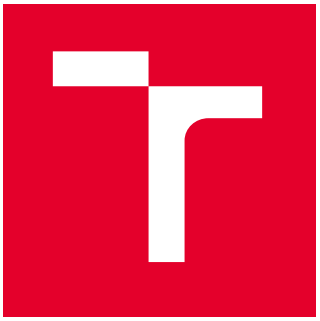


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SYSTÉM PRO VIZUALIZACI DAT Z VÝROBY POLOVODIČOVÝCH SOUČÁSTEK

SYSTEM FOR VISUALIZATION OF SEMICONDUCTOR MANUFACTURING DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Rudolf Turoň**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Petyovský, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Rudolf Turoň ***ID:*** 211188

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Systém pro vizualizaci dat z výroby polovodičových součástek

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a implementovat webovou aplikaci určenou pro vizualizaci a anotaci dat pořízených při výrobě polovodičových součástek.

1. Seznamte se a nastudujte vlastnosti a principy systémů pro vizualizaci a anotaci dat pořízených při výrobě polovodičových součástek.
2. Diskutujte možnosti stávajícího řešení dostupného u zadavatele.
3. Definujte detailní požadavky na nově navrhovaný vizualizační a anotační systém.
4. Navrhněte blokové schéma nově navrhovaného systému.
5. Realizujte frontend a backend část webové aplikace určené pro vizualizaci a anotaci dat z výroby.
6. Na vhodných testovacích datech odzkoušejte a ověřte funkčnost aplikace.
7. Prezentujte výslednou aplikaci zadavateli a zapracujte do ní případné připomínky zadavatele.
8. Demonstrujte použití aplikace na datech z výrobního procesu.
9. Zhodnoťte dosažené výsledky a navrhněte další možná vylepšení.

DOPORUČENÁ LITERATURA:

[1] CROWDER, T. J. JavaScript. Wrox, 2020, 608 stran. ISBN 1119367956.

[2] VANDERKAM, D. Effective TypeScript: 62 specific ways to improve your TypeScript. Sebastopol, CA: O'Reilly Media, 2019. ISBN 978-1-492-05374-3.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 17.5.2023

Vedoucí práce: Ing. Petr Petyovský, Ph.D.

Konzultant: Ing. Zdeněk Vincourek, onsemi SCG Czech Design Center, s.r.o.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato diplomová práce se zabývá návrhem a praktickou realizací webové aplikace pro vizualizaci a anotaci dat z polovodičové výroby. Cílem práce je provedení rešerše dostupných metod pro vizualizaci dat a technologii využívaných pro webové aplikace, definice požadavků na takovýto vizualizační systém, teoretický návrh a implementace tohoto řešení. Součástí je i testování, zapracovávání získané zpětné vazby od uživatelů aplikace a demonstrace použití aplikace na datech z výroby. Výsledná aplikace je realizována za pomocí architektury klient–server využívající REST API pro komunikaci. Frontend část aplikace umožňuje vyhledávání, vizualizaci a editaci defektních map křemíkových desek. Pro vykreslovaní grafiky interaktivní vizualizaci map křemíkových desek na webu bylo využito technologie WebGL a knihovny PixiJS. Backend část, která zajišťuje zpracování dat a komunikaci s ostatními systémy, byla realizována za pomocí Java 17 ve spojení se Spring Boot. V závěru práce jsou diskutovány dosažené výsledky a shrnuty možnosti dalšího vylepšení aplikace.

KLÍČOVÁ SLOVA

Výroba polovodičových součástek, vizualizace výrobních dat, webová aplikace, grafické uživatelské rozhraní, křemíkové desky, mapa křemíkové desky

ABSTRACT

This master thesis focuses on the design and implementation of a web application for visualization and annotation of semiconductor manufacturing data. The aim of the thesis is to conduct a survey of available data visualization methods and technologies used for web applications, define the requirements for such a visualization system, theoretical design and implementation of this solution. It also includes testing, incorporation of feedback obtained from application users and demonstration of application use on production data. The final application is implemented using a client-server architecture through a REST API for communication. The frontend part of the application allows searching, visualization and editing of the wafer map defects. WebGL technology and PixiJS library were used to render graphics for interactive visualization of the wafer maps on the web. The backend part is implemented using Java 17 in conjunction with Spring Boot technology and provides data processing and communication with other systems. The thesis concludes with a discussion of the results obtained and a summary of the possibilities for further improvements of the application.

KEYWORDS

Semiconductor components fabrication, production data visualisation, web based application, graphic user interface, silicon wafers manufacturing, wafer map

TUROŇ, Rudolf. *Systém pro vizualizaci dat z výroby polovodičových součástek*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 115 s. Diplomová práce. Vedoucí práce: Ing. Petr Petyovský, Ph.D

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Rudolf Turoň
VUT ID autora:	211188
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Systém pro vizualizaci dat z výroby polovodičových součástek

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé diplomové práce panu Ing. Petrovi Petyovskému, Ph.D. za cenné rady, odborné vedení, konzultace, trpělivost a hodnotné připomínky k práci. Chci také poděkovat konzultantovi ze společnosti onsemi Ing. Zdeňkovi Vincourkovi za jeho odborné vedení, poskytnuté znalosti z oblasti technologie výroby polovodičových součástek, věcné připomínky a věnovaný čas při vypracovávání mé diplomové práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Diplomová práce se zabývá návrhem a praktickou realizací webové aplikace pro vizualizaci a anotaci dat z polovodičové výroby nazvanou MapSpyWeb pro společnost onsemi. Polovodičové součástky se nejčastěji vyrábí na křemíkových deskách zvaných *wafer*. Na jedné takové křemíkové desce mohou být i statisíce polovodičových součástek. V průběhu výroby jsou součástky na křemíkových deskách mnoha způsoby měřeny a kontrolovány. Může se jednat o elektrické, optické či další typy měření, při kterých je kontrolováno, zda je součástka funkční a splňuje požadované parametry. Tyto data získaná při výrobě se ukládají do souborů obsahujících mapu defektů křemíkové desky. Cílem diplomové práce je modernizovat systém, který umožní zaměstnancům s těmito daty efektivně pracovat a vyhledávat v nich.

Tento anotační a vizualizační systém najde uplatnění u zaměstnanců, kteří mají na starost optimalizaci výrobního procesu. Systém umožní zpětně prohledávat a vizualizovat data z výroby. Na základě nich mohou zaměstnanci hledat příčiny chybovosti ve výrobě a následně optimalizovat výrobu tak, aby byla co nejefektivnější z hlediska výtěžnosti a zisku společnosti. Ruční anotace dat zase umožní zajistit vyřazení polovodičových součástek, u kterých je operátorem již v průběhu výroby nalezen defekt. Tím je zajištěna vysoká výstupní kvalita vyrobených součástek.

Společnost onsemi již má dvě aplikace sloužící k podobnému účelu. Jedna je určena k vizualizaci a vyhledávaní map a druhá k jejich editaci. Obě aplikace však v současné době zastarávají a jejich další podpora z hlediska softwaru přestává být finančně efektivní. Objevují se u nich i výkonnostní problémy při vizualizaci map křemíkových desek již od řádově tisíců vykreslených součástek. Kromě toho existuje nespočet různých verzí těchto aplikací, s čímž souvisí problémy s kompatibilitou a jejich distribucí uživatelům v rámci společnosti. Uvedené souhrnné problémy se snaží řešit v této práci nově navržená a realizovaná aplikace MapSpyWeb.

Mezi hlavní cíle práce patří získání informací o stávajících systémech ve společnosti onsemi a přesná specifikace požadavků zadavatele na vizualizační a anotační systém. Záměrem je také provedení teoretického návrh na základě rešerše webových technologii pro vizualizaci a anotaci dat. V současné době se pro tvorbu webových aplikací užívají zejména jazyky JavaScript [1] a TypeScript [2], které budou využity i pro praktickou realizaci této diplomové práce. Výsledkem diplomové práce má být dokončená implementace systému, díky kterému bude uživatel schopen vyhledávat v datech z výroby, nalezená data budou vhodně vizualizována a bude umožněna anotace těchto dat. Součástí by mělo být také důkladné otestování aplikace a demonstrace reálného využití aplikace ve výrobě. Součástí dalších cílů diplomové práce je i sbírání připomínek a zpětné vazby od budoucích uživatelů, aby bylo možné dodat zadavateli co nejpřínosnější řešení.

1 Výroba polovodičových součástek a testování jejich kvality

První kapitola práce se věnuje úvodu do výroby polovodičových součástek a firmě onsemi, která vyrábí polovodiče v České republice i ve světě. Tato semestrální práce je zaměřená na vývoj anotačního a vizualizačního systému pro IT UMR oddělení společnosti onsemi v Rožnově pod Radhoštěm. Dále je zde stručný popis výroby polovodičových součástek princip jejich testovaní.

1.1 Společnost onsemi

Onsemi (dříve ON Semiconductor) je nadnárodní akciová společnost, která se zabývá výrobou polovodičů a integrovaných obvodů. Společnost se zaměřuje zejména na automobilový průmysl, aerospace, vojenství, lékařské aplikace, LED osvětlení a výpočetní techniku. Firma se také zabývá nejnovějšími trendy, jako je průmysl 4.0, elektromobilita, senzory pro autonomní řízení, čistá energie a 5G sítě. Onsemi sídlí ve Phoenixu v americkém státě Arizona a její pobočky a výrobní závody jsou rozmístěny po celém světě.

Onsemi v České republice

Společnost zaměstnává celosvětově více než 35 000 zaměstnanců. Z toho je přibližně 2 200 zaměstnanců v České republice. Ve vývojovém středisku v Brně se vyvíjí analogové a smíšené analogově-digitální integrované obvody pro automobilový průmysl. Jedná se hlavně o obvody pro bezkontaktní indukčnostní snímače polohy, ultrazvukové snímače, LED kontroléry a kontroléry krokových motorů. Dále se brněnské centrum zaměřuje na integrované obvody pro digitální komunikaci součástek automobilů a obvody pro řídící jednotky automobilů. [3]

Rožnovské vývojové centrum vyvíjí analogové a smíšené analogově-digitální integrované obvody pro spotřební elektroniku, komunikace, výpočetní techniku, průmyslovou elektroniku a automobilový průmysl. Zde vyvíjené integrované obvody využívají levnější bipolární i modernější unipolární technologie (CMOS, BiCMOS, VHV CMOS). Navrhované produkty jsou typicky stabilizátory napětí s nízkým úbytkem napětí, obvody pro řízení nízkonapěťových zdrojů pro přenosné přístroje, ovládače výkonových zesilovačů pro mobilní telefony, řídicí obvody pro nabíječky akumulátorů, obvody pro řízení spínaných zdrojů a převodníky DC-DC.

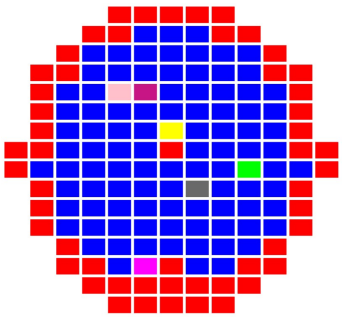
V Rožnově pod Radhoštěm sídlí také divize zabývající se testováním polovodičů a čipů, kde skupina specialistů vyvíjí nové testovací metody a testovací software

i hardware pro testovaní zde navržených integrovaných obvodů a pro automatizované testovaní ve výrobě.

IT centrum v Rožnově pod Radhoštěm bylo založeno v roce 1999 a zabývá se vývojem, podporou a nasazováním IT technologií v rámci celé korporace onsemi v oblasti výroby a automatizace. Hlavními oblastmi IT, kterým se zde věnují, jsou CIM (Computer Integrated Manufacturing), analýza a zpracování dat, ERP, web, business logika, databáze a vývoj aplikací. Místní pobočka podporuje výrobní aplikace v mnoha různých lokalitách v Evropě, USA a Asii. V současné době v IT centru pracuje více než 100 IT odborníků, mimo jiné i několik projektových manažerů, kteří v rámci celé korporace z Rožnova řídí podstatnou část projektů z oblasti informačních technologií. [4]

1.2 Výroba polovodičů

Polovodičové součástky nejčastěji vznikají na křemíkových deskách čistého křemíku. Na těch se následně formují malá elektronická zařízení, které ve výsledku tvoří integrované obvody. Nejprve je vytvořen velký krystal křemíku, že kterého je *Czochralskiho* metodou vytvořen křemíkový válec zvaný ingot. Ten se nařeže na tenké plátky (křemíkové desky), které se poté leští. [5]



Obr. 1.1: Ukázka vizualizace mapy křemíkové desky.

