteny všechny polyline.

Proces samotného čtení jednoho prvku polyline vypadá následovně – nejprve se načte informace reprezentující jakési možnosti/vlastnosti konkrétní polyline. Z těchto vlastností lze zjitit, o jaký konkrétní typ polyline se jedná:

- **Day2** – polyline obsahující pouze denní ikonku o dvou barvách
- **Day2Night2** – polyline obsahující jak denní ikonku, tak i noční o dvou barvách
- **NoBorderDay2Night1** – polyline obsahující denní ikonku o dvou barvách a noční ikonku o jedné barvě, bez ohraničení
- **NoBorderDay1** – polyline obsahující jednobarevnou denní ikonku bez ohraničení
- **NoBorderDay1Night1** - polyline obsahující jednobarevnou denní i noční ikonku bez ohraničení

Dále z načtených vlastností lze zjistit **šířku bitmapy** ikonky. Je-li šířka nulová, jedná se o ikonku bez bitmapy. V případě bez-bitmapové ikonky bude později potřebné načíst šířku vnitřní čáry a případně podle typu polyline šířku ohraničení. Z vlastností lze také zjistit, jestli polyline obsahuje **rozšířené možnosti** nebo **pole popisků**. Jakmile jsou zpracovaná data načtena z vlastností polyline, přichází na řadu načtení **barev/barvy** ikonky. Počet barev k načtení je dán typem polyline. Po načtení barev je dále potřeba načíst informace o ikonce. Jedná-li se o ikonku bez bitmapy, dojde k načtení **tloušťky vnitřní čáry** a **tloušťky ohraničení**. V případě ikonky s bitmapou je potřeba načíst samotnou **bitmapu**. Ta je reprezentována pomocí formátu rastrové grafiky **X PixMap**. Poté dochází k načtení **popisků** polyline. Tyto popisky jsou reprezentovány polem prvků typu klíč – hodnota, kde klíč je kódové označení pro jazyk, ve kterém je popisek napsán, a hodnota je popisek samotný. Ke korektnímu načtení je potřeba data číst se správnou sadou znaků (codepage). Dále se načtou rozšířené možnosti, obsahuje-li je polyline. Rozšířené možnosti obsahují **typ fontu** a **barvu fontu** (denní, noční nebo obě). Tímto je proces načtení jedné polyline ukončen.

Po načtení všech prvků polyline dojde k načtení prvků **POI**. Proces načtení POI probíhá úplně stejným způsobem jako v případě polyline. Jediné, čím se liší, je načtení samotného jednoho POI. Je tedy nutné nejprve zjistit posuny všech prvků POI, včetně jejich typů a subtypů a spočítat výslednou adresu každého POI.

Proces čtení každého prvku POI funguje následovně – načtou se **vlastnosti** POI. Z těch se lze dozvědět, zdali obsahuje kromě **denní bitmapy** i **bitmapu noční**, dále jestli obsahuje popisky a rozšířené vlastnosti. Dále se načtou **rozměry** POI, tedy výška a šířka. Následně se načte **mód barev** POI. Módy barev jsou následující:

- **POI SIMPLE** – jednotná průhlednost, neomezený počet barev (teoreticky)
- **POI TR** – jednotná průhlednost, neomezený počet barev (teoreticky)
- **POI ALPHA** – každá barva může mít vlastní průhlednost, neomezený počet barev (teoreticky)
- **POLY2** – dvě barvy, jednotná průhlednost
- **POLY1TR** - jedna barva, jednotná průhlednost

Po načtení vlastností, rozměrů a barevného módu se přejde na načtení **denní bitmapy** POI. Poté pokud obsahuje dané POI noční bitmapu, dojde k načtení **noční bitmapy**. A nakonec se načtou **popisky** a **typ** a **barvy fontu**, disponuje-li POI těmito daty.

Posledním zbývajícím prvkem k načtení jsou **polygony**. Proces načítání polygonů probíhá stejně jako u polyline i u POI, jen je tentokráte potřeba načíst ještě dodatečné informace o **drawOrder úrovni** každého polygonu. Informace o těchto drawOrder úrovních jsou reprezentovány obdobnou datovou strukturou ve formě tabulek jako při načítaní POI, polyline nebo polygone prvků. Je tedy potřeba vzít posun drawOrder bloku z hlavičky a načíst jednotlivé tabulky obsahující informaci o drawOrder úrovni daného polygonu. Každá drawOrder tabulka také obsahuje informaci o **typu** a **subtypu** polygonu, ke kterému patří. Na základě tohoto typu a subTypu je poté tato draworder úroveň přiřazena každému polygonu s odpovídajícím typem a subTypem v rámci procesu načítání polygonů.

Proces načtení jednoho prvku polygon probíhá v následujících krocích – jak je již zvykem, nejprve se načtou informace o **vlastnostech** prvku polygone. Z vlastností lze identifikovat, o jaký typ polygonu se jedná:

- **Day1** – polygon s jednou denní barvou bez bitmapy
- **Day1Night1** – polygon s jednou denní a jednou noční barvou bez bitmapy
- **BMDay2** – polygon se dvěma denními barvami obsahující bitmapu
- **BMDay2Night2** – polygon se dvěma denními a dvěma nočními barvami obsahující bitmapu
- **BMDay1Night2** - polygon s jednou denní barvou a dvěma nočními barvami obsahující bitmapu

- **BMDay2Night1** – polygon se dvěma denními barvami a jednou noční barvou obsahující bitmapu
- **BMDay1** – polygon s jednou denní barvou obsahující bitmapu
- **BMDay1Night1** – polygon s jednou denní a jednou noční barvou obsahující bitmapu

Z vlastností lze dále zjistit informace, zdali polygon obsahuje **popisky** a **rozšířené nastavení**. Poté dojde k načtení **barev** polygonu. Počet načtených barev je dán typem polygonu. Jedná-li se o polygon s bitmapou, dojde po načtení barev k načtení **bitmapy**. A nakonec se načtou **popisky** a **typ** a **barvy fontu**, pokud polygon obsahuje tato data. Po dokončení procesu načtení jednoho celého polygonu je ještě nutno provést **přiřazení drawOrder úrovně** k polygonu. Prohledá se tedy list drawOrder tabulek a najde se patřičný záznam s odpovídajícím typem a subtypem.

Zápis souboru - proces zápisu dat souboru začíná vytvořením nového objektu ArrayBuffer, do kterého budou data zapisována. Tento ArrayBuffer je poté předán nástroji (objektu BinReaderWriter) pro zápis. Pozice kurzoru v ArrayBufferu se nastaví na adresu rovnou hodnotě délky (velikosti) hlavičky souboru (viz Tabulka 2.1: Obsah hlavičky souboru podle délky). Od této pozice bude zahájen zápis dat.

Jako první na řadu pro zápis přicházejí prvky typu **polygon**. Nejprve je za potřebí modifikovat záznam v hlavičce, který udává, kde se nachází blok dat polygonů (posun). Ten bude nastaven na hodnotu současné pozice kurzoru uvnitř ArrayBufferu. Až poté se přejde na zápis jednotlivých polygonů. Pro každý polygon je potřeba vykonat následující kroky – vytvoří se nová tabulka obsahující informace o typu, subtypu a posunu polygonu. Posun každého polygonu se spočítá jako rozdíl současné pozice kurzoru uvnitř ArrayBufferu a posunu celého bloku dat polygonů z hlavičky. Poté se přejde na zápis dat samotného polygonu. Tato jednotlivá data se zapíšou ve stejném pořadí, jako ve kterém byla načtena. Jediný rozdíl nastává u textu. Při načítání je na jednotlivé byty aplikováno kódování znaků podle patřičného formátu, který je specifikován v hlavičce souboru. Při zápisu se ale veškerý text uloží ve formátu kódování UTF-8. Tedy například pokud byl text souboru načten s kódováním Windows-1251 (1251), tak nyní bude korektně uložen ve formátu UTF-8 (65001). Informaci o změně formátu je taky nutno změnit v hlavičce souboru, která drží informaci o tom, jakou sadu znaků soubor používá. Jakmile se provedou tyto kroky pro každý polygon, je nutno změnit informaci v hlavičce souboru, jež udává celkovou délku bloku dat všech polygonů. Ta bude nastavena na hodnotu rozdílu současné pozice kurzoru uvnitř ArrayBufferu a posunu celého bloku dat polygonů z hlavičky. Poté je zapotřebí změnit v hlavičce údaj o posunu (umístění) bloku tabulek polygonů, k jejichž tvorbě a vyplnění daty došlo na úplném počátku procesu zápisu polygonů. Tento údaj bude nastaven na hodnotu současné pozice kurzoru uvnitř ArrayBufferu. Následně se provede zápis všech těchto tabulek od specifikované pozice. Po dokončení zápisu tabulek je potřeba provést poslední krok, a to nastavit informaci v hlavičce udávající délku bloku dat tabulek polygonů. Ta bude nastavena jako rozdíl současné pozice kurzoru uvnitř ArrayBufferu a posunu celého bloku tabulek z hlavičky. Tímto je proces zápisu polygonů ukončen.

Proces zápisu prvků typu **polyline** probíhá úplně stejným způsobem jako v případě polygonů. Pouze se zapisuje jiná skupina vlastností, jimiž prvek polyline disponuje. To stejné platí i pro zápis prvků typu **POI**.

Po zápisu polygonů, polyline a POI následuje zápis **drawOrder** úrovní polygonů. V rámci tohoto procesu je nutné projít všechny polygony, vzít jejich hodnoty drawOrder úrovní, typy a subtypy a utvořit z nich stejnou datovou strukturu, jako ve které byla načtena (tabulky). To také obnáší seřazení těchto drawOrder úrovní ve vzestupném pořadí. Před zahájením zápisu těchto dat je opět nezbytné nejprve modifikovat záznam v hlavičce udávající posun tabulek drawOrder úrovní. Tento posun bude nastaven na současnou pozici kurzoru uvnitř ArrayBufferu. Nyní jsou zapsána data o těchto drawOrder úrovních. Po dokončení zápisu drawOrder dat je ještě zapotřebí nastavit v hlavičce souboru délku bloku drawOrder dat. Ta bude nastavena na hodnotu rozdílu současné pozice kurzoru uvnitř ArrayBufferu a posunu tabulek drawOrder úrovní z hlavičky.

Jako poslední zbývá zápis samotné **hlavičky** souboru. Zápis hlavičky je nutno provést až jako poslední, jelikož hodnoty jejího aktuálního obsahu jsou známy až po dokončení zápisu polygonů, polyline a POI prvků a drawOrder úrovní. Zápis dat hlavičky začíná na úplném začátku ArrayBufferu, tedy na adrese (pozici) 0. Od této pozice jsou zapsána veškerá data hlavičky ve stejném pořadí, jako ve kterém byly načteny. Před zahájením samotného zápisu je ale ještě nutno provést modifikaci atributu codepage, jež udává formát kódování znaků. Ten bude nastaven na hodnotu 65001 (UTF-8), jak již bylo zmíněno v odstavci věnujícímu se zápisu prvků polygon.

Kapitola 4

Návrh a implementace webového editoru TYP souborů

Obsahem této kapitoly je popis implementace webové aplikace pro editaci Garmin TYP souborů. Budou zde popsány dílčí aspekty webové aplikace a rozebrána její funkcionalita z technického hlediska. Čtenář zde bude seznámen se strukturou aplikace, tedy s tím, jaké komponenty aplikaci tvoří a také jakou logiku tyto komponenty obsahují, jak fungují a jak mezi sebou komunikují.

4.1 Struktura webové aplikace

Aplikace je rozdělena do dílčích částí podle související logiky formou komponent pro přehlednější a snadnější vývoj. Následovat zde bude přehled zdrojové struktury této aplikace:

- **app** – kořenová komponenta, zapouzdřuje komponentu nav
  - **bitmap-canvas** – vykreslení ikony a aplikování grafických efektů na ikonu (scale, inverze barev...)
  - **editor** - editace prvků Polyline/POI/Polygone
    - **confirmation-dialog** – dialog oznamující uživatele o neuložených změnách
    - **icon-editor** – editace ikony Polyline/POI/Polygone
      - **resize** – změna velikosti prvků POI
      - **resize-polyline** – změna velikosti prvků Polyline
      - **select-color-filter** – výběr + ukázka barevných filtrů pro ikonu
      - **select-texture** – výběr + ukázka vygenerovaných vzorů/textur na ikoně
    - **icon-editor-description** – editace popisku Polyline/POI/Polygone
      - **icon-editor-description-form** – formulář pro nastavení popisku ve vybraném jazyce

    - **icon-editor-type** – editace typu a subtypu Polyline/POI/Polygone
- **file-download** - stažení aktuálního souboru do uložiště zařízení
- **file-new** - vygenerování nového, prázdného TYP souboru
    - **file-new-dialog** – formulář pro vytvoření prázdného TYP souboru
- **file-upload** - nahrání nového TYP souboru do aplikce
- **grid-select** - zobrazení Polyline/POI/Polygone prvků pomocí grid layoutu a jejich výběr (označení) po kliknutí
- **icon-descriptions** - tabulka obsahující všechny prvky a jejich popisky, slouží k editaci popisků
    - **icon-description** – formulář pro editaci/přidání popisku
- **main-page** - dashboard informující o vlastnostech souboru
- **nav** - navigační panel pro přesun mezi částmi aplikace
- **options** - nastavení pro celý soubor (FID, PID, limitace barevné palety všech ikon)
- **poi** - grid layout zobrazující všechny POI prvky, editace, přidání a odstranění POI prvků
    - **add-poi** – vytvoření nového POI prvku
- **polygone** - grid layout zobrazující všechny polygon prvky, editace, přidání a odstranění polygon prvků
    - **add-polygone** – vytvoření nového polygon prvku
- **polygone-drawroder-sort** - změna drawOrder úrovně polygonů pomocí drag&drop
- **polyline** - grid layout zobrazující všechny polyline prvky, editace, přidání a odstranění polyline prvků
    - **add-polyline** – vytvoření nového polyline prvku
- **services** - služby (objekty) poskytující metody a data po celou dobu „života“ aplikace
    - **file.service.ts** – poskytování dat a metod týkajících se TYP souboru
    - **dialog.service.ts** - metody pro confirmation-dialog komponentu
- **guards** - vykonání dané logiky před nebo po opuštění současné routy

4.2 Detailní popis komponent webové aplikace

Tato podkapitola bude navazovat na předchozí podkapitolu, kde byl uživatel seznámen se základní strukturou aplikace. Zde bude tato struktura detailně rozebrána postupně komponenta po komponentě. U každé komponenty budou popsány její vstupy/výstupy, k čemu slouží, jak komunikuje s ostatními komponentami a hlavně jak funguje její logika.

4.2.1 Komponenta „Nahrání nového souboru“ (file-upload)

Komponenta file-upload je tlačítko, které slouží k nahrání nového TYP souboru do aplikace. Po kliknutí na toto tlačítko dojde k otevření lokálního průzkumníka souborů v zařízení. Po zvolení souboru a jeho nahrání dojde ke kontrole jeho přípony. Pokud se jedná o TYP soubor, budou jeho data přečtena pomocí JavaScriptového nástroje **FileReader**. Objekt FileReader umožňuje webovým aplikacím asynchronně číst obsah souborů pomocí objektů File nebo Blob. Data souboru budou přečtena metodou FileReader.readAsArrayBuffer. Výsledkem čtení jsou tedy čistá binární data ve formátu ArrayBuffer (pole bytů). Toto pole bytů je použito pro vytvoření objektu DataView, který je následně předán do konstruktoru třídy reprezentující TYP soubor (TypFile.ts). Jakmile je vytvořen objekt TypFile, zavolá se metoda **setFile** ze servisní třídy **file.service.ts**, které je tento objekt předán. Kromě TypFile objektu je této metodě předán objekt ArrayBuffer a jméno souboru. Metoda setFile se postará o to, aby se do uložiště **Local Storage** uložil serializovaný objekt ArrayBuffer. Pro serializaci je použito kódování **Base64**, které převádí binární data na posloupnosti tisknutelných znaků. Toto kódovaní je nutno provést, jelikož uložiště Local Storage podporuje pouze ukládání dát ve formátu párů klíč-hodnota, kde klíč i hodnota musí být datového typu string. Metoda setFile dále uloží do Local Storage jméno souboru a velikost obsahu souboru v bytech, tedy délku objektu ArrayBuffer. Díky těmto datům je možné uchovat v aplikaci nahraný soubor i po zavření okna prohlížeče. Po úspěšném nahrání dat je upozorněna komponenta **main-page** o změně aktuálních dat skrze **BehaviorSubject** z knihovny RxJS. Ta si jako reakci na toto upozornění aktualizuje veškerá svá potřebná data. Nakonec je uživatel přesměrován na hlavní stránku aplikace, tedy do komponenty main-page.

4.2.2 Komponenta „Vytvoření prázdného souboru“ (file-new)

Komponenta file-new je tlačítko, které slouží k odstartování procesu vytvoření nového TYP souboru. Po kliknutí na toto tlačítko se otevře dialog obsažený v komponentě **file-new-dialog**.

4.2.3 Komponenta „Dialog pro tvorbu prázdného souboru“ (file-new-dialog)

Komponenta file-new-dialog je dialog obsahující formulář s údaji potřebnými pro vytvoření nového TYP souboru. Dialog je komponenta **mat-dialog** z knihovny Angular Material, která reprezentuje modální dialog.

Uvnitř mat-dialog-content se nachází formulář, jehož vyplněný obsah je potřebný k vytvoření nového, prázdného TYP souboru. Formulář je vytvořen skrze nástroj zvaný **FormBuilder**. FormBuilder je pomocné API pro vytváření reaktivních formulářů v Angularu. Poskytuje zkratky pro vytvoření instancí objektů FormControl, FormGroup nebo FormArray, čímž redukuje množství kódu potřebného k psaní složitějších formulářů. Umožňuje také nastavit validační pravidla pro ověření každého z ovládacích prvků formuláře.

K vytvoření skupiny formulářových polí je použita metoda **group**. Té je předán seznam ovládacích prvků formuláře ve formě párů klíč–hodnota, kde klíč je název prvku/pole formuláře a hodnota je konfigurace vlastností tohoto pole. Vlastnosti daného pole jsou validátory. Jejich účelem je provést kontrolu obsahu daného formulářového pole. **Validators.required** říká, že pole nesmí být prázdné.

```
this.descriptionForm = this.formBuilder.group({
     fileName: ['', [Validators.required]],
     pid: [null, [Validators.required]],
     fid: [null, [Validators.required]]
   });
```

Výpis kódu 4.1: Ukázka FormBuilder nástroje

Po odeslání formuláře dojde k validaci dat v jednotlivých polích. Tato validace je provedena na základě specifikovaných validátorů při tvorbě formuláře. Pokud jsou veškeré podmínky specifikované validátory splněny, bude zavolána metoda **createFile** ze servisní třídy file.service.ts. Této metodě budou předána data z formuláře a ta na základě těchto dat vytvoří nový objekt TypFile a uloží jeho data do LocalStorage. Poté bude upozorněna komponenta main-page o změně aktuálních dat skrze **BehaviorSubject** a uživatel bude přesměrován do komponenty main-page. V opačném případě bude uživatel upozorněn na nesprávné vyplnění formuláře skrze alert.

4.2.4 Komponenta „Stažení nahraného souboru“ (file-download)

Komponenta file-download je tlačítko pro stažení aktuálně nahraného TYP souboru do zařízení uživatele. Ke stažení souboru je zde využit nástroj **file-saver** (FileSaver.js). FileSaver umožňuje ukládání souborů na straně klienta, je tedy vhodný pro webové aplikace, které potřebují generovat soubory na straně klienta. Ke stažení souboru slouží metoda **FileSaver.saveAs**. Skrze vstupní parametry jsou metodě předána data souboru ve formátu **Blob** (skupina bytů, která obsahuje data uložená v souboru) a řetězec reprezentující jméno staženého souboru s příponou. Pro získání objektu Blob je použita metoda **getBlob** ze servisní třídy **file.service.ts**. Tato metoda se postará o zakódování a zápis dat právě načteného souboru do objektu ArrayBuffer, ze kterého je možno vytvořit objekt Blob.

4.2.5 Komponenta „Vykreslení bitmapy“ (bitmap-canvas)

Komponenta bitmap-canvas je komponenta, jejímž účelem je vykreslování ikon prvků POI, polyline a polygone do canvasu s možností aplikace různých definovaných barevných efektů (inverze barev, černobílý efekt...), zvětšení nebo vykreslení vzorů přes ikonu (šachovnice, diagonály...). Tato komponenta má celkem pět vstupů:

- **drawableItem: GraphicElement** – rodičovská třída prvků POI, polyline, polygone, která je potřebna pro získání bitmapy prvku k vykreslení

- **scaleValue: number** – numerická hodnota udávající, kolikrát bude zvětšena ikona prvku k vykreslení
- **darken: boolean** – boolean proměná, hodnota true – ikona bude zatmavena
- **effect: Effects** – vlatní enum specifikující efekt, jež se má aplikovat na ikonu prvku
- **effectColor: string** – hexa kód barvy efektu

Jak již bylo zmíněno, tato komponenta umožňuje různou manipulaci s ikonou prvku k vykreslení. Jedna z těchto úprav je **zvětšení ikony** nkrát, kde n je předáno jako vstup komponenty. Zvětšení je aplikováno na ikony ve všech částech aplikace, kde jsou ikony prvků zobrazeny, jelikož jejich původní velikost je velmi malá (nejčastěji pár desítek pixelů na výšku/šířku). Ikona je do HTML canvasu vykreslena pomocí čtverců, kde každý čtverec reprezentuje jeden pixel ikony. Velikost strany každého čtverce je dána proměnnou scaleValue. Je-li tedy hodnota scaleValue rovna dvaceti, tak velikost jedné strany čtverce bude 20px a ikona tak bude zvětšena dvacetkrát.

Komponenta dále umí aplikovat různé barevné efekty na ikonu. Prvním z nich je **ztmavení ikony**. O to se stará metoda **darkenCanvas**, která docílí efektu ztmavení vykreslením poloprůhledného tmavého obdélníku přes obsah canvasu. Tento efekt je v rámci aplikace využit pro označení vybrané ikony.

Krom efektu ztmavení umí tato komponenta provést **inverzi barev ikony**. To má na starost metoda **applyInverseColors**, která před vykreslením bitmapy provede inverzi barev všech pixelů. Pro tuto operaci je vytvořena metoda přímo ve třídě Bitmap – metoda **Bitmap.inverseColors**. Tato metoda postupně prochází bitmapu a invertuje barvy jednotlivým pixelům. V průběhu průchodu si algoritmus pro každou původní barvu ukládá její vypočtenou inverzi. Pokud se tedy narazí na pixel s barvou, pro kterou již byla vypočtena inverze, použije se již tato uložená hodnota barvy a není nutné opakovaně počítat inverzi.

```
getInverseColor(currentColor: Color): Color {
    const r: number = 255 - currentColor.r;
    const g: number = 255 - currentColor.g;
    const b: number = 255 - currentColor.b;

    return new Color(r,g,b);
}
```

Výpis kódu 4.2: Inverze barev ikony

Dalším barevným efektem, kterým komponenta disponuje, je **vykreslení ikony v černobílém formátu**. Za touto funkcionalitou stojí metoda **applyGrayScale**, jež před vykreslením provede převod barev bitmapy do černobílého rozsahu. K tomuto převodu slouží metoda třídy Bitmap –

metoda **Bitmap.grayScaleFilter**. Algoritmus převodu jednotlivých barev bitmapy zde funguje na stejném principu jako v případě inverze, kde se ukládají již vypočtené barvy, aby nedocházelo k opakovanému výpočtu pro identické barvy. Pouze s tím rozdílem, že se zde převádí barva do rozsahu černobílé škály místo inverze barvy.

```
Math.floor(0.2126 * currentColor.r + 0.7152 * currentColor.g + 0.0722 *
    currentColor.b)
```

Výpis kódu 4.3: Převod barvy do černobílého rozsahu

Mimo barevné efekty umí tato komponenta také generovat různé vzory/textury na vykreslenou ikonu. Tyto vzory jsou:

- šachovnice
- horizontálně orientované čáry
- vertikálně orientované čáry
- diagonální čáry – zleva i zprava
- diamantový vzor – kosočtverce

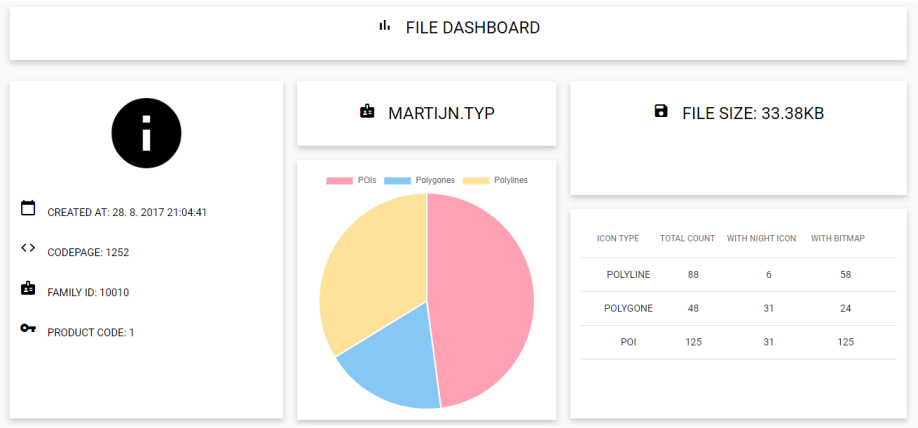
Tyto vzory jsou nejčastěji používány pro prvky polygone, které v mapách reprezentují velké plochy.