Dále křemíková deska mnohokrát projde několika opakujícími se kroky. Při těch jsou vytvářeny požadované vodiče, polovodiče a izolační vrstvy, které nakonec tvoří polovodičovou součástku. Mezi zmíněné kroky patří fotolitografie. Při té se na desku nanese vrstva fotorezistu nereagujícího na leptavé látky, který je pak přes masku osvěcován ultrafialovým světlem. Osvětlený fotoresist je vyvoláván vývojkou, čímž na desce zůstane nanesen pouze v potřebných místech. V dalším kroku se leptáním odstraňuje požadovaný materiál, který není chráněn fotorezistem. Poté jsou do oblastí nepokrytých fotorezistem implantovány ionty, které upravují elektrické vlastnosti těchto oblastí (podporují nebo zabraňují průchodu elektrického proudu). Při

následné difuzi probíhá oxidace oblastí, které nejsou pokryty fotorezistem. Difuze se nejčastěji používá k vytvoření oblastí source, drain a kanálu v tranzistoru. [5]

Následně jsou prováděny testy součástek a rozřezání desky na jednotlivé čipy. Zde jsou vyřazeny všechny čipy, které byly vyhodnoceny jako vadné. Poté jsou čipy vloženy do obalu a pomocí zlatých drátků připojeny ke konektorům (angl. *bonding*). Nakonec jsou čipy zapouzdřeny a je provedeno jejich finální testovaní. [5]

1.3 Testování polovodičů

Při výrobě čipů probíhá velké množství testování a inspekce čipů na křemíkové desce. Jednou z oblastí testování je optická inspekce (angl. AOI *Automated Optical Inspection*). Při té je křemíková deska umístěna pod mikroskop, kde probíhá automatická optická inspekce. Hledá se mechanické poškození, kontaminace částicemi či vady struktury. [5]

V průběhu procesu výroby jsou desky testovány také elektrickým měřením zajištěným přístroji *prober* a *tester*. *Prober* slouží k zarovnání desky a zajištění kontaktu s testovaným čipem pomocí mikroskopických sond. Přístroj *tester* provádí samotné elektrické měření na čipu. Může se jednat o měření napětí, proudu, časovaných konstant a mnoha dalších veličin. Výsledky měření jsou zaznamenány a je vyhodnoceno, zda se nacházejí ve specifikačních limitech. [5]

Kromě fyzického testování je využíváno mnoha statických metod zajišťujících vysokou kvalitu výstupních produktu. Jedna z využívaných statistických metod je označena GDBC (angl. *Good Die in a Bad Cluster*). Tato metoda označí dobrý čip na desce jako špatný, pokud je obklopen mnoha špatnými čipy. Postup vychází z principu, že čím více špatných sousedů čip má, tím je pravděpodobnější, že v budoucnu selže nebo nebude zaručena časová stálost jeho parametrů. [5]

Dalším příkladem statistického testu je DPAT (angl. *Dynamic Part Average Testing*). Tato metoda slouží k detekci odlehlých hodnot, které mohou indikovat vadu čipu. Pro každý čip je určena horní a dolní mez výsledků testu čipu. Následné jsou čipy nacházející se mimo tyto meze vyřazeny. Meze jsou určovány na základě vzorce vycházejícího z průměru a směrodatné odchylky výsledků všech čipů na desce. [5]

Výsledky z testování se ukládají do souborů obsahujících mapy křemíkových desek (*wafer mapy*). Tyto soubory udržuji mnoho informací o křemíkové desce. Samotná mapa obsahuje prvky (čipy), přičemž každý prvek je zařazen do určité kategorie. Kategorie jsou rozděleny na dva typy, a to kategorie obsahující prvky, které prošly testem (angl. *pass*), a kategorie s prvky, které neprošly (angl. *fail*). Prvkům, jež neprošly, je ještě číslem přiřazena skupina, která indikuje příčinu selhání. Tyto mapy jsou následně vizualizovaný tak, že každému typu a skupině prvku je přiřazena barva. Ukázka vizualizace mapy křemíkové desky je možno vidět na obrázku 1.1. [5]

2 Teorie a rešerše nástrojů využitelných pro vizualizaci a anotaci dat

Kapitola s názvem teorie a rešerše nástrojů využitelných pro vizualizaci a anotaci dat se zabývá primárně rešerší a vysvětlením dostupných technologií pro účely vyvíjeného systému. Rešerše se zaměřuje primárně na webové technologie, u kterých je očekávána dlouhodobá podpora a technologie pro tvorbu aplikací architektury klient–server. Jsou zde vysvětleny principy několika programovacích jazyků s využitím pro vývoj frontendu či backendu. S frontend částí aplikace souvisí uvedená rešerše technologií sloužících k vkreslovaní grafiky na webu. Dále jsou v této kapitole zmíněna rozhraní, která je možné využít pro komunikaci mezi aplikacemi, a některé frameworky sloužící ke zjednodušení vývoje komplexních aplikací.

2.1 Programovací jazyk JavaScript

JavaScript je programovací jazyk, který se používá zejména na webu, avšak, jak je možné zjistit dále v této kapitole, používá se i v mnoha jiných prostředích. Tento jazyk aktuálně využívá většina webových stránek a podporují ho všechny moderní webové prohlížeče. Díky tomu je JavaScript nejrozšířenějším programovacím jazykem v historii. Jedná se o dynamický interpretovaný vysokoúrovňový programovací jazyk, který podporuje objektově orientované programování a je netypový. Výraz „netypový“ znamená, že typy proměnných jsou kontrolovány až za běhu programu, a tedy, aby bylo možné typy zjistit, je nutné program spustit. Název JavaScript může vyvolávat dojem, že se jedná o jazyk vycházející z programovacího jazyka Java, avšak tato myšlenka je zavádějící a zmíněné dva jazyky spolu nijak nesouvisí. Respektive pouze povrchová JavaScript syntaxe vychází z jazyka Java, jinak jsou tyto jazyky zcela odlišné. JavaScript je totiž založen na málo známých jazycích *Scheme* a *Self*. Důležité je také zmínit, že se již dávno nejedná pouze o jazyk skriptovací, jak by se mohlo z názvu zdát, ale je to efektivní nástroj vhodný pro všeobecné použití v rozsáhlých projektech. [6]

Jazyk, jejž téměř každý nazývá „JavaScript“, je standardizován organizaci ECMEA *(European Computer Manufacturer's Association)* jako „ECMAScript“. Jednotlivé verze jazyka jsou pojmenovávány dvěma způsoby. V roce 2010 vznikla verze standardu „ECMAScript 5th Edition“ označovaná jako *ES5*. Následně, v roce 2015, byla vydána šestá edice označovaná*ES6*, která definuje standard „ECMAScript 2015“, nebo také *ES2015*. Od té doby jsou nové verze jazyka vydávány ročně a označují se *(ES201d6, ES2017, ES2018, . . .)*, či také *(ES7, ES8, ES9, . . .)*. Významnou verzí programovacího jazyka JavaScript byla *ES6*, která přidala syntaxi tříd a mo-

dulů, díky čemu se z jazyka JavaScript stal jazyk vhodný pro rozsáhlé softwarové inženýrství. Dále v této diplomové práci bude jazyk „ECMAScript“ označován jako „JavaScript“. [1, 6]

Hostitelská prostředí

Stejně jako jiné programovací jazyky, i Javascript potřebuje pro provádění základních vstupně–výstupních operací platformu nebo standardní knihovnu. Jádro jazyka definuje minimální API pro práci s čísly, textem, poli a dalšími, avšak nedefinuje žádné vstupní ani výstupní operace. Mezi vstupně výstupní operace můžeme zařadit například práci se sítí, úložištěm nebo grafikou hostitelského prostředí, na kterém JavaScript běží. Za zmíněné vstupně–výstupní operace je zodpovědné právě hostitelské prostředí. [6]

Nejčastějším hostitelským prostředím pro JavaScript je webový prohlížeč. Jako hostitelské prostředí však lze využít i *Node.js*, který umožňuje jazyku JavaScript přístup k celému operačnímu systému. Díky tomu může například zapisovat a číst soubory, obsluhovat příchozí HTTP požadavky, nebo posílat a přijímat data po síti. Proto je *Node.js* obvykle využíván k implementaci backend částí aplikací, nejčastěji webových serverů, ale i jiných typů serverů či newebových aplikacích a embedded řešení. Hostitelské prostředí musí obsahovat *engine*, někdy také nazývaný *VM* (angl. *virtual machine*), který spouští JavaScriptový kód. Engine je zodpovědný za parsování kódu, interpretaci nebo kompilaci kódu a spuštění kódu v prostředí, které funguje podle specifikace. Různé webové prohlížeče využívají různý engine. Engine V8 od společnosti Google využívá Chromium, Opera a Microsoft Edge verze 79 a novější. Mozilla Firefox používá pro interpretaci engine SpiderMonkey a Internet Explorer engine JScript, který je však velmi zastaralý. [1, 6]

2.2 Programovací jazyk TypeScript

TypeScript je vysokoúrovňový programovací jazyk. Jedná se o trochu neobvyklý jazyk, jelikož neběží interpretovaně ani se nekompiluje do bytového kódu. Tento jazyk se totiž kompiluje do jiného vysokoúrovňového jazyka JavaScript popsaného v kapitole 2.1. V syntaktickém smyslu je TypeScript typová nadmnožina jazyka JavaScript. To znamená, že pokud máme validní program v jazyce JavaScript, vždy se jedná také o validní program v jazyce TypeScript. Naopak však dané pravidlo neplatí. Díky tomu je možné ve stávajících kódech napsaných v jazyce JavaScript jednoduše změnit přípony souborů z *.js* na *.ts* a začít velmi jednoduše využívat výhod TypeScriptu. Důležitým cílem a benefitem tohoto programovacího jazyka je,

že se pokouší detekovat kód, který po spuštění vytvoří výjimku, aniž by bylo nutné kód spouštět. [2]

Kompilátor jazyka TypeScript se nazývá „TypeScript Compiler (TSC)“. Obecně jsou soubory programu analyzovány a převedeny překladačem na abstraktní syntaktický strom (angl. *abstract syntax tree (AST)*). Poté je provedena typová kontrola AST pomocí nástroje zvaného *typechecker*. Díky této kontrole se kompilátor ujistí, že program bude fungovat tak, jak je očekáváno. Nakonec je vygenerován JavaScriptový kód, který pak může být spuštěn pomocí JavaScript enginu. Z daného postupu vyplývá, že kompilátor TSC při generování JavaScript kódu nebere ohled na datové typy, a ty tudíž v rámci programu v jazyce TypeScript nikdy neovlivní generovaný výstupní JavaScript kód. Typy, které TypeScript přidává, tedy nemají žádný vliv na výkon výsledného programu. Kromě typové kontroly kompilátor obsahuje také transpilátor, který převádí nové verze jazyka, například „next-generation“ JavaScript a TypeScript, do starší verze jazyka JavaScript, která má podporu u většiny prohlížečů. Programátor tedy může využívat JavaScript funkcionality, jenž budou podporovány webovými prohlížeči později v budoucnosti. [2, 7]

Datové typy jazyka TypeScript lze definovat explicitně pomocí syntaxe jazyka, avšak je také možné je v některých případech nechat odvozovat ze zdrojového kódu samotný TypeScript, jak lze vidět ve výpisu 2.1.

Výpis 2.1: Ukázka definice proměnné v jazyce TypeScript.

```
let a: number = 5; // definice typu proměnné explicitně pomocí syntaxe
let b = 5; // typ proměnné je odvozen z inicializační hodnoty
```

Skutečnost, že TypeScript zachytí většinu syntaktických i typových chyb v programu, které by za běhu vyvolaly výjimky již při kompilaci, je pro programátora velmi výhodná. Přesto existuje mnoho chyb, které TypeScript při kompilaci nezachytí. Jedná se například o přetečení zásobníku, indexování mimo rozsah pole, chybné vstupy od uživatele a jiné. Další možný problém s typy může nastat při volání požadavku na API. Pokud je požadavek proveden za běhu programu a API vrátí data jiného typu, než byl definován programátorem v jazyce TypeScript, bude výsledný běžící JavaScript program pracovat s daty, která byla přijata. To znamená, že dané datové typy nebudou odpovídat těm, které byly původně programátorem definovány. Z tohoto důvodu může být TypeScript poměrně matoucí, právě pokud datové typy za běhu programu neodpovídají typům deklarovaným v programu. Je proto velmi důležité se pokusit vyhnout všem situacím, kdy by hodnota proměnných mohla mít za běhu jiný datový typ, než je deklarovaný. [2, 7]

Každý projekt napsaný v jazyce TypeScript by měl v kořenové složce obsahovat soubor `tsconfig.json`. Tento soubor definuje například to, které soubory mají být zkompilovány, kam se mají zkompilované soubory ukládat či do jaké verze jazyka

JavaScript se má kód kompilovat. Dále je zde stanoveno, jestli má být kompilátor striktní, což znamená, že kompilátor bude vyžadovat, aby celý kód byl striktně typový. Dalším souborem, který je velmi doporučeno mít v kořenové složce, je `tslint.json`. Ten definuje stylistické konvence pro psaní kódu, jako například používání mezer nebo tabulátorů, počet mezer pro odsazení a mnoho dalších. Je velmi vhodné tento soubor používat, jelikož je díky němu možné udržovat konzistentní styl psaní zdrojového kódu napříč celým projektem, přestože na něm pracuje více programátorů. [7]

2.3 Webové frameworky

Aby vývojář webové aplikace nemusel vytvářet a programovat všechny části webové aplikace sám, používají se často různé knihovny a frameworky. V těch je již vyřešeno mnoho technických problémů a detailů, které by jinak musel řešit každý vývojář individuálně. Jedná se o například o implementaci směrování (angl. *routing*) na webu, síťového klienta, správu stavů frontendu a dalších užitečných nástrojů. Díky tomu je možný mnohem rychlejší vývoj a větší soustředění na podstatné problémy při budování nové aplikace. Velká část webových frameworků využívá jazyk TypeScript (viz podkapitola 2.2). TypeScript umožňuje do frameworků zahrnout i typové konfigurace nízkoúrovňového DOM API (objektovému modelu HTML dokumentu), a vyvíjet tak frontend část bezpečně a s menší chybovostí díky kontrole typů. [7]

2.3.1 Framework Angular

Framework Angular [8] vychází z frameworku AngularJs. Pro vývoj za pomocí této vývojové platformy se využívá jazyk TypeScript, přičemž i samotný Angular je napsán v jazyce TypeScript. Angular je poměrně rozsáhlý framework, který kromě základních nástrojů pro vykreslování webu a práci s komponenty obsahuje také například HTTP klienta pro vytváření síťových požadavků a komunikaci se serverem, směrování (angl. *routing*) mezi stránkami, Dependency Injection pro vkládání závislostí a také nástroje pro testování. [7, 9]

Kompilátor frameworku Angular

Jak již bylo výše zmíněno, framework Angular využívá programovací jazyk TypeScript, avšak nepoužívá přímo kompilátor jazyka TypeScript, ale kompilátor, který je součásti příkazového řádku (CLI) frameworku Angular. Jedná se o Ahead-of-Time (AoT) kompilátor, který nad kódem provede nejdříve mnoho optimalizací a transformací, a až poté je vygenerován TypeScript kód, který je kompilován na JavaScript.