4.2.6 Komponenta „Dashboard TYP souboru“ (main-page)

Komponenta main-page představuje dashboard aplikace. Slouží k zobrazení různých statistik a dat nahraného souboru do aplikace. Zobrazuje jméno nahraného souboru, datum a čas, kdy byl soubor vytvořen, velikost souboru v kilobytech, typ kódování textů v souboru, family ID a product code. Dále zobrazuje koláčový graf, který graficky znázorňuje obsah prvků POI, polygone, polyline v souboru, tedy jejich celkový počet. Pro zobrazení tohoto grafu využívá již zmíněnou knihovnu **ng2-charts**, ze které používá konkrétně graf pieChart. Kromě grafu komponenta zobrazuje tabulku se statistikami pro jednotlivé typy prvků, tedy POI, polygone a polyline. Pro každý z těchto prvků obsahuje informace o celkovém počtu těchto prvků, kolik z nich má kromě denní ikony i ikonu noční a kolik z nich má ikonu reprezentovanou bitmapou. Tato tabulka je reprezentována komponentou **mat-table** z knihovny Angular Material. Všechny tyto zmíněné prvky a komponenty jsou rozmístěny v rámci komponenty main-page pomocí knihovny Flex Layout.

4.2.7 Komponenta „Nastavení TYP souboru“ (options)

Komponenta options představuje globální nastavení pro celý soubor. Umožňuje nastavit **family ID** a **product code** souboru. Krom toho také umí **limitovat barevnou paletu** všech ikon prvků v



Obrázek 4.1: Komponenta main-page

souboru. Toto nastavení pro limitaci spektra barev existuje z důvodu optimalizace vzhledu ikon na různých Garmin zařízeních, jako jsou například jejich chytré hodinky. Spousta přenosných zařízení Garmin je vybavena displejem, který umožňuje zobrazit pouze limitovaný počet barev – pouze 16/64/256 barev. Může tedy nastat, že ikona bude obsahovat barvy, které dané zařízení není schopno zobrazit, a tím pádem bude daná ikona nečitelná, což může značně ovlivnit funkcionalitu mapy. Proto v nastavení aplikace existuje tato možnost převodu barev ikon do barev z vybrané palety (16, 64, 256). O tuto funkcionalitu se stará metoda **limitColorPallete** ze servisní třídy file.service.ts. Té je skrze vstupní parameter předán typ palety (enum), pro kterou se mají bravy upravit. Postupně se projdou všechny polygony, POI a polyline prvky a každému prvku se přepočítají barvy na barvy z palety. Tento přepočet funguje tak, že se pro každou barvu v ikoně prochází list barev palety a hledá se barva s nejmenší vzdáleností vůči původní barvě.

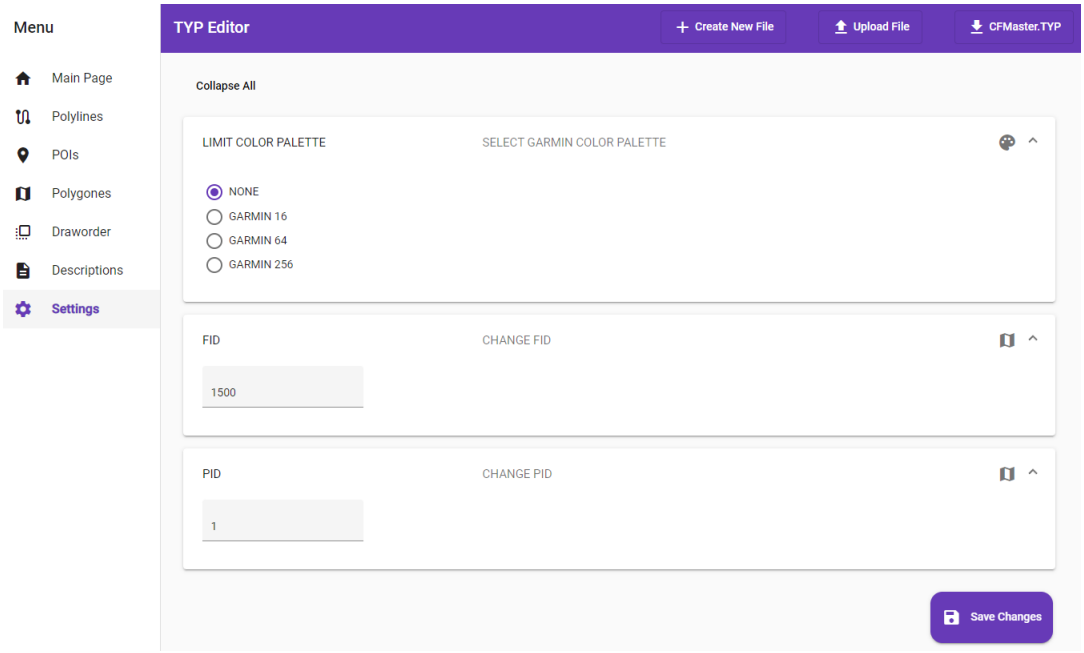
```
getDistance(currentColor: Color, matchColor: Color): number {

    const redDifference: number = currentColor.r - matchColor.r;
    const greenDifference: number = currentColor.g - matchColor.g;
    const blueDifference: number = currentColor.b - matchColor.b;
    return redDifference * redDifference +
           greenDifference * greenDifference +
           blueDifference * blueDifference;
}
```

Výpis kódu 4.4: Výpočet vzdálenosti dvou barev

Pro reprezentaci nastavení byla zvolena komponenta **mat-accordion** z knihovny Angular Material, která poskytuje možnost skrýt/odkrýt části svého obsahu. Uživatel si tak může přehledně zobrazit jen ten obsah, který ho zajímá. Po uložení nových nastavení budou data ze vstupů valido-

vána pomocí FormGroup validátorů a poté v případě, že jsou validní, budou změny aplikovány na soubor a uživatel bude přesměrován na hlavní stránku aplikace.



Obrázek 4.2: Komponenta options

4.2.8 Komponenta „Přehled prvků TYP souboru“ (polyline, POI, polygone)

Tato komponenta slouží k zobrazení všech prvků z dané kategorie, jež jsou obsaženy v aktuálně nahraném souboru. K tomu je použita komponenta **mat-grid-list** z knihovny Angular Material. Tato komponenta slouží k zobrazení responzivního dvourozměrného seznamu, který uspořádává jednotlivé buňky do rozvržení založeného na mřížce. Pro nastavení vlastností této mřížky se využívá její API.

K manipulaci s touto mřížkou nabízí tato komponenta posuvník (slider). Pro posuvník byla použita komponenta **mat-slider** z knihovny Angular Material, která umožňuje výběr hodnoty z definovaného rozsahu. Tento posuvník slouží ke zvětšení/zmenšení dlaždic mřížky. Zvětšení obnáší postupné snižování počtu sloupců mřížky a zvětšování ikony prvků. Naopak při zmenšení dochází k nárůstu počtu sloupců mřížky a zmenšování ikon. O tuto popsanou funkcionalitu zmenšení/zvětšení se stará metoda **updateGrid**, která je volána s každou změnou hodnoty posuvníku. Ta na základě hodnoty posuvníku vypočítává nový počet sloupců mřížky a také kolikrát bude zvětšena ikona prvku.

Každý prvek je reprezentován jako dlaždice v mřížce. Po kliknutí na dlaždici dojde k přesměrování do komponenty editoru, kde lze daný prvek editovat. Uvnitř každé dlaždice se nachází vlastní komponenta **bitmap-canvas**. Pomocí ní je zobrazena ikona prvku, která je zvětšena nkrát použitím

Nearest Neighbor algoritmu. Každá dlaždice obsahuje také patičku. V ní se nachází popisek daného prvku v jednom jednotném jazyce a také komponenta **mat-menu** z knihovny Angular Material. Mat-menu je komponenta reprezentující plovoucí panel obsahující seznam možností. Tento seznam obsahuje možnost odstranění prvku a možnost přesunu přímo do konkrétní části editoru – editace denní ikony/noční ikony/popisku/typu polyline.

K **odstranění** prvků tato komponenta nabízí také separátní tlačítko, které přepne tuto komponentu do režimu odstranění, kde po kliknutí na dlaždici mřížky dojde k přidání daného prvku do „koše“. Kliknutí na dlaždici způsobí její označení formou červeného ohraničení a ztmavením ikony prvku. K odstranění vybraných prvků dojde po kliknutí na potvrzovací tlačítko, které také informuje o počtu prvků k odstranění. Výběr označených dlaždic k odstranění lze vyresetovat pomocí tlačítka cancel nebo postupným klikáním na již označené dlaždice. Po potvrzení o odstranění se komponenta automaticky přepne do normálního režimu. O samotné odstranění se stará servisní třída **file.service.ts**, konkrétně její metoda **deleteItems**. Té je skrze vstupní parametry předána informace o tom, jestli se jedná o prvek typu POI/polygone/polyline a také list vybraných prvků k odstranění. Podle typu a subtypu těchto prvků v listu dojde k vyfiltrování všech prvků ze souboru, které mají odlišný typ a zároveň i subtyp. Poté dojde k aktualizaci dat souboru, která jsou uložena v uložišti **Local Storage**.

Komponenta dále nabízí možnost přidání nových prvků do souboru. Toho lze docílit dvěma způsoby, a to vytvořením nového prvku nebo přidáním prvků z jiného TYP souboru. Tyto možnosti jsou zapouzdřeny pomocí již zmíněné komponenty mat-menu. V případě zvolení možnosti vytvoření nového prvku se otevře komponeta **add-polyline/add-polygone/add-poi** , která obsahuje dialog s formulářem, po jehož vyplnění, dojde k vytvoření nového prvku, aktualizaci dat v Local Storage a přesměrování do komponenty editoru, kde bude nově vytvořený prvek. V případě zvolení druhé možnosti, tedy možnosti přidání prvků z jiného TYP souboru, dojde k otevření průzkumníka souboru, pomocí kterého lze vybrat TYP soubor, ze kterého budou spojeny vybrané prvky. Po výběru se soubor zpracuje stejným způsobem jako v komponentě **file-upload** a otevře se dialog obsahující komponentu **grid-select**, která slouží k výběru prvků typu POI/polyline/polygone. Po označení prvků ke spojení je nutno tento výběr ještě potvrdit. Nově spojené prvky poté budou uloženy do aktuálního souboru a dojde k aktualizaci dat uvnitř Local Storage.

Obrázek 4.3: Komponenta polygone

4.2.9 Komponenta „Editor prvků TYP souboru“ (editor)

Komponenta editor slouží k editaci jednotlivých částí prvků TYP souboru, tedy prvků POI, polygone a polyline. Samotný editor je rozdělen do čtyř částí na základě vlastností, jimiž tyto prvky disponují. Tyto části jsou: **grafický editor pro denní ikonu a pro noční ikonu, editor popisků prvku a editor typu prvku**. Pro přechod mezi jednotlivými částmi editoru slouží komponenta mat-tab z knihovny Angular Material. Ta umožňuje organizovat obsah do samostatných záložek, kde může být v jeden okamžik viděna pouze jedna zvolená záložka.

První a hlavní částí editoru je **grafický editor** ikonek prvků TYP souboru. Jeho účelem je editace bitmapy. Uprostřed editoru se nachází HTML Canvas, který slouží pro zobrazení bitmapy. Tato bitmapa je vykreslena do sítě čtverců o velikosti strany 20px, kde každý čtverec reprezentuje jeden pixel bitmapy. Bitmapa je tedy takto zvětšena 20krát. K samotné editaci bitmapy editor poskytuje následující nástroje:

Nástroj „Štětec“ slouží pro vykreslení souvislé čáry do sítě buněk (čtverců) v Canvasu. Tah čáry je zahájen stiskem levého tlačítka myši a následným posunem myši je čára postupně vykreslována, dokud nedojde k uvolnění stištěného levého tlačítka. K zachytávání a zpracování pohybu myši je využit operátor z knihovny **RxJS fromEvent()**, který vrací prvek jako observable, jenž vysílá události specifického typu, jako jsou například kliknutí. Pro vykreslení samotné čáry je využita **lineární interpolace** mezi dvěma body, kde první bod je předešlá pozice kurzoru a druhý bod je současná pozice kurzoru. Algoritmus postupně prochází po čáře od poslední pozice myši do aktuální pozice pohybu myši. Poté vybarví buňky sítě (čtverce), které mine po této cestě. U tohoto algoritmu je také nutno rozhodnout, kolik bodů na přímce zahrnout, tedy jakým krokem bude algoritmus postupovat. Bude-li jich příliš málo, čára bude obsahovat mezery. Bude-li jich příliš mnoho, čára se přetáhne. Dále je při průchodu čáry vhodné kontrolovat, jestli tento algoritmus našel novou buňku k vybarvení, jelikož mnoho iterací nenajde novou buňku a došlo by tak k duplicitnímu vybarvování již navštívených buněk. K samotnému vybarvení dané buňky je zapotřebí identifikovat, které buňce v síti daný bod na přímce odpovídá. To lze zjistit tímto výpočtem: x = x – (x mod šířka buňky); y = y – (y mod výška buňky).

```
private interpolateLine(prevPos: { x: number; y: number }, currentPos: { x: number
    ; y: number }): void {
    for (let pct = 0; pct <= 1; pct += 0.03) {
      const dx = currentPos.x - prevPos.x;
      const dy = currentPos.y - prevPos.y;
      const X = prevPos.x + dx * pct;
      const Y = prevPos.y + dy * pct;
      if (positionChange({x: X, y:Y})) {
          drawColorCell(X, Y);
      }
    }
  }
```

Výpis kódu 4.5: Interpolace přímky

Nástroj „Přímka“ slouží k vykreslení přímky do sítě buněk v Canvasu. Počáteční bod přímky je vytvořen stisknutím levého tlačítka myši a přímka je vykreslována následným pohybem myši od tohoto bodu, dokud nedojde k uvolnění stisknutého levého tlačítka. Tento nástroj funguje obdobně jako nástroj „Štětec“ s použitím lineární interpolace mezi dvěma body, pouze s tím rozdílem, že se interpolace pokaždé provádí od původního počátečního bodu po současnou pozici, a ne od poslední pozice jako v případě čáry.

Nástroj „Obdélník“ slouží pro vykreslení obdélníku bez výplně do sítě buněk v Canvasu. Jako ostatní nástroje funguje na stejné mechanice stisku levého tlačítka a potažení myší. K vykreslení se zde taktéž využívá lineární interpolace mezi dvěma body. Ta se provádí zvlášť pro každou stranu obdélníku:

- strana a = interpolace mezi body A x: počáteční pozice X, y: počáteční pozice Y a B x: současná pozice X, y: počáteční pozice Y
- strana b = interpolace mezi body A x: počáteční pozice X, y: počáteční pozice Y a B x: počáteční pozice X, y: současná pozice Y
- strana c = interpolace mezi body A x: současná pozice X, y: počáteční pozice Y a B x: počáteční pozice X, y: počáteční pozice Y
- strana d = interpolace mezi body A x: počáteční pozice X, y: současná pozice Y a B x: současná pozice X, y: počáteční pozice Y

Nástroj „Obdélník s výplní“ slouží pro vykreslení obdélníku s výplní do sítě buněk v Canvasu. Taktéž jako ostatní nástroje funguje na stejné mechanice stisku levého tlačítka a potažení

myší. K vykreslení obdélníku s výplní je nutno nadefinovat následující hranice, a to odkud (počáteční souřadnice) a kam (současná souřadnice):

- Horní hranice = počáteční souřadnice Y
- Spodní hranice = současná souřadnice Y
- Levá hranice = počáteční souřadnice X
- Pravá hranice = současná souřadnice X

Pokud je současná pozice kurzoru po ose X menší než počáteční pozice, je nutné redefinovat levou a pravou hranici následovně:

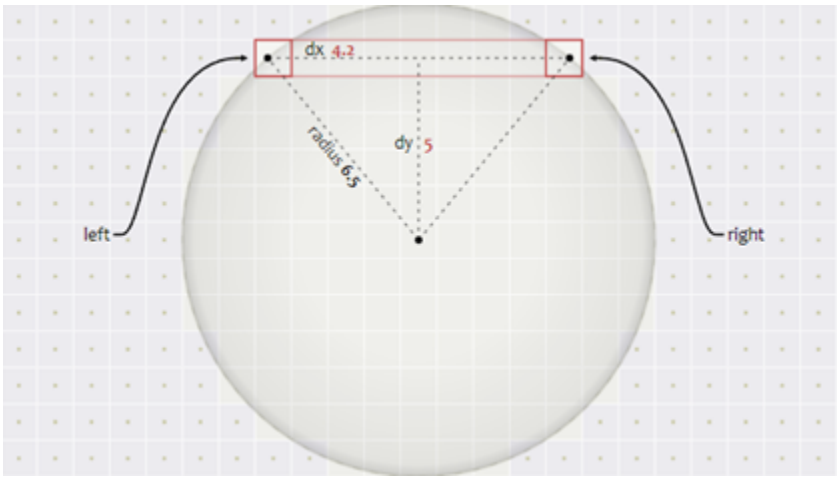
- Levá hranice = současná souřadnice X
- Pravá hranice = počáteční souřadnice X

To stejné platí i pro horní a spodní hranici, pokud je současná pozice kurzoru po ose Y menší než počáteční pozice:

- Horní hranice = současná souřadnice Y
- Spodní hranice = počáteční souřadnice Y

Jakmile jsou takto nadefinovány hranice pro vybarvení obdélníku, je potřeba projít tento obdélník a obarvit jemu náležející buňky v síti.

Nástroj „Kruh s výplní“ slouží pro vykreslení plného kruhu do sítě buněk v Canvasu. Taktéž jako ostatní nástroje funguje na stejné mechanice stisku levého tlačítka a potažení myší. K vykreslení kruhu je zapotřebí nadefinovat ohraničující kruh. Každý řádek tohoto kruhu bude mít rozdílnou šířku. Tu lze vypočítat pomocí Pythagorovy věty. Vznikne zde pravoúhlý trojúhelník s poloměrem kruhu jako přeponou a řádkem jako jednou stranou trojúhelníku. Druhá strana trojúhelníku udává šířku řádku. Pro každý řádek lze takto nalézt levou a pravou buňku v síti pomocí Pythagorovy věty. Poté lze vyplnit všechny buňky v tomto rozsahu mez levou a pravou buňkou.



Obrázek 4.4: Kruhová výplň s ohraničující kružnicí [3]

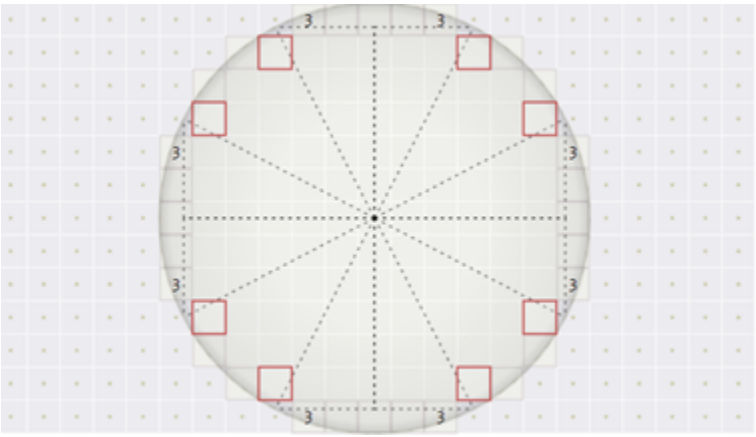
```
drawFilledCircle(prevPos: { x: number; y: number }, currentPos: { x: number; y:
    number }): void {
    const centerCoord = convertCoord(prevPos);
    const curCoord = convertCoord(currentPos);
    const radius = Math.sqrt((curCoord.x - centerCoord.x) * (curCoord.x -
        centerCoord.x) +
                          (curCoord.y - centerCoord.y) * (curCoord.y - centerCoord
                              .y));
    const top = Math.floor(centerCoord.y - radius), bottom = Math.ceil(centerCoord
        .y + radius);

    for (let y = top; y <= bottom; y++) {
      const dy = y - centerCoord.y;
      const dx = Math.sqrt(radius*radius - dy*dy);
      const left = Math.ceil(centerCoord.x - dx), right = Math.floor(centerCoord.x
          + dx);
      for (let x = left; x <= right; x++) {
        drawColorCell2(x * scale, y * scale);
      }
    }
  }
```

Výpis kódu 4.6: Vykreslení kruhu s výplní

Nástroj „Kruh“ slouží pro vykreslení kružnice do sítě buněk v Canvasu. Taktéž jako ostatní nástroje funguje na stejné mechanice stisku levého tlačítka a potažení myší. K vykreslení kružnice

zde nelze využít pouze algoritmu pro výpočet ohraničujícího kruhu jako v případě kruhu s výplní a nechat vybarvené pouze hraniční buňky, jelikož tento postup by zanechal mezery a kružnice by tedy nebyla souvislá. Je zde tudíž zapotřebí poupravit logiku kódu. Bude se zde iterovat přes délku kratší strany pravoúhlého trojúhelníku vzniklého uvnitř kružnice (viz obrázek X). Z této délky lze poté spočítat délku delší strany pravoúhlého trojúhelníku. S těmito délkami lze nyní najednou vyplnit osm buněk.