Daný proces je potřebný proto, aby prohlížeč ani TypeScript kompilátor nebyl schopen zpracovat syntaxi frameworku Angular. Ahead-of-Time kompilátor převádí kód do efektivního JavaScriptu již během sestavení aplikace. Díky tomu se do prohlížeče stáhne předkompilovaný kód, který je možné okamžité spustit a vykreslovat aplikaci. Také odpadá nutnost separátních požadavků na externí HTML a CSS soubory, jelikož kompilátor tyto šablony a styly přidá inline do výsledného JavaScriptu. Díky kompilaci šablon a stylů ještě předtím, než je kód spuštěn u klienta, je zajištěna větší bezpečnost proti útoku nazvanému *JavaScript Injection*[1]. [7, 10]

V rámci frameworku Angular je možné použít i Just-in-Time (JIT) kompilátor, který kód kompiluje až při běhu přímo v prohlížeči. Ten neslouží pro produkční účely, ale pro vývoj aplikace. Narozdíl od Ahead-Of-Time překládače, zde není potřeba provést sestavení celé aplikace, která může obsahovat stovky zdrojových souborů při každé malé změně, což je obrovskou výhodou při vývoji. V tomto případě je kompilován pouze soubor, který byl změněn. Programátor webové aplikace může tedy velmi rychle postupovat ve vývoji aplikace, neboť jeho dílčí změny ve zdrojovém kódu se téměř okamžitě projevují v aplikaci. [10]

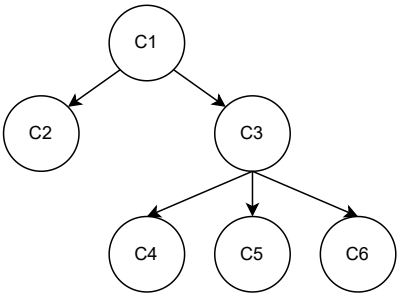
Komponenty ve frameworku Angular

Komponenty můžeme považovat za základní stavební kameny frameworku Angular. Ve většině případů komponenta zodpovídá za jednu určitou malou část webová stránky. Může například obsahovat formulář, do kterého uživatel zadává informace, nebo zobrazovat seznam zpráv. Komponenty jsou seskupovány do stromové struktury tak, že tvoří větší celek, a tato stromová struktura je vykreslena v rámci HTML značky `<body>`. Z daného principu vyplývá, že komponenty mohou mít ve stromu rodičovskou komponentu a několik potomků, se kterými může komponenta různými způsoby komunikovat. V obrázku 2.1 lze vidět možnou stromovou strukturu komponent. [9]

Komponenta se většinou skládá z jednoho až čtyř souborů. Je definována souborem `název-komponenty.component.ts`, ve kterém je třída jazyka TypeScript anotovaná klíčovým slovem `@Component`. V bodech níže je uveden seznam souborů, které komponenta typicky obsahuje.

- Třída komponenty (soubor `nazev-komponenty.component.ts`), která obsahuje program v jazyce TypeScript s logikou a zajišťuje interakci s HTML této komponenty.
- HTML komponenty (soubor `nazev-komponenty.component.html`) definující zobrazení, které interaguje s třídou komponenty. Ve třídě komponenty je cesta

[1] Útok metodou JavaScript Injection spočívá ve vložení škodlivého kódu do stránky, kdy tento kód je zobrazen a proveden u jiných uživatelů.



Obr. 2.1: Ukázka možné stromové struktury komponent v jazyce Angular.

k tomuto souboru definována pomocí vlastnosti `templateUrl:'./nazev-komponenty.component.html'` ve výpisu 2.2 na třetím řádku.

- Styl komponenty (soubor `nazev-komponenty.component.scss`) definuje specifický styl, který je použit pouze pro danou přiřazenou komponentu. Cestu k souboru se stylem obsahuje vlastnost `styleUrls:'[./nazev-komponenty.component.scss']` ve výpisu 2.2 na čtvrtém řádku.
- Unit test komponenty (`nazev-komponenty.component.spec.ts`) je určen pro testování této komponenty.

Ve výpisu 2.2 lze vidět ukázkovou definici komponenty ve frameworku Angular [9].

Výpis 2.2: Ukázka definice komponenty v Angular frameworku

```
1  @Component({
2    selector: 'app-my-component',
3    templateUrl: './my-component.component.html',
4    styleUrls: ['./my-component.component.css']
5  })
6  export class MyComponent {
7    @Input() user: User;
8  }
```

Komponenty jsou organizovány do funkčních bloků zvaných moduly. Moduly pomáhají rozdělit aplikace do funkcích celků, díky tomu je jednoduší se ve zdrojovém kódu aplikace orientovat a také ji udržovat. Každá Angular aplikace musí vždy obsahovat minimálně jeden modul nazvaný `AppModule` a každá komponenta musí být registrována alespoň u jednoho modulu, aby jí framework rozpoznal. [9]

Šablony frameworku Angular určené pro vykreslení komponent kombinují syntaxi značkovacího jazyka HTML a syntaxi frameworku Angular. Každé komponentě je přiřazen selektor, což je HTML značka (angl. *HTML tag*) identifikující danou

komponentu a umožňující její použití v HTML souboru. Ukázku definice selektoru je možné vidět ve výpisu 2.2 na řádku číslo 2 jako `selector: 'app-my-component'`. V HTML šabloně pak můžeme pomocí selektoru `<app-my-component>` určitou komponentu použít, a framework poté na daném místě vytvoří požadovanou komponentu. [9]

Služby ve frameworku Angular

Framework Angular obsahuje mechanizmus pro vkládání závislostí (DI), které je řešeno pomocí takzvaných služeb. Služba je třída jazyka TypeScript označená anotací `@Injectable`. Tato anotace službu identifikuje, a následně je možné službu vložit jako závilost pomocí konstruktoru do komponenty nebo do jiné Angular služby. [9]

Služby jsou většinou využívány pro implementaci logiky, která nepatří do komponent jazyka Angular. Komponenty by měly obsahovat kód, který je zodpovědný za to, co je v rámci dané komponenty vykreslováno do webového prohlížeče. Ostatní logika aplikace, jako například zabezpečení volání API či jiné funkce nesouvisející se zobrazením, by měly byt umístěny převážně do služeb. [9]

2.4 Webová grafika

Pro vykreslovaní grafiky na webu existuji tři nejpoužívanější přístupy podporované drtivou většinou aktuálních prohlížečů. Jedná se o vykreslování pomocí SVG, Canvas a WebGL. Právě zmíněným nástrojům a jejich využití se věnuje tato sekce.

Dříve než byla vytvořena technologie HTML5, se pro webovou grafiku používal často buď Flash, nebo applety. Avšak dané způsoby se dnes již pro vykreslování grafiky na webu nepoužívají, a to z mnoha důvodu, ať už se jedná o výkon, skutečnost, že applety a Flash nefungovaly v prohlížečích bez doplňkových softwarů či mnoho bezpečnostních rizik, která přinášely. [11]

2.4.1 Škálovatelná vektorová grafika SVG

První nástroj, který bude popisován, je škálovatelná vektorová grafika SVG (angl. *Scalable Vector Graphics*). Pomoci ní je možné řešit mnoho problému týkajících se dvourozměrné grafiky a designu. Podpora pro SVG existuje u všech hlavních prohlížečů, jako je Chrome, Firefox, Safari a Microsoft Edge. SVG je značkovací jazyk založený na XML. Při vložení SVG do HTML stránky je možné s SVG grafikou manipulovat pomocí CSS stylů a jazyka JavaScript, čehož lze dobře využít při vytváření interaktivních grafů, animací a vizualizací. [12, 13]

Škálovatelná vektorová grafika pracuje *retained mode*. To znamená, že objekty, které jsou vykreslovány, jsou uchovávány vykreslovačem grafiky. Pozice na obrazovce

a průhlednost se pak definuje v kódu programu. Díky tomu, že SVG pracuje v *retained mode*, jsou na webové stránce objekty SVG dostupné přímo jako elementy DOM. Velkou výhodou tohoto přístupu je, že lze interagovat s každým objektem zvlášť přímo přes DOM. Možným problémem při práci s SVG na webu je však výkon prohlížeče. Existuje zde totiž přímá závislost mezi počtem elementů v SVG a výkonem prohlížeče. Pokud tedy bude SVG obsahovat tisíce interaktivních prvků, a tedy i DOM bude obsahovat tisíce prvků, velmi pravděpodobně to bude mít podstatný vliv na výkon celé aplikace v prohlížeči. SVG se proto hodí spíše pro vykreslování grafiky, která neobsahuje velké množství vykreslovaných prvků. [12]

2.4.2 Rozhraní HTML5 Canvas

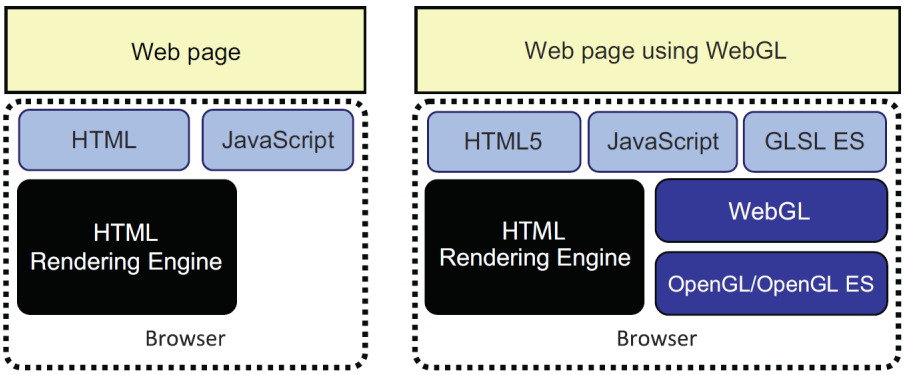
Další z možností pro zobrazování dvourozměrné grafiky na webu je HTML5 Canvas. Jedná se o novější technologii, než je SVG. Canvas pracuje s bitmapovou rastrovou grafikou, která se vykresluje do HTML elementu `<canvas>`. Vykreslování do zmíněného elementu probíhá pomocí Canvas API, které je voláno z kódu jazyka JavaScript. Oproti škálovatelné vektorové grafice SVG pracující v *retained mode* (kapitola 2.4.1) funguje Canvas odlišně. Pracuje totiž v *immediate mode*, který se vyznačuje tím, že Canvas bitmapa na obrazovce je na každém snímku kompletně překreslena pomocí volání Canvas API. Další vlastností Canvas je, že neobsahuje grafické objekty, ale pouze instrukce, co má vykreslit pro každý snímek zvlášť. Samotný Canvas element je přístupný jako DOM element webového prohlížeče přes *Canvas 2D context*, ale grafické elementy, které vykresluje, nejsou přes DOM dostupné tak, jako tomu bylo u SVG. Pokud je žádoucí s objekty na obrazovce interagovat, je potřeba využít nepřímé metody pro interakci. Jako modelová situace může sloužit například chvíle, kdy programátor chce zjistit, na jaký objekt bylo klinkuto myší. V tomto případě může využít pozice $\{x, y\}$ kliknutí na obrazovku a zpětně spočítat, na který objekt v Canvas rastru bylo kliknuto. [13]

Pomocí API rozhraní *Canvas 2D context* a JavaScript programu je možné vykreslovat geometrické útvary, text, obrázky, čáry, křivky a další, a to do daného bodu v elementu Canvas. Zároveň je možné nastavit jejich orientaci, orámování, barvu, gradient a mnoho dalšího. Jelikož Canvas nevytváří pro každý prvek DOM reprezentaci, dosahuje vyššího výkonu při vykreslování velkého počtů objektů. Na úkor toho je však v případě použití technologie Canvas složitější implementace interakce s objekty. HTML5 Canvas je aktuálně podporována všemi hlavními moderními prohlížeči Chrome, Firefox, Safari a Microsoft Edge. [13]

2.4.3 Rozhraní WebGL

WebGL je technologie používaná k vykreslování, zobrazování a vytváření grafiky na webu ve webovém prohlížecí. Oproti předchozím metodám, je tato metoda určená pro vykreslování velmi složité grafiky včetně interakce s touto grafikou. WebGL bylo vytvořeno a je primárně určeno pro vytváření složité trojrozměrné bitmapové rastrové grafiky, ale lze jej využít i pro složitou 2D grafiku. Technologie WebGL je odvozena z technologie pro zobrazování 3D grafiky nazvané OpenGL ES určené pro embeded systémy. Vykreslování grafiky probíhá stejně jako v případě technologie Canvas do elementu `<canvas>`. WebGL však rozšiřuje jeho možnosti o trojrozměrnou grafiku a grafikou akceleraci. Technologie WebGL je podporovaná všemi hlavními prohlížeči, tedy Chrome, Firefox, Safari a Microsoft Edge. Podporováno je také mnoho mobilních zařízení, a je tak možné využít grafické akcelerace i v operačním sytému Android a iOS. [14, 15]

Výše popsaná vykreslování pomocí technologií SVG a Canvas využívají takzvaného softwarového vykreslování, zatímco WebGL je založeno na hardwarovém vykreslování. Softwarové vykreslování se provádí za pomocí procesoru (CPU). Hardwarové vykreslování je možné využít v těch případech, kdy je k dispozici grafický procesor (GPU), který provádí grafické výpočty. Toto vykreslování umožňuje dosahovat mnohem lepšího výkonu při vykreslování grafiky ve srovnání se softwarovým vykreslováním. [15]



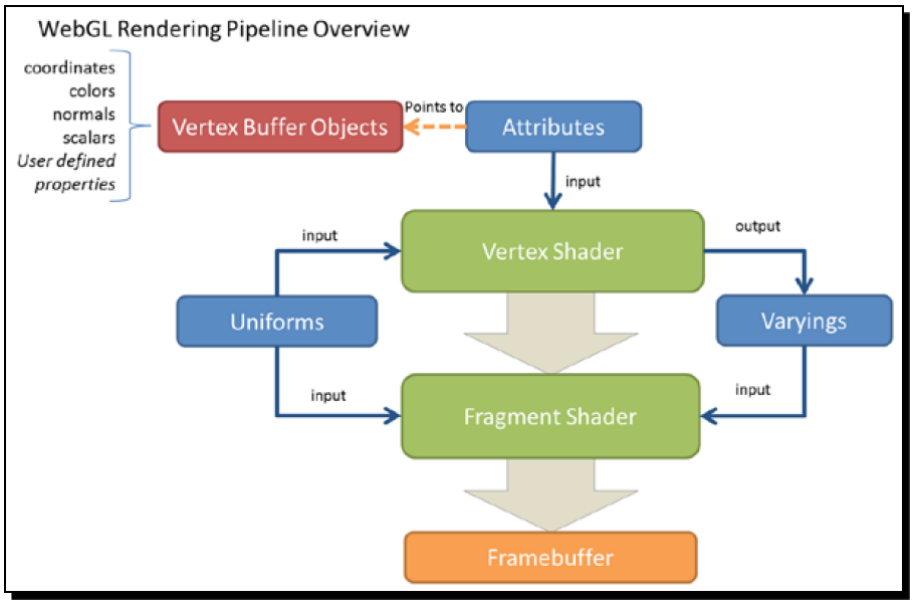
Obr. 2.2: Architektura dynamického webu využívající WebGL (vpravo) a webu bez WebGL (vlevo). [14]

Technologie WebGL je vlastně webové API pro práci s trojrozměrnou grafikou. Struktura webové aplikace využívající WebGL je znázorněna na obrázku 2.2 vpravo. V levé části obrázku je pro srovnání zobrazeno i schéma běžné webové aplikace, která nevyužívá WebGL. K vytvoření aplikace využívající WebGL pro vykreslování je potřeba znát značkovací jazyk HTML, dále programovací jazyk JavaScript a shader

jazyk GLSL ES. Shader programy umožňují implementovat složité vizuální efekty za pomocí jazyka GLSL, jenž je založený na shader jazyku z OpenGL. Jelikož je ale GLSL ES napsán v jazyce JavaScript, pro vytváření programů je potřeba pouze soubory jazyka HTML a JavaScript. [14, 15]

Schéma fungování vykreslování pomocí technologie WebGL

Celé schéma vykreslování za pomocí technologie WebGL je znázorněno v diagramu na obrázku 2.3. V následujících odrážkách jsou popsány jednotlivé části schématu vykreslování grafiky pomocí technologie WebGL.



Obr. 2.3: Schéma vykreslování ve WebGL. [15]

- **VBO (angl. *Vertex Buffer Objects*)** obsahují data s vlastnostmi vrcholů, která jsou potřeba pro zobrazení požadované geometrie pomocí WebGL. Jedná se většinou o souřadnice vrcholů, barvy nebo souřadnice textur v obraze. Poloha vrcholů se zapisuje pomocí souřadnic x, y a z. [15]
- **IBO (angl. *Index Buffer Objects*)** obsahuje indexy vrcholů v dané scéně, které poskytují informaci, jak propojit vrcholy ve VBO tak, aby vznikl povrch objektů. [15]
- **Vrcholový shader** (angl. *vertex shader*) je typ shader programu. Jak již bylo zmíněno shader programy se využívají pro WebGL vykreslování a různé vizuální efekty. Shader programy jsou načítány z JavaScript programu a uloženy do WebGL pro vykreslení. WebGL využívá dva typy shader programů. Prvním z nich je vrcholový shader, sloužící k manipulaci s vlastnostmi vrcholů (bodů

v 2D či 3D prostoru) vykreslovaných objektů. Danými vlastnostmi může být například barva, poloha či další atributy. Vrcholový shader je volán pro každý vrchol scény. [14, 15]

- **Fragmentový shader** (angl. *fragment shader*) je druhým typem shader programu. Slouží ke zpracování fragmentů jako je například světlo, nebo barva určitého místa povrchu objektu v zobrazované scéně. Je třeba si totiž uvědomit, že pokud pracujeme s 3D grafikou, musí být brán ohled na to, kde budou umístěny zdroje světla osvětlující scénu, či jaká je perspektiva kamery zachycující scénu. V tom případě musí být každému bodu plochy (fragmentu) přiřazená barva, a právě za vypočet barvy je odpovědný fragmentový shader. [14, 15]
- **Framebuffer** je dvojrozměrná vyrovnávací paměť, která obsahuje fragmenty, které již zpracoval fragment shader. Po tom, co jsou zpracovány všechny fragmenty a již byly nahrány do framebufferu, je vytvořen výsledný 2D obraz, který je zobrazen na obrazovce. [15]
- **Atributy** jsou vstupní proměnné jednotlivých vrcholů při volání vrcholového shaderu. Výše již bylo zmíněno, že se jedná o souřadnice v prostoru, barvu a další. Dané atributy jsou odlišné pro každý vrchol požadované scény. [15]
- **Uniforms** jsou vstupní proměnné jak vrcholového shaderu tak fragmentového shaderu. Oproti atributům jsou však uniforms pro jeden vykreslovací cyklus konstantní a stejné pro všechny vrcholy a fragmenty. Příkladem takové proměnné může být pozice osvětlení scény. [15]
- **Varyings** zajišťují předávaní dat z vrcholového shaderu do fragmentového shaderu. [15]

Vykreslování za pomocí technologie WebGL pak probíhá v souladu s obrázkem 2.3 v následujících krocích. Nejdříve je potřeba definovat geometrii vykreslované grafiky v jazyce JavaScript pomocí polí. Poté následuje druhý krok, kterým je vytvoření WebGL VBO bufferu obsahujícího vrcholové souřadnice z JavaScript polí. Třetím krokem je nasměrování atributů vrcholového shaderu na VBO buffer. Dále jsou za pomocí IBO bufferu, vrcholového a fragmentového shaderu vytvořeny a uloženy povrchy do framebufferu. Nakonec, když jsou všechny fragmenty zpracovány, je vykreslen finální 2D obraz do HTML elementu `<canvas>`. [15]

Knihovna PixiJs

PixiJS je vykreslovací engine určený pro dvourozměrnou grafiku. Jedná se o extrémně rychlý engine umožňující zobrazování, animování a interaktivitu s vykreslovanou grafikou. Za využití jazyka JavaScript a značkovacího jazyka HTML lze pomocí vysokoúrovňového PixiJS API jednoduše tvořit složitou 2D grafiku. PixiJS

umožňuje vytvářet stromovou hierarchii vykreslovaných objektů, dále také přímou interakci s objekty pomoci myši, a to vše za využití velmi malého množství zdrojového kódu. [16]

Pro vykreslování využívá PixiJS technologii WebGL.Právě díky využití WebGL, a také proto, že vykreslování probíhá na grafické kartě, je PixiJS opravdu velmi rychlé a dokáže pracovat se složitou 2D grafikou. Kromě WebGL umožňuje tato knihovna využít i vykreslování pomocí Canvas API. Tato možnost se však vyznačuje mnohem menším výkonem a nepodporuje všechny funkcionality, například přidávaní obrazových filtrů[2] pro vykreslování. Výhodou využití Canvas API namísto WebGL je naopak možnost vykreslit scénu i na zařízeních, které neobsahují grafický procesor, nebo nemají webový prohlížeč podporující WebGL. Tím je možné zajistit kompatibilitu i se staršími zařízeními, avšak samozřejmě se značně sníženým vykreslovacím výkonem. [16]

Vykreslovací knihovna PixiJS je založená na modulárním principu, kdy za různé typy úloh zodpovídají různé komponenty. Může se jednat například o úlohy vytváření obsahu, aktualizace scény nebo vykreslování scény. Výhoda tohoto rozdělení spočívá v možnosti udržovat čistší kód při vývoji. V následujících odrážkách jsou uvedeny komponenty, ze kterých se skládá vykreslovací program PixiJS. [17]

- **Renderer** zajišťuje vykreslování scény. Zde se může jednat buďto o WebGL renderer, který je výchozí, nebo o Canvas API renderer. [17]
- **Container** je hlavním zobrazovacím objektem PixiJS. Container obsahuje strom objektů, které se mají zobrazit na obrazovce. Objekty mohou být například základní geometrické útvary, jako jsou obdélníky, kruhy, trojúhelníky, mnohoúhelníky, lomené čáry a podobně. Jinými objekty jsou textury (obrázky a další) a text. [17]
- **Loader** představuje nástroj pro práci s asynchronním načítáním zdrojových souborů pro vykreslování. Většinou se jedná o obrázky nebo hudební soubory. [17]
- **Ticker** slouží k poskytování periodického volání funkcí založeného na čase. Často se používá k volání funkce aktualizující logiku, pozici objektů, velikost objektů atd. pro každý snímek vykreslený na obrazovku. [17]
- **Application** je komponenta seskupující Renderer, Loader a Ticker do jednoho objektu. [17]

[2]Příkladem filtru může být například barevný filtr, filtr pro rozmazaní obrazu (angl. *blur*) nebo filtr invertující barvy.

- **Interaction** zajišťuje interakci se scénou. Díky tomu je možné nad objekty zpracovávat události jako je přejetí myší přes objekt, kliknutí na objekt, držení objektu a mnoho dalších událostí. [17]
- **Accessibility** poskytuje nástroje pro další možnosti ovládání a interakce se scénou. Jde například o možnost ovládání scény pomocí klávesnice. [17]

2.5 Rozhraní REST API

API (angl. *Application Programming Interface*) je rozhraní, které umožňuje aplikacím vzájemně komunikovat. REST (angl. *Representational State Transfer*) je architektonický styl pro vytváření a organizaci distribuovaných systémů. REST tedy není standard, návod nebo definice pravidel, která je třeba dodržovat. Dále budou v této části práce popisovány webové služby s rozhraním API odpovídajícím architektonickému stylu REST nazývané REST API. [18, 19]

Před zavedením REST rozhraní byl k implementaci API nejčastěji vývojáři využíván protokol SOAP (angl. *Simple Object Access Protocol*). Pro samotné volání API byl potřeba ručně psaný XML dokument, jehož tělo obsahovalo RMI (angl. *Remote Procedure Call*), a odeslání následně proběhlo pomocí protokolu SOAP. [20]

Pro vysvětlení, SOAP je, narozdíl od REST, definicí standardního komunikačního protokolu pro předávání zpráv ve formátu XML. Implementace protokolu SOAP je ve srovnání s REST rozhraním složitější. Avšak pro popis rozhraní REST neexistuje sada pravidel, takže uživatelé musí rozumět předávanému kontextu a obsahu. U protokolu SOAP je předáván i popis. [20]

V roce 2000 se skupina vývojářů rozhodla vytvořit standard pro komunikaci mezi jakýmikoliv servery, načež Roy Fielding ve své disertační práci definoval REST. V roce 2002 začal REST API používat Amazon a v roce 2004 Flickr. Během roku 2006 se pak přidal i Twitter a Facebook, a navíc Amazon jej začal využívat pro své cloudové služby. To vše vedlo k obrovskému zájmu o využívaní rozhraní REST API mezi vývojáři. [20]

Pro shrnutí jsou zde uvedeny hlavní myšlenky REST API. První z nich je výkonnost, která říká, že komunikace pomocí API by měla být, pokud možno, efektivní a jednoduchá. Dalším důležitým aspektem je škálovatelnost a jednoduchost rozhraní. Mělo by tedy být možné aplikaci snadno škálovat a interakce mezi aplikacemi využívajícími toto rozhraní by měla být jednoduchá. Dále lze zmínit modifikovatelnost. Komponenty API navrženého za pomocí architektury REST by měly být nezávislé na sobě a snadno modifikovatelné. Poslední zde zmíněnou myšlenkou REST rozhraní je přenositelnost. Ta sděluje, že REST by měl být technologicky a jazykově nezávislý, a proto by jej mělo být možné implementovat za pomocí jakékoliv technologie. [18]

Standardních metody protokolu HTTP

Webové služby REST jsou založeny na zdrojích, ke kterým se přistupuje pomocí standardních metod protokolu HTTP (angl. *HyperText Transfer Protocol*). Principiálně se jedná o komunikaci postavenou na architektuře klient - server. Základem REST API jsou základní CRUD (angl. *Create, Read, Update, Delete*) metody. Ty nepoužívanější jsou uvedeny v následujících odrážkách. [18, 19]

- GET - Účelem je získání reprezentace požadovaného zdroje v určité reprezentační formě. Metoda GET by nikdy neměla způsobovat změnu reprezentace zdroje. [19]
- POST - Tato metoda většinou vytváří nový zdroj. V těle metody je navrhovaná reprezentace nového zbroje, která má být přidána do datového úložiště na serveru. Dále je metoda POST aplikována všude tam, kde nedochází k získávání, ukládání nebo mazání zdrojů. Používá se tady k provádění všech operací, kterým nelze přiřadit ostatní HTTP metody. [19]
- PUT - Metoda je využívána hlavně pro aktualizaci nebo nahrazení již existující reprezentace zdroje. Tělo metody obsahuje reprezentaci zdroje, kterým chce uživatel nahradit na serveru zdroj již existující, nebo reprezentaci obsahující změněné části zdroje. [19]
- DELETE - Metoda Delete slouží k odstranění reprezentace zdroje na serveru. Poté již nemůže být nalezena pomocí metody GET či HEAD. Pokud je odstraněním myšleno pouze označení objektu jako smazaného, reálně však ze serveru smazán není, je vhodné využít metodu POST místo DELETE. [19]
- HEAD - Tato metoda je velmi podobná metodě GET, avšak odpověď serveru neobsahuje tělo s reprezentací požadovaného zdroje. Vrácena ze serveru je pouze hlavička[3] odpovědi. [19]

Stavové kódy HTTP protokolu

Důležitou součásti protokolu HTTP jsou stavové kódy. Každá metoda REST API vrací stavový kód, který informuje o výsledku požadavku na API. Kódy je možné rozdělit do pěti skupin. Kódy začínající jedničkou jsou pouze informační. Dvojkou začínají kódy, které informují, že požadavek byl úspěšně přijat. Pokud kód začíná trojkou, znamená to, že klient musí provést určitou akci, aby bylo možné požadavek uskutečnit. Čtyřka na začátku informuje o chybách vzniklých vinou klienta. Poslední kategorií jsou kódy začínající pětkou označující chyby, za které je zodpovědný sever. Některé základní stavové kódy jsou uvedeny v odrážkách níže. [19]