Obrázek 4.5: Obrys kruhu [3]

```
drawCircle(prevPos: { x: number; y: number }, currentPos: { x: number; y: number
    }): void {
    const centerCoord = convertCoordinates(prevPos);
    const curCoord = convertCoordinates(currentPos);

    let radius = Math.sqrt((curCoord.x - centerCoord.x) * (curCoord.x -
        centerCoord.x) +
                           (curCoord.y - centerCoord.y) * (curCoord.y - centerCoord
                               .y)
    );

    for (let r = 0; r <= Math.floor(radius * Math.sqrt(0.5)); r++) {
      const d = Math.floor(Math.sqrt(radius*radius - r*r));
      drawColorCell((centerCoord.x - d) * scale, (centerCoord.y + r) * scale);
      drawColorCell((centerCoord.x + d) * scale, (centerCoord.y + r) * scale);
      drawColorCell((centerCoord.x - d) * scale, (centerCoord.y - r) * scale);
      drawColorCell((centerCoord.x + d) * scale, (centerCoord.y - r) * scale);
      drawColorCell((centerCoord.x + r) * scale, (centerCoord.y - d) * scale);
      drawColorCell((centerCoord.x + r) * scale, (centerCoord.y + d) * scale);
```

```
      drawColorCell((centerCoord.x - r) * scale, (centerCoord.y - d) * scale);
      drawColorCell((centerCoord.x - r) * scale, (centerCoord.y + d) * scale);
    }
  }
```

Výpis kódu 4.7: Vykreslení kruhu bez výplně

Nástroj „otočení horizontálně“ a „otočení vertikálně“ slouží k otočení bitmapy po ose x a po ose y. Na začátku se vytvoří nová bitmapa o stejných rozměrech jako bitmapa původní. Poté se projde původní bitmapa a její jednotlivé pixely se umístí do nové bitmapy na nově vypočtené souřadnice. V případě horizontálního otočení zůstává pixelu původní souřadnice na ose y a souřadnice na ose x se vypočte následovně:

x = šířka bitmapy – souřadnice pixelu na ose x–1

A v případě vertikálního otočení zůstává pixelu původní souřadnice na ose x a souřadnice na ose z se vypočte následovně:

y = výška bitmapy – souřadnice pixelu na ose y–1

Nástroj „rotace doleva“ slouží k rotaci bitmapy o 90 stupňů proti směru hodinových ručiček. Na začátku se vytvoří nová bitmapa o stejných rozměrech jako bitmapa původní. Poté se projde původní bitmapa a její jednotlivé pixely se umístí do nové bitmapy na nově vypočtené souřadnice. Souřadnice na ose y se pixelu nastaví na hodnotu souřadnice na ose x a souřadnice na ose x se změní na základě následujícího výpočtu:

x = výška bitmapy – souřadnice pixelu na ose y–1

Nástroj „posun bitmapy“ slouží k „uchycení“ bitmapy levým tlačítkem a následnému posunu požadovaným směrem tažením myši uvnitř sítě buněk v canvasu. Po uvolnění levého tlačítka bude bitmapa vykreslena na nových souřadnicích. Je zapotřebí vypočítat nové počáteční souřadnice, od kterých bude bitmapa po posunu vykreslena. Ty lze zjistit výpočtem rozdílu současných souřadnic a původních souřadnic kurzoru. Tento rozdíl představuje posun, se kterým bude nově bitmapa vykreslena. Nové počáteční souřadnice bitmapy tedy budou rovny součtu původních hodnot se získaným posunem.

Nástroj „Guma“ umožňuje postupně čistit/umazávat části bitmapy. Tento nástroj funguje obdobně jako ostatní nástroje – k mazání začne docházet po stisku levého tlačítka myší a následným tažením se začnou „čistit“ pixely bitmapy (nastavovat na bílou barvu).

Nástroj „Výplň plochy“ slouží k vyplnění zvolené plochy vybranou barvou. K vyplnění dojde po kliknutí do oblasti požadované plochy. K realizaci tohoto nástroje byl zvolen Flood fill algoritmus. Pro implementaci tohoto algoritmu byl zvolen rekurzivní přístup, který nejprve nahradí barvu na současném pixelu a poté rekurzivně nahradí barvu na okolních pixelech.

```
fillUtil(x: number, y: number, newCol: Color, prevCol: Color): void {
    if (x < 0 || x >= width || y < 0 || y >= height) return;
    if(!(compareCols(getPixelCol(x, y), prevCol))) return;
    if(compareCols(getPixelCol(x, y), newCol)) return;

    setPixel(x, y, newColor);

    fillUtil(x + 1, y, newCol, prevCol);
    fillUtil(x - 1, y, newCol, prevCol);
    fillUtil(x, y + 1, newCol, prevCol);
    fillUtil(x, y - 1, newCol, prevCol);
  }
```

Výpis kódu 4.8: Vykreslení kruhu bez výplně

Nástroj „Výběr barev“ umožňuje navolit konkrétní barvu pro vykreslování do bitmapy. Tento nástroj je komponenta mtx-colorpicker z knihovny Material Extensions. Jedná se v podstatě o formulářový vstup, po jehož zakliknutí se zobrazí okno obsahující barevné spektrum, ve kterém si lze vybrat požadovanou barvu a její průhlednost nebo také zadat konkrétní barvu pomocí hexa kódu barvy.

Nástroj „Seznam barev“ je expanzní panel, který obsahuje dlaždice reprezentující jednotlivé barvy uvnitř bitmapy. Po kliknutí na danou dlaždici se provede nastavení její barvy jako právě zvolená barva uvnitř nástroje „Výběr barev“.

Nástroj „Kapátko“ slouží pro nastavení právě aktivní (zvolené) barvy v grafickém editoru na základě vybraného pixelu z bitmapy. Po kliknutí na daný pixel v bitmapě je zjištěna jeho barva a ta se následně nastaví jako právě zvolená barva uvnitř nástroje „Výběr barev“.

Nástroj „Text“ umožňuje vkládat text do sítě buněk uvnitř Canvasu. Tento nástroj se skládá ze vstupu pro textový řetězec k vložení a numerického vstupu pro velikost fontu. K převodu textu do používané sítě buněk je nutno provést jeho korektní zvětšení. Toho lze docílit vykreslením zadaného textu do jiného Canvasu, který není viditelný, poté se vezme obsah tohoto Canvasu, tedy jeho bitmapa ve formátu Uint8ClampedArray, zvětší se její obsah na požadovanou velikost pomocí Nearest Neighbour algoritmu a následně se tato bitmapa vykreslí do hlavního Canvasu uvnitř grafického editoru. Text je možno umístit na požadovanou pozicí jeho uchycením levým tlačítkem myši a následným posunem na pozici.

Nástroj „Limitace barevné palety“ slouží k omezení barev bitmapy prvku na barvy definované dle standardů Garmin16, 64 a 256. Nástroj postupně projde všechny barvy bitmapy a nalezne jim nejbližší ekvivalent v rámci zvolené Garmin palety.

Nástroj „Barevné filtry“ umožňuje aplikovat barevný filtr nebo efekt na bitmapu prvku, konkrétně lze převést jednotlivé barvy do škály šedi nebo provést jejich inverzi. V rámci toho nástroje je dialogové okno, skrze které si lze zvolit vybraný barevný filtr na základě náhledu na již upravenou bitmapu daným filtrem.

Nástroj „Generování textur/vzorů“ umožňuje vykreslit do bitmapy různé předdefinované vzory, těmi jsou: šachovnice, horizontálně orientované čáry, vertikálně orientované čáry, diagonální čáry – zleva i zprava a diamantový vzor – kosočtverce. V rámci toho nástroje je dialogové okno, skrze které si lze zvolit vybraný vzor na základě náhledu na již upravenou bitmapu daným vzorem.

Nástroj „Nahraní obrázku“ umožňuje importovat obrázek z lokálního úložiště počítače do grafického editoru a vykreslit jej do sítě buněk v Canvasu.

Nástroj „Stažení bitmapy“ slouží ke stažení právě editované bitmapy v původní velikosti.

Nástroj „Krok vpřed a vzad“ slouží pro přechod mezi provedenými akcemi v editoru. Lze pomocí něj postupně „odčinit“ provedené změny. Tento nástroj je implementován formou dvou zásobníků – jeden pro akce k „vrácení“ a jeden pro akce k „předělání“.

Nástroj „Změna velikosti bitmapy“ slouží k úpravě rozměrů bitmapy editovaného prvku. Své vlastnosti mění podle typu editovaného prvku. V případě prvků POI umožňuje nastavit výšku a šířku bitmapy jak se zachováním původního poměru stran, tak i bez. U prvků polyline se jeho schopnosti liší podle toho, jestli se jedná o polyline s bitmapou nebo bez. U polyline bez bitmapy umožňuje pouze změnu šířky vnějšího ohraničení, změnu tloušťky vnitřní čáry a převod na polyline s bitmapou. Polyline s bitmapu umožňuje nastavit pouze šířku celé bitmapy. V případě prvků polygone je tento nástroj zablokován, jelikož polygone prvky mají fixní velikost.

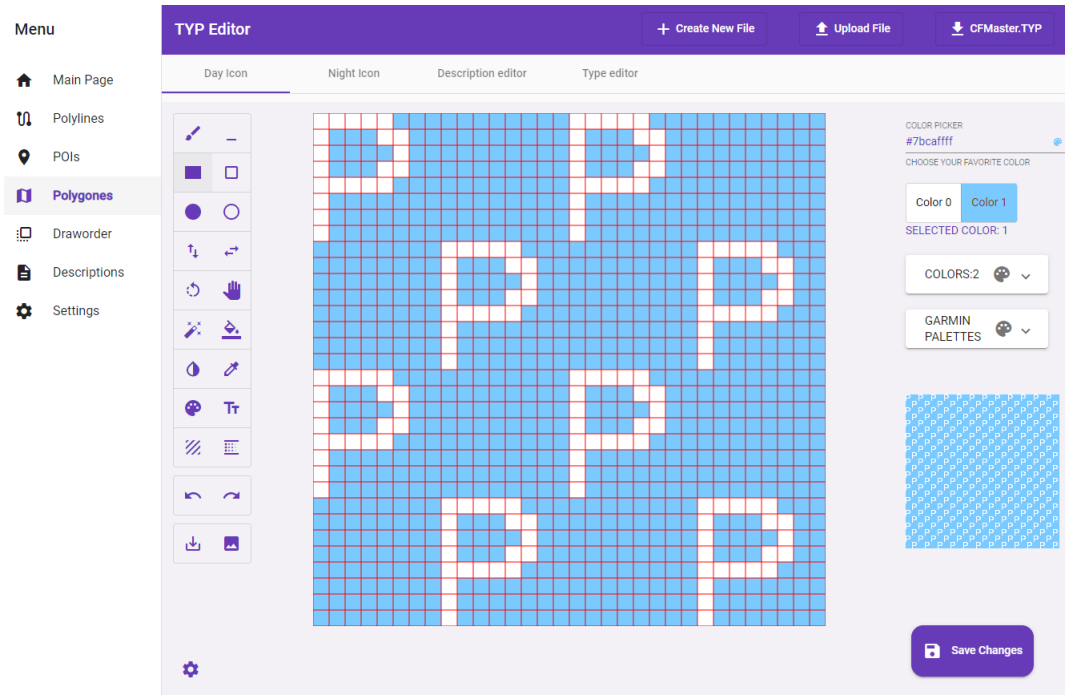
Nástroj „Kontextové menu“ je menu, které se zobrazí po kliknutí pravým tlačítkem myši do prostoru Canvasu. V tomto menu se nacházejí možnosti pro úpravu bitmapy prvku. Seznam dostupných možností se liší podle toho, jestli se edituje denní nebo noční ikona. V případě denní ikony umožňuje vyčistit celou plochu Canvasu, tedy celou bitmapu, nebo provést import noční ikony, pokud jí daný prvek disponuje. V případě noční ikony taktéž umožňuje promazání celé plochy Canvasu, ale místo importu noční ikony nabízí import denní ikony.

Grafický editor umí měnit své vlastnosti podle typu editovaného prvku. V případě, že je editovaný prvek polygone nebo polyline, editor bude limitovat maximální počet barev v bitmapě na dvě barvy. Pro přepínání mezi těmito dvěma barvami se v editoru zobrazí dvě přepínací tlačítka, kde každé reprezentuje jednu barvu. Po stisku vybraného tlačítka se jeho barva nastaví jako aktivní v rámci nástroje pro výběr barev. V případě změny aktivní barvy na jinou skrze nástroj pro výběr barev dojde k přepsání barvy pixelů bitmapy, které obsahují původní barvu, na nově zvolenou barvu. Tato změna barvy se projeví i v přepínacích tlačítkách.

Kromě limitace barev mají prvky polygone a polyline v grafickém editoru další specifickou funkcionalitu, kterou je náhled toho, jak daná bitmapa prvku bude vypadat uvnitř mapy. Jedná se o Canvas, ve kterém je daná bitmapa prvku opakovaně vykreslena ve své původní velikosti. V

případě polyline je bitmapa několikrát vykreslena v řadě za sebou – reprezentace cesty. Zato v případě polygone je bitmapa vykreslena opakovaně do šířky i do výšky – reprezentace plochy.