[3]Hlavička ve zprávě HTTP obsahuje různá metadata. Příkladem můžou být informace o požadovaném prostředku nebo informace o reprezentaci dat v těle zprávy. [19]

- **200 OK** - Nespecifický úspěšný požadavek. [19]
- **201 Created** - Informace, že byl úspěšně vytvořen nový zdroj. [19]
- **401 Unauthorized** - Klient se pokusil pracovat s prostředkem, aniž by zadal příslušné oprávnění. [19]
- **403 Forbidden** - Klient se snaží pracovat se zdrojem, který je mimo jeho oprávnění. [19]
- **404 NotFound** - Požadavek klienta nemůže být přiřazen k žádnému zdroji na serveru. [19]
- **409 Conflict** - Oznamuje že se klient pokusil vložit zdroj do nepovoleného, nebo nemožného stavu. [19]
- **500 Internal Server Error** - Jedná se o chybu, která není na straně klienta a je za ni odpovědný server. Často je tento kód vrácen automaticky, pokud je na serveru vyvolána výjimka. [19]

Reprezentace dat pomocí JSON

JSON (angl. *JavaScript Object Notation*) je textový standard reprezentace dat určený pro výměnu informací, který je čitelný jak strojově, tak člověkem. Jeho výhodu je nezávislost na programovacím jazyku a velikostní úspornost oproti reprezentaci XML, což jej činí rychlejší. Mezi nevýhody lze zařadit skutečnost, že do něj nelze zapisovat komentáře, dále není tak úsporný jako binární reprezentace a nelze u něj vynucovat specifické schéma, jak je tomu u XML. JSON umožnuje reprezentovat tyto typy: text, čísla, objekty, pole, pravdivostní hodnoty a null pro prázdné hodnoty. [18]

Následující výpis s číslem 2.3 ukazuje zápis totožného objektu v reprezentaci XML a JSON.

Výpis 2.3: Ukázka definice objektu v XML a JSON.

```
1    <car> // Zápis pomocí XML
2        <brand>Audi<brand>
3        <model>A6</model>
4        <price>50000</price>
5    </car>
6
7    { // Zápis pomocí JSON
8        "brand" : "Audi",
9        "model" : "A6",
10       "price" : 50000
11   }
```

Reprezentace dat pomocí XML

XML (angl. *Extensible Markup Language*) je textový značkovací jazyk, který je velmi často využíván pro výměnu dat. Zápis v XML dat je velmi podobný formátu HTML, jelikož využívá značek < a >. Formát je hierarchický a velmi dobře strojově i lidmi čitelný. Umožňuje totiž i zápis komentářů, díky kterým se mohou lidé v datech lépe orientovat. Dále je možné využít schéma, které může definovat, jaké atributy má XML obsahovat nebo například jaká je výchozí hodnota daného atributu. Při strojovém čtení je však XML náročnější na výkon než formát JSON, a to proto, že jeho velikost v paměti je větší. [18]

2.6 Programovací jazyk Java

Java je vysokoúrovňový, objektově orientovaný, striktně typový programovací jazyk. První stabilní verze jazyka Java byla vydána již v roce 1996 a do dnes se jedná o velice populární jazyk, který hojně využívají jak malé, tak ty největší společnosti po celém světě. Programy napsané v jazyce Java potřebují pro svůj běh kromě operačního sytému také JVM (angl. *Java Virtual Machine*). JVM je abstraktní výpočetní jednotka, která převádí kód z jazyka Java do strojového jazyka, a ten následně spouští. JVM tedy slouží ke spuštění kódu napsaného v Javě. Velkou výhodou toho přístupu je, že zmíněná vypočtení jednotka vytváří prostředí, které je nezávislé na platformě. Tudíž je možné program v jazyce Java spouštět v operačním systému Windows, Linux, macOS a mnoha dalších. [21]

Java ve srovnání s jinými vysokoúrovňovými jazyky, které kompilují zdrojový kód přímo do strojového kódu, využívá odlišného přístupu. JVM si lze představit jako procesor jazyka Java, která zdrojový kód interpretuje jako posloupnost akcí na libovolném procesoru bez ohledu na operační systém. Jak již bylo zmíněno, velkou výhodou je u tohoto přístupu nezávislost na prostředí, kde program běží. Jako nevýhodu lze vnímat to, že kvůli interpretaci je výkon nižší než u programů v jazycích kompilovaných přímo do strojového kódu (C, C++). [21]

Syntaxe jazyka Java vychází z jazyků C a C++, a je jim proto v tomto ohledu velmi podobná. Verze jazyka Java od společnosti Oracle se aktuálně číslují od čísla 7. Verze 7, 8, 11 a 17 jsou verze s plánovanou dlouhodobou podporou, ty je tedy nejvhodnější využívat na dlouhodobé projekty. [21]

Pro vývoj v jazyce Java je nutné využít JDK (angl. *Java Development Kit*), jenž obsahuje nástroje potřebné k vývoji. Primárně se jedná o JRE (angl. *Java Runtime Environment*), který obsahuje JVM a prostředky nutné ke spuštění Java programu, dále také interpreter, kompilátor a další nástroje potřebné pro vývoj. [22]

Struktura programu v jazyce Java

Všechny soubory se zdrojovým kódem v jazyce Java mají příponu *.java*. Tyto soubory jsou dále uspořádány do balíčků (angl. *packages*). Balíčky jsou definovány hierarchií adresářů a většinou logicky seskupují třídy s podobným účelem. Příkladem může být hierarchie adresářů znázorněna následující strukturou.

```
com
└── onsemi
    └── mojeaplikace
        ├── MyReader.java
        └── MyUtils.java
```

Balíček se zmíněnou strukturou obsahující soubory `MyReader.java` a `MyUtils.java` má název `com.onsemi.mojeaplikace`. Při kompilaci jsou balíčky baleny do jednoho nebo více JAR (angl. *Java Archives*) souborů. Další úrovní je knihovna, která shlukuje jeden či více JAR souborů. U větších aplikací však můžou vznikat problémy. Balíčky rozptýlené ve více JAR souborech a tím vznikající závislosti mezi JAR soubory, způsobují, že je v programu využíváno více verzí stejné třídy nebo že chybí závislosti na jiné JAR soubory. Popsané problémy řeší funkcionalita uvedená v Javě verze 9, která je nazvaná moduly. Ty sdružují a zapouzdřují balíčky. V modulech je možné explicitně definovat, jaké balíčky exportují a jaké vyžadují. Tato funkcionalita zlepšuje přehlednost pro vývojáře ve velkých aplikacích. [22]

Správa paměti při běhu Java programu

Jelikož při běhu programu může být opakovaně vytvářeno mnoho objektů, je třeba řešit správu paměti, zejména uvolňování paměti, kterou si rezervují nevyužívané objekty. O to se stará GC (angl. *garbage collection*), který je součásti JVM. V jazycích jako je C a C++ nic takového jako GC neexistuje a za správu paměti (alokaci a dealokaci) je zodpovědný vývojář aplikace. GC však pracuje automaticky a zajišťuje, aby paměť objektů, jejichž reference již nejsou využívány, byla uvolněna. Automatická správa paměti je sice pohodlná pro vývojáře a zjednodušuje kód vyvíjeného programu, avšak je potřeba myslet na její správné použití. [21]

2.7 Spring Framework

Spring je framework určený pro programovací jazyk Java, který je primárně určen pro vývoj podnikových (angl. *enterprise*) aplikaci. Tento framework integruje mnoho nástrojů, například Servlet API využívající se pro vývoj webových aplikací, WebSocket API, zpracování formátu JSON, nástroje pro vytváření REST API a další. Kromě zmíněných funkcionalit zajišťuje Spring také možnost využívat vkládání závislostí DI (angl. *dependency injection*) v programu.

Framework je rozdělen na několik částí, lze zmínit Spring Security pro zajištění zabezpečení aplikace, Spring Data k práci s databázemi či například Spring Batch k vyváření aplikací pro dávkové zpracování. Jedná se o jeden z nejpoužívanějších frameworků pro jazyk Java a využívá jej velká komunita vývojářů. [23]

Spring Boot

Dalo by se říci, že Spring Boot je Spring Framework, který značně usnadňuje způsob vývoje Spring aplikací. Ve Spring Boot aplikaci může být například do webové aplikace integrován Tomcat nebo Netty web server, díky čemuž je možné aplikaci rovnou spustit bez nutnosti použití vlastního webového serveru. Vývoj a nasazení požadované aplikace je poté mnohem rychlejší. Spring boot také umožňuje využít soubor `application.propersites`, který slouží ke konfiguraci aplikace, či profily, které umožňují spouštění aplikace v různých konfiguracích. Výhoda Spring Boot oproti Spring Frameworku, která je často zmiňována v literatuře, je automatická konfigurace. Automatická konfigurace označuje proces, který při spuštění Spring Boot aplikace rozhoduje o tom, jaká konfigurace Spring Boot by měla být použita, a to primárně na základě použitých anotací. Tato automatická konfigurace tedy usnadňuje práci vývojářům, neboť nemusí Spring konfigurovat, když to není nutné. [23, 24]

Pokud je potřeba pomocí Spring Boot aplikace vytvořit REST API, lze to učinit velmi jednoduše. Stačí vytvořit specifickou Java třídu se Spring anotacemi. Ukázka tvorby jednoduchého REST API pomocí Spring Boot je uvedena ve výpisu 2.4. Za pomocí anotace `@RestController` je specifikováno, že třída `MyController` bude zajišťovat REST API kontrolér s cestou „/api“ definovanou pomocí anotace `@RequestMapping("/api")`. Metoda `getTestResult()` je anotována pomocí `@GetMapping("/testResult")`, což definuje skutečnost, že po zavolaní API s relativní adresou „/api/testResult“ bude vrácen string „My result string“. [25]

Výpis 2.4: Ukázka definice REST API ve Spring Boot.

```
1  import org.springframework.web.bind.annotation.*;
2  @RestController
3  @RequestMapping("/api")
4  public class MyController {
5
6      @GetMapping("/testResult")
7      public String getTestResult() {
8          return "My-result-string";
9      }
10 }
```

3 Stávající řešení dostupné u zadavatele

Cílem této kapitoly je seznámit čtenáře s řešeními, která aktuálně existují u zadavatele a mají souvislost s aplikací, jejímž vývojem se zabývá tato diplomová práce. Vyvíjená aplikace musí být schopna se všemi užívanými systémy komunikovat, a proto je třeba využívat standardy a postupy, které se ve společnosti onsemi běžně využívají. Dále jsou v kapitole popsány systémy sloužící k ukládání mapových souborů a poskytování metadat k *wafer mapám* (mapám křemíkových desek při výrobě čipů) z polovodičové výroby. Mimo to je zde popsán systém, který udržuje a spravuje uživatelská privilegia, oprávnění k různým akcím a seznam mapových repozitářů. Systémy shrnuté v této kapitole spravuje ve společnosti onsemi tým nazvaný UMR (angl. *Universal Map Repository*). Většina již existujících softwarových systémů zde využívá programovacího jazyka Java.

3.1 Stávající softwarová řešení pro správu map křemíkových desek

Ve společnosti onsemi existuje již mnoho hotových aplikací a řešení. Na daná řešení bude navazovat vývoj aplikace MapSpyWeb, jenž je popsán v následujících kapitolách této práce. Existující systémy usnadní vývoj aplikace, neboť implementují mnoho logiky, která bude moci být využita pro aplikaci MapSpyWeb.

Mapový repozitář MapVault

Prvním systémem, který zde bude popisován je mapový repozitář MapVault. Jedná se o aplikaci napsanou v jazyce Java spravující úložiště mapových souborů a databázi obsahující mapová metadata. Jedná se o hlavní zdroj dat pro aplikaci MapSpyWeb, kterou se zabývá tato diplomová práce. Základní funkcí, kterou aplikace bude umožňovat, je dotazování se na mapová metadata uložená v databázi. Dále je možné z MapVault stahovat mapové soubory či nahrávat nové mapové soubory zpět do úložiště.

Každé výrobní místo (angl. *site*) má svůj produkční (*PROD*) MapVault a většinou také MapVault pro zajištění jakosti (*QA*), kromě toho existují i instance MapVault pro vývojové účely (*DEV*). Ve společnosti onsemi tedy existuje mnoho instancí tohoto systému. Napříč instancemi je nasazeno mnoho verzí aplikace MapVault, a proto je třeba myslet na vzájemnou kompatibilitu vyvíjené aplikace s tímto systémem. MapVault využívá jako úložiště mapových souborů NetApp cloud. Mapové soubory jsou v tomto cloudovém úložišti uloženy jako soubory ve formátu XML. Zmíněné XML soubory obsahují kromě mapových podkladů i množství důležitých

doplňujících informací a metadat o dané mapě. Jako rozhraní pro komunikaci s ostatními systémy používá MapVault dvě API, prvním z nich je SOAP API a druhým REST API. SOAP API je možné využít pro komunikaci s všemi instancemi MapVault, zatímco REST API bylo přidáno až v pozdější verzi tohoto systému, a proto existují i instance, jejichž verze REST API ještě neobsahuje.

Do repozitářů MapVault jsou mapové soubory a jejich metadata nahrávána automaticky během výroby polovodičových součástek, či manuálně zaměstnanci. Během procesu výroby jsou jednotlivé mapy vytvářeny například pomocí elektrického měření zajištěného přístroji *prober* a *tester* (viz kapitola 1.3).

Systémy UMR Services a MapSpyAccessPoint

UMR Services a MapSpyAccessPoint jsou další aplikace napsány v jazyce Java, jež jsou součástí UMR. Aplikace MapSpyAccessPoint poskytuje přístup k databázi UMR, která udržuje metadata ze všech map křemíkových desek napříč všemi instancemi MapVault z určitého okruhu (*PROD*, *QA* a *DEV*). Jedná se o úložiště, které sdružuje metadata napříč výrobními místy, ale neobsahuje mapové soubory. Mapové soubory jsou uloženy vždy pouze v lokálních MapVault repozitářích. Databáze UMR také udržuje informace o všech existujících systémech MapVault z odpovídajícího okruhu instancí. MapSpyAccessPoint, stejně jako MapVault, obsahuje dvě API, a to SOAP a REST. Ve srovnání s MapVault však MapSpyAccessPoint REST API umožňuje pouze získávat mapové soubory, ale neumožňuje soubory nahrávat.

Synchronizaci metadat mezi systémy MapVault a databází UMR zajišťuje aplikace UMR Services. Díky ní jsou do globální databáze synchronizována data ze všech bežících systémů MapVault. Jednou z dalších funkcí, kterou tato aplikace zabezpečuje, je přenos mapových metadat a souborů mezi systémy MapVault. Ta je nejčastěji potřebná z důvodu fyzického transportu křemíkové desky do jiné výrobní oblasti.

Narozdíl od aplikace MapVault zde běží jen tři hlavní instance této aplikace. Jedna je produkční pro výrobu (*PROD*), druhá slouží k zajištění jakosti (*QA*) a třetí pro vývojové účely (*DEV*). Další funkcionalitou aplikace MapSpyAccessPoint je poskytování privilegii. Accesspoint API poskytuje přístup k privilegiím pro všechny zaměstnance. Tato privilegia obsahují specifické informace, k jakým akcím má daný uživatel přístup a nad jakými daty, za jakých podmínek může uživatel definované akce provádět.

Nástroj WaferMapJr

WaferMapJr je softwarový nástroj vyvinutý společností onsemi. Jedná se o balíček v jazyce Java, který obsahuje nástroje pro práci s mapovými soubory. WaferMa-

pJr umožňuje načíst mapové soubory v jednom z mnoha podporovaných mapových formátů a následně s nimi pracovat. Nejběžnějším formátem mapových souborů využívaným ve společnosti onsemi je XML. Balíček poskytuje možnost načtení různých vrstev map a jejich metadat. Dále je díky němu možné editovat mapové vrstvy a metadata a poté je opět uložit do souboru. Nástroj umožňuje mnoho jiných funkcionalit jako například rotaci map nebo porovnaní s jinou mapou.

3.2 Aplikace MapSpy a MapEditor

Aplikace MapSpy je předchůdcem aplikace MapSpyWeb, jejímž vývojem se zabývá tato diplomová práce. MapSpy obsahuje možnost vyhledávání map pouze pomocí několika parametrů. Následně zajišťuje zobrazení metadat a vizualizací mapových souborů, jedná se ale o technologicky zastaralou aplikaci. Využívá jazyk Java 6 a k jejímu spuštění je také potřeba mít na každém zařízení nainstalovánu specifickou verzi JRE (angl. *Java Runtime Environment*). Instalace či aktualizace JRE však vyžaduje administrátorská oprávnění. Proto není jednoduché spustit aplikaci na libovolném firemním zařízení a je nutné o instalaci či aktualizaci JRE nejprve žádat administrátora. K tvorbě UI je využito stárnoucí technologie Swing v kombinaci s NetBeans IDE k vytváření GUI. To vše způsobuje vysoké požadavky na správu aplikace, jak z pohledu jejího vývoje, tak z pohledu distribuce a udržování její funkčnosti. Samotné spuštění aplikace trvá poměrně dlouhou dobu a také její výkon není dostatečný. Při větších velikostech vizualizovaných map se aplikace zasekává.

Druhou aplikací je MapEditor sloužící k editaci map. Jedná se o nezávislou aplikaci založenou na stejných technologiích jako původní MapSpy popsaný výše, s tím, že se u ní objevují i podobné problémy. Pokud chce uživatel například editovat mapu, kterou vyhledal v aplikaci MapSpy, ale nemá editor spuštěný, trvá velmi dlouhou dobu, než se samotný editor spustí.

Distribuce aplikací MapSpy a MapEditor uživatelům

Velkým nedostatkem původních aplikací je skutečnost, že v každém výrobním místě běží specifická verze repozitáře MapVault a k tomu specifická kompatibilní verze aplikací MapSpy a MapEditor. Ve společnosti onsemi aktuálně existuje mnoho výrobních míst a je třeba brát v úvahu, že jejich množství se bude pravděpodobně do budoucna zvyšovat. Distribuce aplikací uživatelům zde probíhá tak, že existuje centrální stránka UMR, kde je ke stažení soubor zajišťující instalaci aplikace určité verze pro specifický repozitář. Toto řešení je však velmi náročné na správu.

4 Požadavky na nový anotační a vizualizační systém

V této části diplomové práce jsou definovány požadavky zadavatele na nový anotační a vizualizační systém. Tyto požadavky byly definovány na základě odborné diskuze s konzultantem z firmy onsemi v Rožnově pod Radhoštěm a analýzy požadavků na anotační a vizualizační systém. Vzhledem k požadavkům popsaným v této kapitole je pak v dalších částech práce proveden teoretický návrh schématu aplikace a také jeho praktická realizace. Cílem je tedy vytvořit anotační a vizualizační systém zaměřující se na data získaná ve výrobě čipů.

Základním požadavkem vyvíjeného systému s názvem MapSpyWeb je zajištění možnosti jeho využívání operátorem nebo inženýrem výroby ve webovém prohlížeči jak na svém přenosném počítači, tak na pracovní stanici přímo ve výrobě. Velká výhoda tohoto přístupu spočívá v tom, že již nebude nutné aplikaci instalovat přímo na zařízení. Zmíněná aplikace by také měla být výkonnější, a umožnit tak uživatelům pracovat svižněji, než tomu bylo u aplikací původních. Dalším z důležitých aspektů je kompatibilita řešení s technologiemi a standardy, které se využívají ve společnosti onsemi. Dále v kapitole jsou požadavky popsány detailněji.

4.1 Uživatelské rozhraní aplikace MapSpyWeb

Velmi důležitým prvkem nové aplikace je uživatelské rozhraní. Cílem je vytvořit, pokud možno, takovou aplikaci, která se bude uživatelsky přívětivá a umožní zrychlení a zefektivnění práce. Proto by mělo být uživatelské prostředí co nejintuitivnější a přizpůsobitelné specifickým potřebám daného pracovníka. Aplikace by měla být pohodlně ovládatelná za pomocí klávesnice a myši. Do budoucna je potřeba počítat i s faktem, že MapSpyWeb začnou využívat i operátoři na tabletech přímo ve výrobě. Již při návrhu je tedy nutno brát tuto možnost v úvahu, aby v budoucnu nenastal problém v přidání podpory dotykového ovládání. Co se týče zobrazování webu, nejmenší požadované rozlišení, při kterém musí být celá aplikace správně vykreslena, je 1280x720 pixelů.

4.1.1 Vyhledávání map křemíkových desek

Primární operací, kterou by aplikace měla umožňovat, je vyhledávání. Základem by mělo být hledání map pomocí různých parametrů (metadat), které jsou jednotlivým mapám přiřazeny. Je potřeba, aby vyhledávání bylo maximálně uzpůsobeno

uživatelům pro snadné a intuitivní využití. Měla by zde existovat také možnost pokročilejšího vyhledávání, kde si uživatel může určit, na základě kterých metadat chce hledat, a výsledek tak upřesnit. Pokud by bylo pro vyhledávaní použito více parametrů, měl by výsledek odpovídat logickému součinu zadaných parametrů. V případě, že pro jeden vyhledávací parametr bude zadáno více požadovaných hodnot, výsledek vyhledávání nad těmito hodnotami se bude řídit logickým součtem.

Data, se kterými aplikace pracuje a která je požadováno prohledávat, jsou uložena v různých repozitářích. Proto musí být uživatelům umožněno načítat data z repozitáře, který požadují.

4.1.2 Souhrn s výsledky vyhledávání

Dalším požadovaným prvkem je obrazovka obsahující souhrn výsledků vyhledávání. Mělo by se jednat o hlavní obrazovku, na které se budou uživateli zobrazovat všechny podstatné informace. Obrazovka by měla být schopna vhodně vizuálně zobrazit výsledky hledaní. Nalezené mapy budou seskupeny podle toho, do jakého lotu patří (lot je dávka v rámci dávkové výroby čipů). V daném seskupeném zobrazení map by měla být jasně viditelná i metadata dané dávky. Příkladem metadat může být název dávky, ID a mnoho dalších. Uživatel zde bude mít možnost vybrat jednu či více požadovaných dávek. Po tomto výběru se mu zobrazí přehled všech map patřících do vybraných dávek.

Přehled map křemíkových desek z vybraných dávek by měl mít podobnou strukturu jako zobrazení popsané v předchozím odstavci. Je nutné uživateli přehledně zobrazit metadata pro jednotlivé mapy a umožnit výběr jedné či více map. Přehled map pak má umožňovat několik funkcionalit. Měla by zde existovat možnost zkopírovat požadovaná metadata vybrané mapy do schránky, zobrazit mapu v editoru určeném pro editaci map a stažení souboru obsahujícího definici dané mapy. Kromě toho by měla existovat volba filtrace zobrazené mapy dle určitých metadat, například zobrazit pouze poslední verze map, odstraněné mapy nebo určité typy map. V přehledu map bude mít uživatel také možnost si zobrazit grafickou vizualizaci požadované mapy. Vizualizace mapy je jednou ze stěžejních částí aplikace. Detailnější popis požadavků na zmíněnou vizualizaci je popsán v kapitole 4.1.5.

Legenda vizualizace mapy je jiným podstatným prvkem, který aplikace musí obsahovat. Ta by měla být umístěna v blízkosti vizualizace mapy a jejím hlavním úkolem je informovat uživatele o tom, čemu na mapě odpovídají jednotlivé barevné odstíny a příznaky. Díky legendě bude uživatel schopen jednoduše rozlišit, které prvky mapy jsou *pass* a které *fail*. Uživatel bude mít také možnost rozklíčovat všechny ostatní parametry a informace, které vizualizace mapy poskytuje.

Dále by aplikace měla zahrnovat možnost zobrazit základní statistiku týkající se výsledků kvality výroby. U jednotlivých map tato statistika vyjadřuje procentuální množství jednotlivých typů prvků mapy a *pass* či *fail* prvků.

4.1.3 Přihlášení do aplikace MapSpyWeb

MapSpyWeb musí umožňovat přihlášení uživatele. Přihlášení je zde proto, že provádění značné části funkcí, které aplikace má obsahovat, by mělo být umožněno pouze konkrétním uživatelům. Akce, které mají být omezeny uživatelskými právy, jsou popsány v následujícím odstavci. Autentizaci uživatelů pro přihlašovaní zajistí firemní Active Directory, kde má každý zaměstnanec svůj profil. Kromě toho by však mělo být všem umožněno používat aplikaci bez přihlášení, a to v režimu hosta. Pro tento případ by ale mělo být povoleno pouze vyhledávání a zobrazení dat bez možnosti jakékoli editace, tudíž všechny akce, které vyžadují oprávnění, by měly být zakázány.

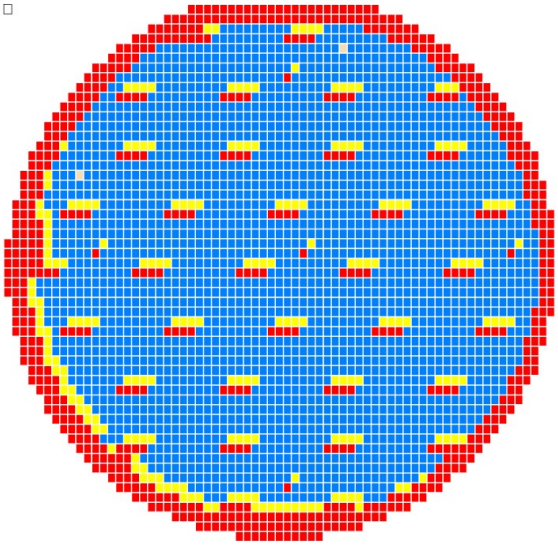
4.1.4 Autorizace uživatelských akcí

V předchozím odstavci bylo zmíněno, že některé akce, které bude možné v aplikaci provádět, by měly být povoleny pouze určitým uživatelům. Privilegia pro autorizaci k provádění akcí je možné získat z existujícího systému, který se nazývá *MapSpyAccesspoint*. Jednou z omezených akcí je nahrání mapy jako poslední verze. Dále je třeba omezit možnost odstraňování map a obnovování map, které byly v minulosti odstraněny. Stejně tak možnost editace map musí být umožněna pouze uživatelům, kteří mají přiděleno toto privilegium.

4.1.5 Vizualizace mapy křemíkové desky

Jednou z hlavních funkcí vyvíjené aplikace je vizualizace map křemíkových desek (takováto vizualizace mapy je vyobrazena na obrázku 4.1). Mapy obsahují data s informacemi o defektech součástek na křemíkové desce. Mapu si lze představit jako obdélníkovou či čtvercovou mřížku, kdy každý díl této mřížky představuje jeden prvek (často polovodičová součástka). Zobrazovací oblast vizualizace mapy by měla být navržena a implementována tak, aby byla co nejuniverzálnější a aby ji bylo možné případně použít i v jiných systémech. Dalším požadavkem je možnost rozšiřovat vizualizaci v budoucnu o vylepšené a nové funkce, a to ideálně s co nejmenším zásahem do původního kódu softwaru. Je také vhodné, aby byla vizualizace plně responzivní, což znamená, že zobrazovací oblast musí být schopna se přizpůsobit požadované velikosti a poměru stran zobrazení.