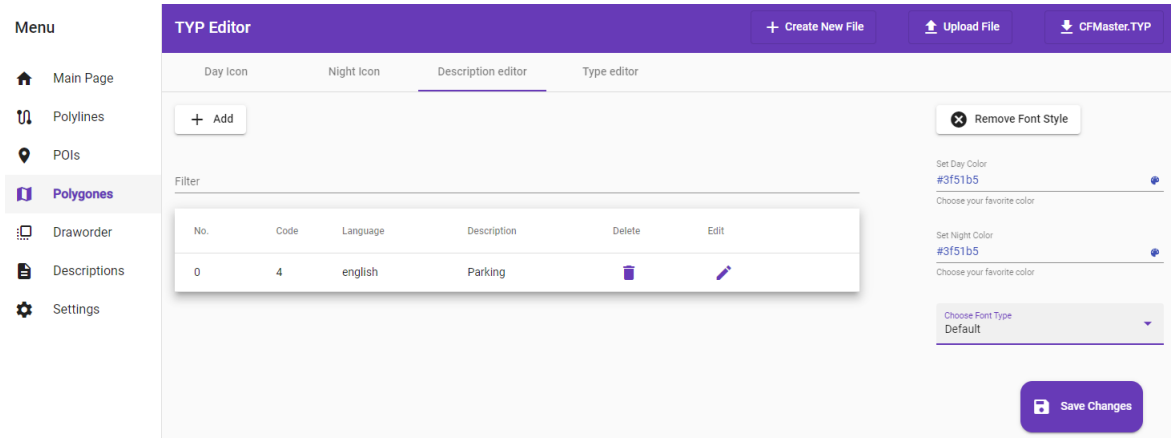
Pokud je editován prvek polyline bez bitmapy, editor limituje použití svých nástrojů jejich zablokováním. Jelikož jsou polyline bez bitmapy reprezentovány pouze jako čáry s tloušťkou ohraničení a tloušťkou vnitřní čáry, je nutno uživateli zamezit v užití jakýchkoliv kreslících nástrojů, které mění bitmapu. To obnáší například zablokování nástrojů pro kreslení tvarů, čar nebo textu. Tento limit nástrojů také platí pro editaci nočních ikon prvků polyline a polygone, a to z toho důvodu, že tyto prvky mají sdílenou bitmapu pro denní a noční ikonu. Lze jim tedy pouze měnit barvy.



Obrázek 4.6: Grafický editor

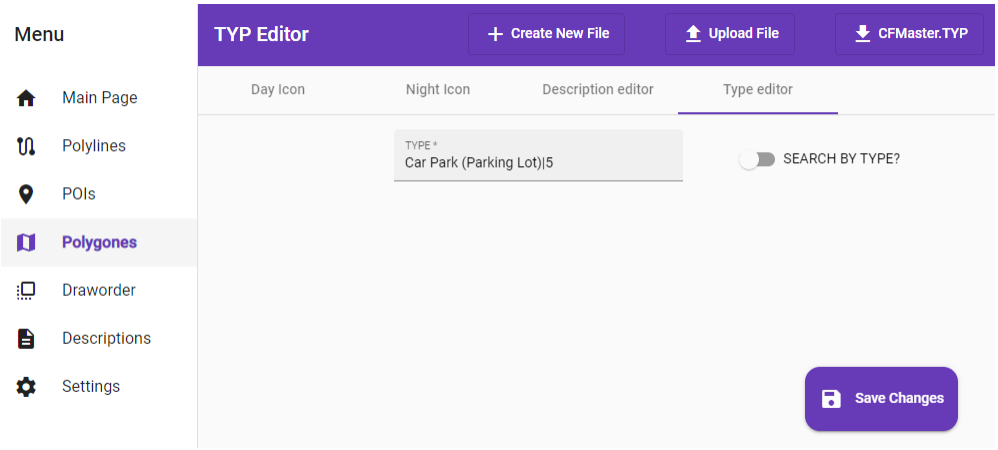
Další částí editoru je **editor popisků prvků**. Hlavní prvkem tohoto editoru je tabulka zobrazující, jaké popisky prvek obsahuje. Na každém řádku tabulky je zobrazena informace o tom, k jakému jazyku je daný popisek přiřazen, jaký je kód tohoto jazyka a tlačítka s možností editace a odstranění popisku. V případě volby odstranění dojde rovnou k vymazání daného záznamu z tabulky. V případě editace se otevře dialogové okno se vstupem pro popisek, který je již předvyplněný stávající hodnotou. Nad samotnou tabulkou se nachází tlačítko pro přidání nového popisku s jazykem. To otevře dialog se vstupem pro jazyk, který obsahuje dostupné jazyky, a se vstupem pro popisek. Dále tento editor umožňuje nastavit styl fontu, konkrétně jeho denní a noční barvu, a jeho předdefinovaný styl. Pokud daný prvek nemá nastavený žádný styl fontu, zobrazí se nad tabulkou tlačítko pro přidání stylu, které v editoru po stisknutí zobrazí formulář pro nastavení tohoto stylu. Obsahuje-li už daný prvek nějaký styl fontu, tlačítko pro přidaní bude nahrazeno tlačítkem pro odebrání stylu

a formulář pro nastavení stylu fontu bude rovnou zobrazen a předvyplněn nastavenými hodnotami.



Obrázek 4.7: Editor popisků

Poslední částí editoru je **editor typu prvku**. Jedná se o vstup s možností výběru a hledání ze seznamu definovaných Garmin typů. Každý typ je reprezentován číselným kódem, který označuje, co daný prvek reprezentuje, tedy jestli se například jedná o parkoviště, nemocnici, chodník... Tento vstup umožňuje vyhledávat jak na základě číselného kódu, tak i na základě popisku daného kódu.



Obrázek 4.8: Editor typu

V pravém dolním rohu editoru se nachází tlačítko pro uložení provedených změn. Po stisknutí tohoto tlačítka se změny uloží a uživatel bude přesměrován na hlavní stránku aplikace. Pokud ale nastane situace, že uživatel bude chtít opustit editor, aniž by uložil provedené změny, bude mu zobrazen informační dialog o neuložených změnách s možností zahození neprovedených změn a odchodu a s možností zrušení odchodu z editoru. Tato funkcionalita je zajištěna pomocí **Angular Guards**. Guards představují určitou logiku/funkcionalitu, která se má provést před načtením routy nebo před opuštěním routy. Jedná se tedy o rozhraní, která po implementaci umožňují řídit dostupnost routy

na základě podmínek poskytnutých v implementaci třídy tohoto rozhraní. Zde je konkrétně využita ochrana **CanDeactivateGuard**, která se používá k tomu, aby uživateli zabránila v navigaci mimo konkrétní routu.

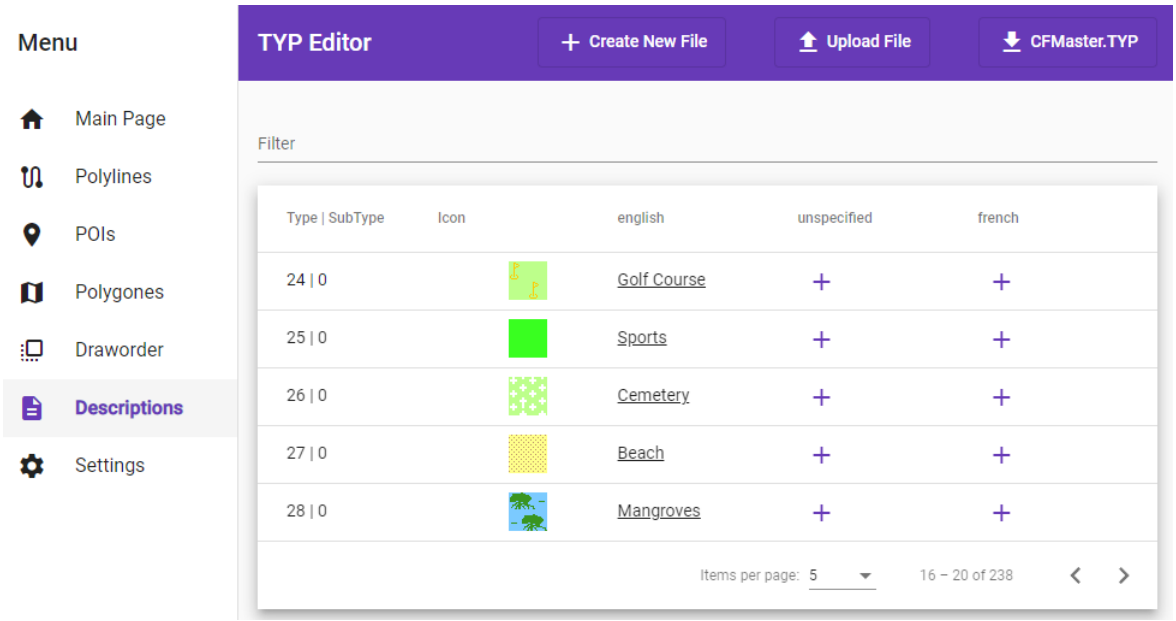
4.2.10 Komponenta „Popisky prvků TYP souboru“ (icon-descriptions)

Komponenta icon-descriptions je tabulka s možností vyhledávání a rozdělení obsahu do stránek. Tato tabulka slouží pro zobrazení a editaci popisků všech prvků v TYP souboru, tedy všech POI, polyline a polygone prvků. Tabulka obsahuje tyto sloupce:

- **Type | SubType** – sloupec obsahující typ a podtyp prvků.
- **Icon** – sloupec zobrazující denní ikonu prvků.
- **Sloupce jazyků** – tabulka dále obsahuje sloupce, které reprezentují všechny jazyky, ve kterých jsou popisky prvků napsány.

Pokud má prvek v daném jazyce popisek, bude zobrazen tento popisek. Pokud ale prvek nemá přiřazený žádný popisek k danému jazyku, bude zde zobrazeno tlačítko pro přidání popisku ke konkrétnímu jazyku. Po kliknutí na popisek nebo na tlačítko pro přidání se otevře dialogové okno, které slouží pro editaci popisku/přidání popisku pro daný jazyk.

Tato tabulka je realizována pomocí komponenty **mat-table** a komponenty **mat-paginator** z knihovny Angular Material. Komponenta mat-paginator poskytuje možnost navigace v tabulce nebo v jiném typ obsahu rozděleného do stránek.

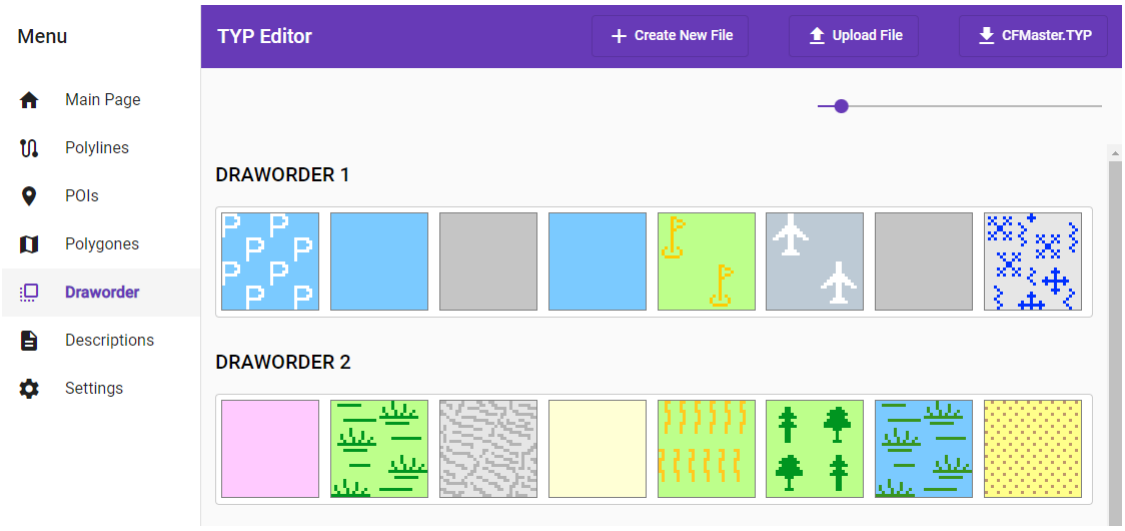


Obrázek 4.9: Komponenta icon-descriptions

4.2.11 Komponenta „Nastavení pořadí pro vykreslení polygonů“ (polygone-draworder-sort)

Komponenta polygone-draworder-sort slouží ke změně drawOrder úrovně polygonů pomocí Drag & Drop mechanismu. Jedná se o skupinu kontejnerů, kde každý jeden kontejner reprezentuje jednu draworder úroveň. V každém kontejneru se nachází polygony příslušné draworder úrovně, které jsou reprezentovány dlaždicí s jejich denní ikonou. Polygonům lze měnit draworder úroveň kliknutím na danou dlaždici a přetažením této dlaždice do kontejneru pro požadovanou draworder úroveň.