Celkově by u vizualizace měla být zajištěna dostatečná plynulost. Základním požadavkem je, aby navrhovaný systém zvládl vykreslit velké mapy křemíkových desek, které obsahují mřížku o velikosti až 1 000 x 1 000 prvků. Z toho vyplývá, že by mělo být možné vykreslit 1 000 000 objektů, se kterými bude uživatel schopen aktivně interagovat, a to vše při zachování dostatečného výkonu při vykreslovaní. Zmíněná velikost mapy je již dostatečně naddimenzovaná vzhledem k aktuálním maximálním požadavkům, tedy tak, aby aplikace poskytovala dostatečný výkon i do budoucna. Je však potřeba si uvědomit, že velikosti map stejně jako jejich mřížky se mohou do budoucna dále zvětšovat.



Obr. 4.1: Další ukázka vizualizace mapy křemíkové desky.

Interaktivita s mapou křemíkové desky

Oblast obsahující vizualizaci mapy musí umožňovat dynamické překreslovaní na základě aktuálně zvolené mapy. Pomocí pohybu myši musí být možné se po mapě plynule pohybovat. S ovládáním myší souvisí i další funkce, kterou je přiblížení a oddálení mapy (změna měřítka). Uživatel by tedy měl mít možnost mapu křemíkové desky přiblížit a oddálit, přičemž tento proces by měl být pro uživatele co nejplynulejší s minimální odezvou. Se změnou měřítka mapy souvisí funkce, která zajistí při prvním načtení mapy její zarovnání do zobrazovací oblasti tak, aby byla viditelná celá mapa a zabírala co největší část této oblasti.

Práce s prvky mapy křemíkové desky

Definováno je i několik požadavků pro skupiny prvků mapy (*bins*)[1], které vizualizace zobrazuje. Základní funkcionalitou by měla být možnost výběru dílčích prvků. To znamená, že uživatel bude schopen vybrat daný prvek mapy, například za pomocí kliknutí myší. Na základě zmíněného výběru by se měly uživateli zobrazit doplňující informace k vybranému prvku. Důležitým požadavkem také je, aby různé skupiny prvků byly zobrazovány ve vizualizaci jinou barvou. Uživatelé by měli mít možnost barvy změnit a přizpůsobit si je dle svých požadavků. S barvou prvků souvisí i další požadavek, kterým je mód vysokého kontrastu. Ten by měl zajistit zlepšení kontrastu pro dvě hlavní skupiny prvků označované jako dobré (*pass*) a vyřazené (*fail*). Mód vysokého kontrastu mapy značně napomůže vylepšit přehlednost map a usnadní jejich čitelnost pro uživatele zejména tam, kde se vyskytuje mnoho různých typů prvků, a tedy i odstínů barev.

4.1.6 Editor map křemíkových desek

Důležitým požadavkem aplikace je editor map křemíkových desek. Ten by měl zprostředkovat editaci mapových souborů. Editace by měla být uživatelsky co přívětivější a zároveň umožnit uživateli s využitím myši do mapy zakreslovat chybné prvky. Součástí by měla být i funkce gumy či funkce zpět a znovu pro kreslení. Vedle vizualizace umožnující editaci by měly být přehledně zobrazeny základní parametry a statistiky editované mapy.

4.2 Další požadavky na vyvíjený systém

Při vývoji aplikace je třeba předejít velmi vysokým požadavkům na správu různých verzí aplikace, které trápí stávající řešení aplikací MapSpy a MapEditor. Proto by se návrh a vývoj nové aplikace, na kterou je zaměřena tato práce, měl ubírat směrem, který zredukuje nutnost správy mnoha verzí a běžících instancí aplikace.

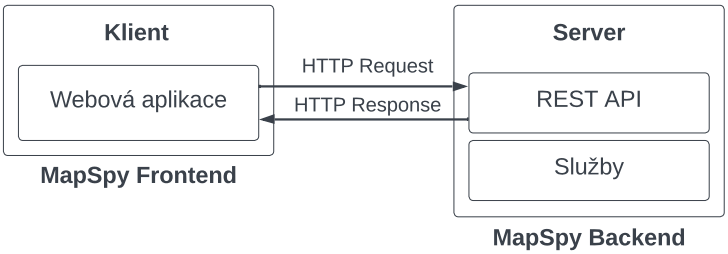
Dále je požadováno, aby uživatelské rozhraní vyvíjené aplikace bylo kompatibilní s operačním systémem Windows 10. Co se týče prohlížečů, zde je nutné zajistit, aby grafické uživatelské rozhraní aplikace podporovalo Google Chrome, Microsoft Edge a Mozilla Firefox v jejich aktuálních verzích. Tyto tři webové prohlížeče jsou totiž doporučovány společnosti onsemi a běžně využívány zaměstnanci.

[1]Prvky jsou jednotlivé elementy na mřížce mapy křemíkové desky. V reálu se většinou jedná o jednotlivé polovodičové součástky na křemíkové desce.

5 Teoretický návrh systému

Tato kapitola práce se zabývá teoretickým návrhem systému MapSpyWeb pro vizualizaci a anotaci dat z výroby polovodičů. Jednotlivé části kapitoly se zaměřují jak na frontend tak na backend navrhované aplikace. Celý návrh je založen na požadavcích, které jsou detailně popsány v kapitole 4. Dále návrh vychází z postupů a technologií popsaných v teoretickém úvodu práce (kapitola 2).

Již dříve v rámci práce bylo uvedeno v kapitole 3.1, že v současnosti existují ve společnosti onsemi dva systémy pro správu mapových souborů a jejich metadat. Jedná se o systémy MapVault a MapSpyAccessPoint. Teoretický návrh, který je popsán dále, byl proveden s přihlédnutím ke kompatibilitě s těmito dvěma systémy.



Obr. 5.1: Návrh architektury systému MapSpyWeb.

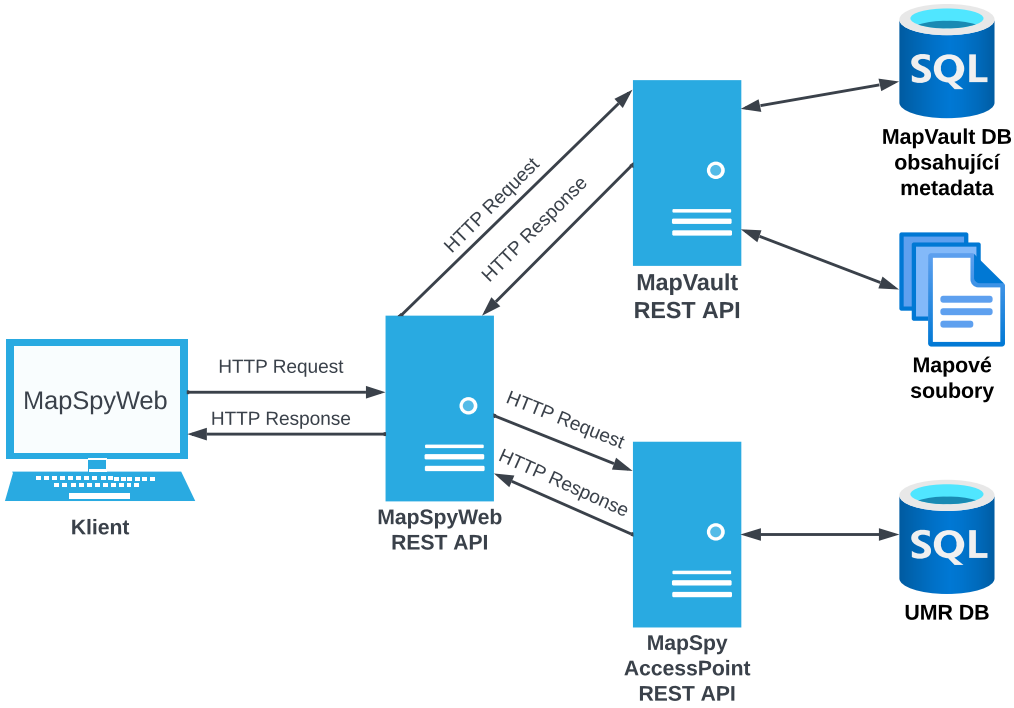
Jelikož zadavatel požaduje, aby se ve výsledku jednalo o webovou aplikaci, bylo zvoleno následující schéma systému. Schématické vyobrazení návrhu je na obrázku 5.1. Byla navržena architektura aplikace klient-server obsahující backend a frontend část. Ve webovém prohlížeči u klienta poběží frontend část aplikace. Backend část aplikace pak poběží na serveru. Pro komunikaci mezi frontend a backend částí se využije REST API umístěné na backendu aplikace. Dále bude backend obsahovat služby, které zajistí poskytování dat pro REST API, komunikaci s ostatními systémy a zdroji dat a mnoho dalších úloh potřebných pro fungování aplikace.

5.1 Návrh backend části aplikace

Backend část aplikace MapSpyWeb je navržena k poskytovaní dat frontend části a zpracovaní dat z frontend části aplikace. Úkolem backend části je tedy získávat data z existujících zdrojů a poskytovat je frontend části ve vhodném formátu, dále také zpracovávat akce a data z frontend části a tyto akce vykonávat nad existujícími daty.

Jak již bylo řečeno, při návrhu sytému bylo vycházeno z existujících řešení ve společnosti onsemi. Tato řešení jsou popsána v kapitole 3.1. Jedná se především o dva

hlavní systémy MapVault a MapSpyAccesspoint, které budou využity jako zdroje dat pro vyvíjenou aplikaci. Schéma propojení aplikace MapSpyWeb a existujících systémů MapVault a MapSpyAccesspoint je schématicky vyobrazeno na obrázku 5.2. MapVault i MapSpyAccespoint obsahují dvě API, a to REST API a SOAP API. Pro komunikaci se zmíněnými systémy bylo navrženo využití HTTP požadavků na REST API systémů.

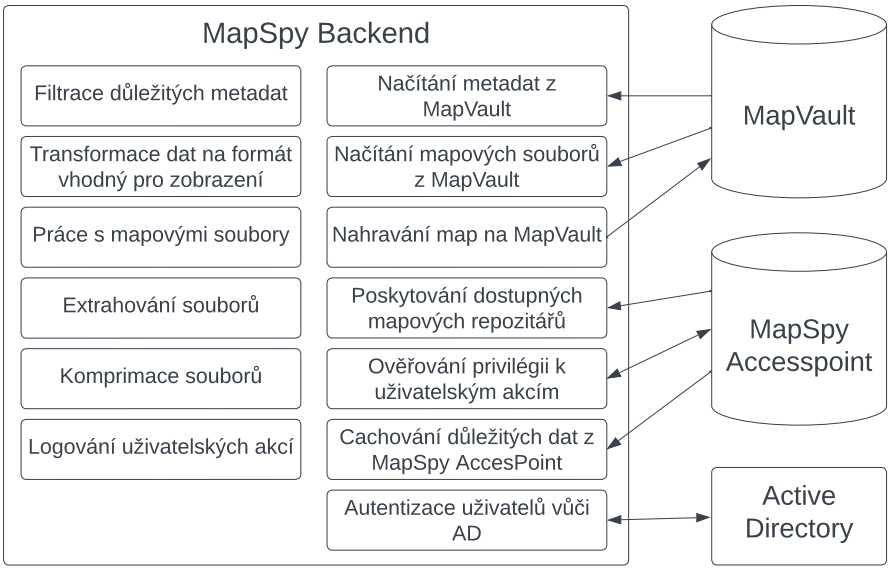


Obr. 5.2: Zasazení architektury MapSpy Web do stávajících řešení.

Výše již bylo zmíněno, že backend část vyvíjené aplikace umístěná na serveru bude sloužit pro komunikaci s frontend části, i systémy MapVault a MapSpyAccesspoint využívat REST API. Tento typ API byl zvolen proto, že je jednoduše škálovatelné, nezávislé na aplikované technologii a velmi vhodné pro využití při tvorbě webových aplikací. REST navíc velmi často využívá pro přenos dat formát JSON, díky tomu je jednoduše využitelný s webovými aplikacemi využívajícími JavaScript. Jelikož SOAP API umožňuje pro přenos dat použít pouze formát XML, pomocí REST API a například formátu JSON bude přenášeno menší množství dat při komunikaci. Tuto skutečnost lze jednoduše vyvodit z faktu, že JSON je znatelně úspornější formát co se velikosti týče, než XML.

5.1.1 Navrhované funkcionality backend části

Přehled všech hlavních navržených a zde popsaných funkcionalit, které by měla obsahovat backend část aplikace MapSpyWeb, je vizuálně zobrazena na obrázku 5.3.



Obr. 5.3: Návrh architektury backend části systému MapSpyWeb.