Pro realizaci mechanismu Drag & Drop byl použit modul **drag-drop** z knihovny Angular Material. Tento modul poskytuje způsob, jak snadno vytvářet Drag & Drop rozhraní z volného přetahování, řazení, přesunu mezi seznamy a jiné.



Obrázek 4.10: Komponenta polygone-draworder-sort

4.2.12 Komponenta „Navigační panel“ (Nav)

Komponenta nav představuje responzivní navigační panel pro celou aplikaci. Umožňuje snadný přesun mezi jednotlivými částmi aplikace, které jsou reprezentovány jako záložky uvnitř tohoto panelu. Navigační panel se nachází u levého okraje aplikace, pokud má dané zařízení dostatečně velkou obrazovku. V případě mobilních zařízení či jiných zařízení s malou obrazovkou se tento panel schová a lze jej „vysunout“ pomocí tlačítka. Tato komponenta obsahuje ještě další panel, který obsahuje tlačítka pro nahraní, stažení a vytvoření nového souboru.

Základní struktura této komponenty byla vygenerována pomocí **Angular Material Schematics**, konkrétně pomocí **navigation** schématu. Toto schéma vygeneruje komponentu s responzivním bočním navigačním panelem a panelem nástrojů pro zobrazení názvu aplikace.

Kapitola 5

Uživatelské testování

Kromě samotného návrhu a realizace řešení bylo potřeba v rámci této bakalářské práce, otestovat i výsledné řešení, konkrétně jednoduchost použitelnosti uživatelského rozhraní webové aplikace. A právě tímto se bude zabývat tato kapitola. Nejprve zde bude popsán způsob, jakým bude testování probíhat a poté zde budou rozebrány výsledky získané tímto testováním.

5.1 Přístup k testování

Testování uživatelského rozhraní aplikace bude probíhat formou scénářů. Každý scénář bude představovat nějakou běžnou úlohu, kterou může uživatel v aplikaci vykonat. Tyto úlohy se budou skládat ze skupiny kroků, které je nutné provést k úspěšnému dokončení dané úlohy. Ke každé úloze bude uveden vzorový postup formou kroků. Tento vzorový postup poté bude porovnán s postupem, který zvolili testovací uživatelé. Pokud bude postup uživatele shodný se vzorovým postupem nebo mu bude alespoň velmi podobný, bude to znamenat, že daná část uživatelského rozhraní aplikace je navržena správně. Pokud se ale bude uživatelský postup značně lišit od vzorového, bude to jistá indikace nevhodně navrženého uživatelského rozhraní, a bude tedy nutné vyhodnotit, jak danou část vylepšit. Každý scénář začne tím, že se testovací uživatel bude nacházet na hlavní stránce aplikace.

5.2 Scénář 1

Záměrem tohoto scénáře je otestovat nahrání nového TYP souboru do aplikace a poté v něm provést úpravu libovolného prvku polygon. Tato úprava se bude skládat z vykreslení plného zeleného obdélníku do denní ikony a přidání libovolného popisku, který bude přidružen k italskému jazyku. Provedené změny budou uloženy a editovaný soubor stažen. Viz příloha A.

5.2.1 Získané poznatky z testování

Všichni uživatelé zvládli úspěšně dokončit tento scénář. Tito uživatelé narazili na společný problém, a to na problém se systémem pro přepínáni barev v grafickém editoru u prvků, které mají limitovaný počet barev. Toto nepochopení bylo způsobeno částečně neznalostí uživatelů, jelikož nevěděli, jaké vlastnosti mohou mít různé prvky v TYP souboru, ale také tím, že nebyli schopni rozlišit rozdíl mezi dvěma tlačítky pro přepínání mezi barvami v polygonu (systém pro limitaci počtu barev) a expanzním panelem, který obsahuje veškeré barvy v bitmapě a po kliknutí na danou barvu nastaví tuto barvu do nástroje „Color Picker“. Jeden uživatel také zmínil, že by ocenil text „Download“ uvnitř tlačítka pro stažení souboru.

5.3 Scénář 2

Tento scénář bude testovat vytvoření nového prázdného TYP souboru se jménem test, PID = 1 a FID = 7. Po vytvoření se do souboru vytvoří nový prvek POI o rozměrech 20 x 20 pixelů s typem Exit a libovolným popiskem v anglickém jazyce. Do nově vytvořeného prvku POI bude poté vykresleno kolečko s modrou výplní pro denní ikonu a vloženo žluté písmeno E o velikosti fontu 12px. Provedené změny budou uloženy a nový soubor stažen. Viz příloha B.

5.3.1 Získané poznatky z testování

Všichni uživatelé zvládli úspěšně dokončit tento scénář. Nenastaly zde žádné plošné komplikace, které by ovlivnily všechny nebo alespoň většinu z nich. Jeden uživatel editoval ikonu prvku POI v domnění, že má limitovaný počet barev jako prvky Polygone či Polyline, a proto zbytečně využíval expanzního panelu s barvami bitmapy pro nastavení barvy nástroje „Color Picker“, i když tento panel nemá nic společného se systémem pro limitaci barev. Další uživatel byl překvapen při vkládání textu do ikony z mírně odlišné mechaniky oproti zbylým nástrojům grafického editoru.

5.4 Scénář 3

Účelem tohoto scénáře bude otestovat spojení prvků z jednoho souboru do druhého. V aplikaci se již bude nacházet nahraný TYP soubor. Do tohoto souboru uživatel přidá libovolné prvky polyline z jiného TYP souboru nacházejícího se lokálně v PC uživatele. Po přidání nových prvků polyline odstraní uživatel tři libovolné prvky polyline ze souboru. Nakonec uživatel limituje všem prvkům v souboru barevnou paletu podle standartu Garmin 64. Provedené změny budou uloženy a editovaný soubor stažen. Viz příloha C.

5.4.1 Získané poznatky z testování

Všichni uživatelé zvládli úspěšně dokončit tento scénář, nenastaly zde žádné plošné komplikace. Někteří uživatelé by ocenili oznámení informující o vložení nových prvků a jejich dočasné zvýraznění po vložení z jiného TYP souboru. Jeden uživatel si v bodu číslo 8 myslel, že po kliknutí na dlaždici dojde automatickému odstranění daného prvku, a ne pouze k jeho výběru/označení. K tomuto uvažování ho vedl design dlaždic v režimu odstranění, kde se v hlavičce každé dlaždice nachází ikona koše.

5.5 Scénář 4

V rámci tohoto scénáře se bude testovat změna velikosti ohraničení a vytvoření noční ikony pro libovolný prvek polyline bez bitmapy. V aplikaci se již bude nacházet nahraný TYP soubor. Uživatel tedy nejprve zvolí libovolný prvek polyline bez bitmapy a provede nastavení šířky ohraničení na hodnotu o 2px větší než stávající hodnota a poté této ikoně vytvoří noční ikonu s inverzními barvami vůči ikoně denní. Provedené změny budou uloženy a editovaný soubor stažen. Viz příloha D.

5.5.1 Získané poznatky z testování

Všichni uživatelé zvládli úspěšně dokončit tento scénář. Většina z těchto uživatelů narazila na problém se změnou velikosti ikony, konkrétně velikosti ohraničení. Dlouho jim trvalo, než našli patřičný nástroj. Jeden uživatel také očekával, že se noční ikona vygeneruje včetně provedených změn na denní ikoně, které ale ještě nebyly uloženy.

5.6 Scénář 5

Tento scénář bude testovat editaci prvku polygon. V aplikaci se již bude nacházet nahraný TYP soubor. Uživatel si vybere libovolný polygon. Tomu upraví bitmapu denní ikony následovně – vykreslí libovolně přímku jakékoliv barvy, horizontálně otočí ikonu a následně s ní provede rotaci doleva o 90 stupňů. Poté na tento polygon aplikuje diamantový vzor (texturu). Nakonec danému polygonu nastaví nejnižší hodnotu drawOrder úrovně. Provedené změny budou uloženy a editovaný soubor stažen. Viz příloha E.

5.6.1 Získané poznatky z testování

Všichni uživatelé zvládli úspěšně dokončit tento scénář. Většině z těchto uživatelů chvíli zabralo, než našli záložku v menu pro změnu draworder úrovně polygonů. Očekávali, že tato vlastnost bude editována v rámci editoru prvků. Tito uživatelé by také preferovali při změně draworder úrovně seřazení kontejnerů reprezentující jednotlivé úrovně od nejvyšší po nejnižší a ne naopak.

5.7 Výsledné poznatky

Z poznatků získaných tímto uživatelským testováním aplikace lze vyvodit následující závěr: Většina uživatelů měla na začátku problém s pochopením systému přepínání barev u prvků, kde je jejich počet barev limitován. Toto bylo zapříčiněno uživatelskou neznalostí vlastností prvků TYP souboru, ale také určitou podobností expanzního panelu s barvami bitmapy a systémem toggle tlačítek pro přepínání barev – nastavení aktivní barvy. Někteří uživatelé také zmínili, že by bylo vhodné po přidání nových prvků z jiného souboru do aplikace zakomponovat oznámení o úspěšném přidání nových prvků a tyto prvky nějak dočasně zvýraznit. Další komplikace nastala u některých uživatelů, když potřebovali změnit velikost ikony prvku. Chvíli jim trvalo, než v grafickém editoru nalezli potřebný nástroj. Za zmínku také stojí seřazení kontejnerů reprezentujících jednotlivé draworder úrovně polygonů. Uživatelé by preferovali, kdyby tyto kontejnery byly seřazeny sestupně podle úrovně.

5.8 Navrhované řešení nalezených nedostatků

Jako řešení problému týkajícího se nepochopení nástroje pro přepínání barev a expanzního panelu obsahujícího barvy bitmapy bude zvolena komponenta mat-tooltip z knihovny Angular Material. Tato komponenta představuje popisek, který se zobrazí po najetí myší na daný prvek. V tomto případě se bude konkrétně jednat o popisek expanzního panelu barev a nástroje pro přepínání barev, který bude vysvětlovat jejich účel. Další problém nastal u změny draworder úrovně polygonů, kde uživatelé očekávali sestupné seřazení kontejnerů pro jednotlivé úrovně. Proto budou tímto způsobem tyto kontejnery seřazeny.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a implementovat knihovnu pro práci s Garmin TYP soubory a také webovou aplikaci pro editaci a tvorbu těchto TYP souborů. Práce se zabývá popisem použitých moderních technologií pro tvorbu webových aplikací, návrhem a implementací knihovny pro TYP soubory a webového editoru a uživatelským testováním přívětivosti prostředí aplikace.

Nejprve bylo za potřebí vytvořit knihovnu, která bude umožňovat načtení, zápis, editaci a vytvoření TYP souboru. Tato knihovna vznikla na základě již existující implementace pro jazyk C#. Bylo ji tedy nutno přepsat do TypeScriptu, ve kterém je napsána i samotná webová aplikace, což obnášelo jisté modifikace. Poté byla knihovna rozšířena o vlastní metody a pomocné třídy. Následně byl vyvinut webový editor pro TYP soubory. Pro tvorbu editoru byl zvolen populární front-end framework Angular. Tento editor se postupně vyvíjel od pouhého načtení a zobrazení dat TYP souboru až po tvorbu nástrojů a komponent pro jeho editaci a jiné úpravy, jako například grafický editor. Nakonec bylo provedeno uživatelské testování prostředí této webové aplikace. Poznatky testovacích uživatelů byly vyhodnoceny a na jejich základě byly navrženy úpravy pro zlepšení přívětivosti a jednoduchosti tohoto prostředí.

Webovou aplikaci by bylo možno v budoucnu rozšířit o nové nástroje, zejména v grafickém editoru. Ten by mohl obsahovat možnost rozdělení obsahu bitmapy do vrstev, jako tomu je v jiných grafických editorech, jako například Adobe Photoshop. Dále by mohla aplikace umožňovat editaci zbylých dat TYP souboru, jako Active Routing a Extra POIs. Pro usnadnění používání aplikace by bylo vhodné přidat vícejazyčnost – nabídku tří nejpoužívanějších jazyků. V rámci této aplikace by také mohla být přidána možnost registrace uživatelů, která by umožňovala ukládání určitého počtu TYP souborů na uživatele.

Literatura

1. *Front-end Frameworks* [online] [cit. 2022-04-03]. Dostupné z: https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/.
2. *OBSERVABLES, OBSERVERS SUBSCRIPTIONS | RxJS TUTORIAL* [online] [cit. 2022-04-03]. Dostupné z: https://www.youtube.com/watch?v=Tux1nhBPl_w.
3. *Circle fill on a grid* [online] [cit. 2022-04-03]. Dostupné z: https://www.redblobgames.com/grids/circle-drawing/.
4. *13 Best JavaScript Frameworks For 2020* [online] [cit. 2022-04-03]. Dostupné z: https://www.lambdatest.com/blog/best-javascript-framework-2020/.
5. *2021 Developer Survey* [online] [cit. 2022-04-03]. Dostupné z: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks.
6. *List of 10 Best Front end Frameworks to Use For Web Development* [online] [cit. 2022-04-03]. Dostupné z: https://www.monocubed.com/blog/best-front-end-frameworks/.
7. *The Good and the Bad of Vue.js Framework Programming* [online] [cit. 2022-04-03]. Dostupné z: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/.
8. *Ten criteria for choosing the correct framework* [online] [cit. 2022-04-03]. Dostupné z: https://symfony.com/ten-criteria.
9. *Angular Material* [online] [cit. 2022-04-03]. Dostupné z: https://www.javatpoint.com/angular-material.
10. *Introduction* [online] [cit. 2022-04-03]. Dostupné z: https://www.learnrxjs.io/.
11. *RxJS and Angular: Why and how to use* [online] [cit. 2022-04-03]. Dostupné z: https://www.educative.io/blog/rx-js-angular.
12. *Reactive programming* [online] [cit. 2022-04-03]. Dostupné z: https://en.wikipedia.org/wiki/Reactive_programming.
13. *RxJS Operators* [online] [cit. 2022-04-03]. Dostupné z: https://rxjs.dev/guide/operators.

14. *When To Use RxJS Subject, BehaviourSubject, ReplaySubject, AsyncSubject, or Void Subject in Angular* [online] [cit. 2022-04-03]. Dostupné z: https://betterprogramming.pub/when-to-use-rxjs-subject-behavioursubject-replaysubject-asyncsubject-or-void-subject-in-angular-c2e9db61b4a0.

15. *Advantages and Disadvantages of RxJS* [online] [cit. 2022-04-03]. Dostupné z: https://www.javatpoint.com/advantages-and-disadvantages-of-rxjs.

16. *ng2-charts* [online] [cit. 2022-04-03]. Dostupné z: https://github.com/valor-software/ng2-charts.

17. *Angular Charts Demo* [online] [cit. 2022-04-03]. Dostupné z: https://valor-software.com/ng2-charts/.

18. *flex-layout* [online] [cit. 2022-04-03]. Dostupné z: https://github.com/angular/flex-layout/wiki.

19. *Angular Flex-Layout: The Alternative Layout Library for Flex-box and CSS Grid* [online] [cit. 2022-04-03]. Dostupné z: https://medium.com/ngconf/angular-flex-layout-ddf1c8fad37e.

20. *Angular Flex Layout Tutorial with examples* [online] [cit. 2022-04-03]. Dostupné z: https://www.angularjswiki.com/flexlayout/.

21. *Canvas* [online] [cit. 2022-04-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Canvas.

22. *Canvas API* [online] [cit. 2022-04-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API.

23. *Line drawing on a grid* [online] [cit. 2022-04-03]. Dostupné z: https://www.redblobgames.com/grids/line-drawing.html.

24. *Linear interpolation* [online] [cit. 2022-04-03]. Dostupné z: https://en.wikipedia.org/wiki/Linear_interpolation.

25. *Basic Image Manipulation* [online] [cit. 2022-04-03]. Dostupné z: https://www.codingame.com/playgrounds/2524/basic-image-manipulation/introduction.

26. *Flood fill Algorithm – how to implement fill() in paint?* [Online] [cit. 2022-04-03]. Dostupné z: https://www.geeksforgeeks.org/flood-fill-algorithm-implement-fill-paint//.

27. *DataView* [online] [cit. 2022-04-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/DataView.

28. *Endianness* [online] [cit. 2022-04-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Endianness.

29. *TextDecoder* [online] [cit. 2022-04-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/TextDecoder.

30. *TextEncoder* [online] [cit. 2022-04-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/TextEncoder.

31. *TYPWiz* [online] [cit. 2022-04-03]. Dostupné z: https://www.pinns.co.uk/osm/typwiz7.html.

32. *X PixMap* [online] [cit. 2022-04-03]. Dostupné z: https://en.wikipedia.org/wiki/X_PixMap.

33. *GarminCore* [online] [cit. 2022-04-03]. Dostupné z: https://github.com/FSofTlpz/GarminCore/tree/738d765132f899f2dcf72400a7ce42629f71cf5f.

34. FAIN, Yakov a Anton MOISEEV. *Angular Development with TypeScript*. Manning Publications, 2018. ISBN 9781617295348.

35. SAVKIN, Victor a Jeff CROSS. *Essential Angular*. Packt Publishing Ltd, 2017. ISBN 9781788291040.

36. SAVKIN, Victor a Jeff CROSS. *Angular Router*. Packt Publishing Ltd, 2017. ISBN 9781787287150.

Příloha A

Scénář 1 - vzorový postup v krocích

1. Kliknutí na tlačítko „Upload File“ v horním panelu a zvolení souboru k nahraní.
2. Přechod do záložky „Polygones“.
3. Kliknutí na libovolnou dlaždici reprezentující polygon prvek.
4. Zvolení nástroje „Filled Rectangle“ kliknutím do panelu nástrojů.
5. Nastavení požadované barvy v nástroji „Color Picker“.
6. Vykreslení obdélníku stlačením levého tlačítka a potažením myši.
7. Přesun do záložky „Description editor“.
8. Kliknutí na tlačítko „Add“.
9. Nastavení pole jazyka na hodnotu „italian“ a vyplnění pole pro popisek.
10. Potvrzení nového popisku kliknutím na tlačítko „Save“.
11. Uložení provedených změn v editoru kliknutím na tlačítko „Save Changes“.
12. Stažení editovaného souboru kliknutím na tlačítko s ikonou pro stažení a názvem souboru.

Příloha B

Scénář 2 - vzorový postup v krocích

1. Kliknutí na tlačítko „Create New File“ v horním panelu.
2. Vyplnění pole formuláře „File Name“ na hodnotu test, pole „PID“ na hodnotu 1 a pole „FID“ na hodnotu 7.
3. Potvrzení formuláře kliknutím na tlačítko „Create“.
4. Přesun do záložky „POIs“.
5. Kliknutí na tlačítko s ikonou znaku + v pravém dolním rohu.
6. Zvolení možnosti „New POI“ v menu.
7. Vyplnění pole formuláře „Type“ na hodnotu „Exit|32“, pole „Description Language“ na hodnotu „english“ a pole „Width“ a „Height“ na hodnotu 20.
8. Potvrzení formuláře kliknutím na tlačítko „Save“.
9. Zvolení nástroje „Filled Circle“ kliknutím do panelu nástrojů.
10. Nastavení požadované barvy v nástroji „Color Picker“.
11. Vykreslení kolečka stlačením levého tlačítka a potažením myši.
12. Zvolení nástroje „Text“ skrze panel nástrojů nebo expanzní panel „Text“.
13. Nastavení velikosti fontu na 12px a pole pro text na hodnotu „E“ uvnitř otevřeného expanzního panelu.
14. Nastavení požadované barvy v nástroji „Color Picker“.
15. Aplikování textu do bitmapy přes tlačítko „Apply“.

16. Uložení provedených změn v editoru kliknutím na tlačítko „Save Changes“.

17. Stažení editovaného souboru kliknutím na tlačítko s ikonou pro stažení a názvem souboru.

Příloha C

Scénář 3 - vzorový postup v krocích

1. Přesun do záložky „Polylines“.
2. Kliknutí na tlačítko s ikonou znaku + v pravém dolním rohu.
3. Zvolení možnosti „Merge From File“ v menu.
4. Zvolení TYP souboru pro spojení v průzkumníkovy souborů.
5. Výběr prvků polyline klikáním na dlaždice.
6. Potvrzení výběru kliknutím na tlačítko „Merge“.
7. Přepnutí do režimu odstranění skrze tlačítko „Remove Polylines“ v levém horním rohu.
8. Výběr tří prvků polyline k odstranění klikáním na dlaždice.
9. Provedení odstranění pomocí tlačítka „Remove x items“ v levém horním rohu.
10. Přesun do záložky „Settings“.
11. Rozbalení expanzního panelu s názvem „LIMIT COLOR PALETTE“.
12. Zvolení možnosti GARMIN 64.
13. Aplikování změn kliknutím na tlačítko „Save Changes“.
14. Stažení editovaného souboru kliknutím na tlačítko s ikonou pro stažení a názvem souboru.

Příloha D

Scénář 4 - vzorový postup v krocích

1. Přesun do záložky „Polylines“.
2. Přesun na editaci denní ikony polyline kliknutím na dlaždici nebo volbou skrze menu v patičce dlaždice.
3. Kliknutí na tlačítko s ikonou ozubeného kolečka nacházejícího se pod panelem nástrojů.
4. Zvolení možnosti „Resize v menu“.
5. Nastavení pole „Border Width“ na hodnotu o dva pixely větší, než je stávající hodnota.
6. Potvrzení změny pomocí tlačítka „Apply“.
7. Přesun do záložky „Night“ v editoru.
8. Kliknutí na tlačítko s ikonou znaku + uprostřed editoru.
9. Zvolení možnosti „With Inverse Colors“ v menu.
10. Uložení provedených změn v editoru kliknutím na tlačítko „Save Changes“.
11. Stažení editovaného souboru kliknutím na tlačítko s ikonou pro stažení a názvem souboru.

Příloha E

Scénář 5 - vzorový postup v krocích

1. Přesun do záložky „Polygones“.
2. Přesun na editaci denní ikony polygonu kliknutím na dlaždici nebo volbou skrze menu v patičce dlaždice.
3. Zvolení nástroje „Line“ kliknutím do panelu nástrojů.
4. Nastavení barvy v nástroji „Color Picker“.
5. Vykreslení přímky stlačením levého tlačítka a potažením myši.
6. Kliknutí na nástroj „Flip Horizontally“ v panelu nástrojů.
7. Jedno kliknutí na nástroj „Rotate Left“ v panelu nástrojů.
8. Zvolení nástroje „Apply Texture“ v panelu nástrojů.
9. Výběr možnosti „Diamond Pattern“ kliknutím na příslušnou dlaždici.
10. Aplikování vzoru/textury na iikonu kliknutím na tlačítko „Apply“.
11. Uložení provedených změn v editoru kliknutím na tlačítko „Save Changes“.
12. Přesun do záložky „Draworder“.
13. Přetažení dříve editovaného polygonu pomocí drag & drop do kontejneru s nejnižší hodnotou.
14. Stažení editovaného souboru kliknutím na tlačítko s ikonou pro stažení a názvem souboru.