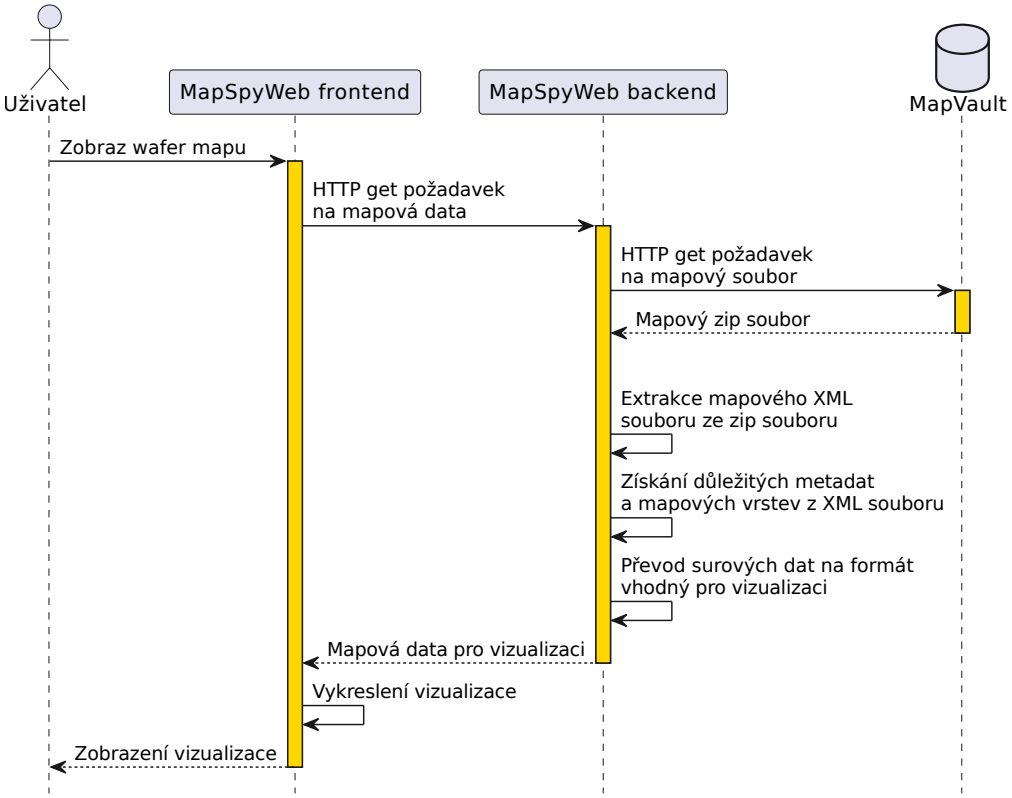
Načtení mapových metadat

V první řadě je nutné, aby backend část aplikace obsahovala funkcionalitu, která dokáže načítat metadata z repozitáře MapVault. Načtená metadata by mělo být možné specifikovat a filtrovat na základně požadavků uživatelů. Pro tyto účely již MapVault obsahuje možnost vytváření dotazů na specifická metadata. Avšak při testování získávání požadovaných metadat přes REST API ze systému MapVault bylo zjištěno, že některé typy dotazů nefungují správně. Při složitějších dotazech na více mapových metadat se stávalo, že API nevrátilo celý požadovaný výsledek nebo nevrátilo požadovaná data vůbec. Proto bude při implementaci potřeba prozkoumat příčinu těchto problémů a opravit chyby v aplikaci MapVault tak, aby splňovalo požadavky pro aplikaci MapSpyWeb.

Jelikož ale metadata získaná z databáze systému MapVault obsahují i metadata, která nejsou pro aplikaci MapSpyWeb potřeba, bude na backend části realizováno odfiltrování nepotřebných metadat. Tento krok bude mít několik pozitivních důsledků. Nejdůležitějším z nich je menší množství dat přenášených do frontend části. S tím souvisí rychlejší načítání dat ve frontend části aplikace a snížení nároků na využívanou operační paměť.

Načtení mapových souborů

Vyvíjená aplikace musí umožňovat vizualizaci map křemíkových desek. Mapy jsou uloženy v MapVault repozitářích a pro jejich načtení bude použito, stejně jako v případě metadat, REST API. MapVault však poskytuje mapy komprimované metodou ZIP. Tento ZIP soubor pak obsahuje mapový soubor ve formátu XML s definicí mapy. Mapové soubory obsahují kromě mapových podkladů i velké množství doplňujících metadat a informací nepotřebných pro aplikaci MapSpyWeb. Nejdříve je tedy potřeba mapové soubory převést do formátu vhodného pro frontend část aplikace.



Obr. 5.4: Návrh postupu získávání mapových dat ve vyvíjené aplikaci.

Návrh stanovuje, že backend část aplikace MapSpyWeb bude zajišťovat načtení požadovaného souboru z repozitáře MapVault. Dále zde bude přítomna funkcionalita, která zajistí extrahování mapy z formátu ZIP. Jedná se totiž o proces, který by zbytečně zatěžoval a zvyšoval požadavky na klientská zařízení, proto návrh počítá s prováděním extrakce mapových souborů na serveru.

Extrahovaný mapový soubor ve formátu XML však stále není vhodný pro účely zobrazování wafer mapy ve vizualizaci. Proto bude za použití knihovny WaferMapJr popsané v sekci 3.1 provedena extrakce užitečných dat ze souboru. Primárně se jedná

o získání požadovaných mapových vrstev obsahujících informace o jednotlivých prvcích mapy. Z pohledu využití paměti představují tyto mapové vrstvy ve většině případů největší část mapového souboru. Vrstvy mapy tedy bude třeba převést na úsporný formát vhodný pro přenos do frontend části aplikace.

Kromě vrstev mapy křemíkové desky obsahuje mapový soubor také značné množství informací o dané mapě. Některé z těchto informací bude nutné zobrazovat ve frontend části aplikace. Proto návrh zahrnuje načtení metadat z mapového souboru pomocí nástroje WaferMapJr, ale pouze těch metadat, která jsou nutná pro aplikaci MapSpyWeb. Důvodem je zvýšení celkové rychlosti aplikace u klienta a snížení paměťových nároků.

Získané mapové vrstvy a mapová metadata bude nakonec ještě potřeba převést na úsporný formát vhodný pro přenos do frontend části aplikace. Zmiňovaný postup získaní mapy, kterým se zabývá tato sekce práce, je pro přehlednost vizualizován diagramem na obrázku 5.4.

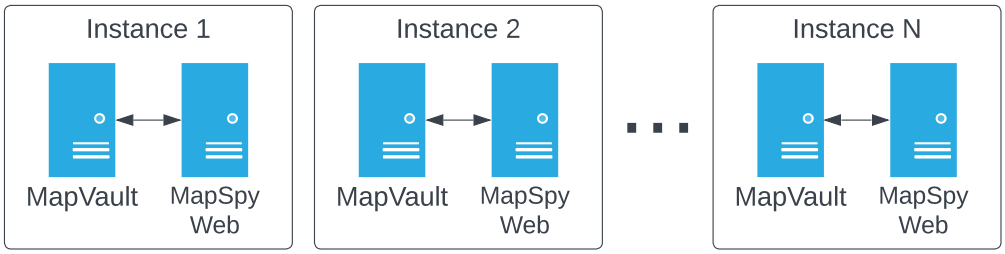
Jelikož by vyvíjená aplikace MapSpyWeb měla poskytovat i možnost editace mapových souborů, bude potřeba, aby backend umožňoval i proces opačný k získávání map, kterým je zpětné nahrávání map do repozitáře MapVault. S tím souvisí editování mapových vrstev a metadat mapového souboru a také generování nového mapového XML souboru za pomocí knihovny WaferMapJr. Kromě zmíněných funkcionalit bude backend část obsahovat funkce pro komprimaci souboru s mapou do formátu ZIP a následné odeslání do repozitáře MapVault.

Řešení přístupu k repozitářům MapVault

Skutečnost, že ve společnosti onsemi existuje mnoho výrobních míst, vede k tomu, že zde běží mnoho instancí aplikace MapValut (podrobněji viz podkapitola 3.1). Vyvíjená aplikace MapSpyWeb by měla být schopna v budoucnu pracovat se všemi běžícími MapVault instancemi. Je tedy třeba určitým způsobem zajistit, aby si uživatel mohl vybrat, se kterou instancí aplikace MapVault chce v MapSpyWeb pracovat. Pro řešení tohoto problému se nabízejí dvě hlavní možnosti.

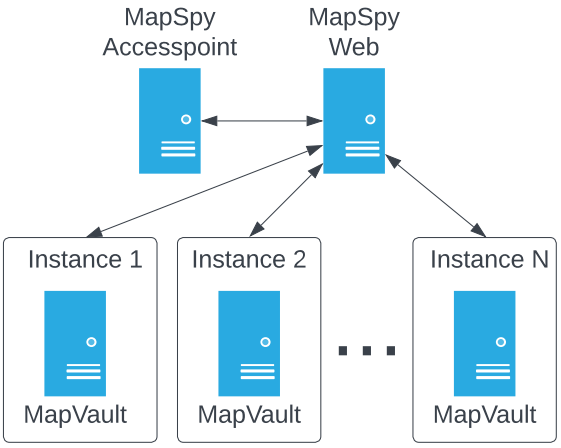
První z nich je možnost, kde pro každou instanci aplikace MapVault bude existovat specifická instance MapSpyWeb. Takové řešení je vizualizováno na obrázku 5.5. Výhodou uvedeného řešení je zamezenívzniku potíží s kompatibilitou mezi aplikacemi MapVault a MapSpyWeb při změně jejich API, neboť nasazené verze těchto dvou aplikací budou vždy zvoleny tak, aby byly vzájemně kompatibilní. Další benefit spočívá v tom, že MapSpyWeb poběží ve stejné geografické lokaci jako většina jeho uživatelů, což by mělo v důsledku vést ke stabilnějšímu připojení a menší odezvě při načítání aplikace. Velkou nevýhodou takového přístupu je velká náročnost na správu, jelikož bude nutné udržovat v běhu mnoho různých verzí aplikace MapSpyWeb na

různých serverech. Problémem pak jsou i aktualizace, neboť při vydání nové verze aplikace MapSpyWeb bude potřeba provádět aktualizaci mnoha instancí a zároveň bude nutné dbát na to, aby nebyl nasazen MapSpyWeb nekompatibilní s odpovídajícím systémem MapVault.



Obr. 5.5: Vizualizace řešení, kde má každý MapVault svou instanci aplikace MapSpyWeb.

Druhou možností, která se nabízí, je existence pouze jedné instance aplikace MapSpyWeb pro všechny instance aplikace MapVault. Řešení je graficky znázorněno na obrázku 5.6. Uživatel si v tomto případě zvolí MapVault, se kterým bude chtít pracovat přímo v aplikaci MapSpyWeb. Dojde tak ke snížení náročnosti údržby a správy mnoha běžících instancí aplikace MapSpyWeb. Avšak popisované řešeni vytváří větší požadavky na vývoj aplikace. Během vývoje nových verzí aplikace MapSpyWeb bude nutné udržovat API všech instancí aplikací MapSpyWeb a MapVault navzájem kompatibilní, aby se předešlo nefunkčnosti a nekompatibilitě s některou starší verzí systému MapVault.



Obr. 5.6: Vizualizace řešení, kde pro různé instance MapVault existuje jeden centrální MapSpyWeb.

Na základě srovnání dvou předchozích přístupu a informací o systémech společnosti onsemi bylo navrženo využití druhého zmíněného přístupu, tedy možnosti, kdy pro všechny instance aplikace MapValut bude existovat pouze jedna instance aplikace MapSpyWeb. Daný přístup velmi zjednoduší distribuci aplikace a správu jejích verzí. Dále napomůže, aby již při vývoji bylo předcházeno možným potížím s kompatibilitou s API aplikace MapVault. Tímto bude aplikace MapSpyWeb v průběhu svého životního cyklu pro vývojáře přehlednější.

V databázi systému MapSpyAccesPoint se nachází záznamy s přehledem existujících instancí MapVault včetně webových adres jejích REST API. Díky tomu bude možné tato data zpřístupnit přes REST API aplikace MapSpyAccesPoint, aby mohla dynamicky poskytovat seznam dostupných repozitářů MapVault.

Další vlastností, která byla u návrhu uvažována, je zajištění vysoké dostupnosti aplikace. Pokud by tedy došlo z nějakého důvodu k nedostupnosti REST API aplikace MapSpyAccessPoint, nebylo by možné používat ani aplikaci MapSpyWeb, jelikož by nebylo možné načíst seznam dostupných repozitářů MapVault. Proto návrh počítá s tím, že budou informace o repozitářích v aplikací MapSpyWeb cachovány. Díky tomu bude zajištěno poskytování seznamu dostupných repozitářů MapVault i v případě, že bude systém MapSpyAccessPoint nedostupný.

Autentizace a autorizace uživatelů

Autentizace a autorizace uživatelů v aplikaci MapSpyWeb bude řešená za pomocí již existujících řešení. Každý zaměstnanec společnosti má své uživatelské ID a účet na firemním Active Directory. Proto bylo pro účely autentizace uživatelů vybráno právě Active Directory. Backend bude připojen k tomuto Active Directory a na požadavek frontend části aplikace bude ověřovat uživatele za použiti jejich uživatelských ID a hesel.

Privilegia jednotlivých uživatelů pro autorizaci uživatelských akcí zajistí aplikace MapSpyAccessPoint (podrobněji viz podkapitola 3.1). Ta totiž spravuje privilegia pro uživatele. Vzhledem ke zmíněnému faktu tedy bylo navrženo, že autorizaci akcí bude provádět backend část aplikace. Pokud se uživatel přihlásí, backend část získá z REST API systému MapSpyAccessPoint přehled všech jeho privilegii. Na základě těch pak budou akce uživatele autorizovány. Privilegia však budou předány i frontend části aplikace. A to z toho důvodu, aby mohl frontend dynamicky vyhodnocovat, k jakým akcím má uživatel oprávnění, a podle toho mu zpřístupňovat pouze možnosti, ke kterým má přístup. Vyhodnocování privilegii na frontend části aplikace je zde však pouze proto, aby došlo ke zlepšení uživatelské přívětivosti. Nejedná se o bezpečné vyhodnocování, jelikož uživatel může svá privilegia prohlížeči podvrhnout nebo upravit JavaScript kód aplikace. Bezpečné vyhodnocování privilegii bude

zajišťovat právě backend část aplikace MapSpyWeb.

Výše v této práci byla popisována navržená funkcionalita na cachování dostupných repozitářů pro případ, že bude nedostupný MapSpyAccessPoint. Toto cachování je vhodné použít i v případě uživatelských privilegii. Primárním důvodem je to, že pokud dojde k nedostupnosti MapSpyAccessPoint, nebude možné načíst uživatelská oprávění a uživatel nebude moci vykonávat akce, ke kterým by měl mít oprávnění. Právě proto se zde nabízí možnost cachování uživatelských privilegii. Zde je však důležité myslet na to, že u cache bude potřeba zajistit, aby byla validní jen po určitý časový úsek, a následně po rozumně stanoveném čase zneplatněna z důvodu bezpečnosti. Jinak by se mohlo stát, že se v momentě, kdy nebude dostupný MapSpyAccessPoint, pokusí o přihlášení uživatel, který při minulém přihlášení například před půl rokem, měl určitá oprávnění, která aktuálně již nemá. Pak by systém poskytl uživateli oprávnění, která měl před půl rokem a mohl by provádět neoprávněné akce, což se nesmí stát. Proto je velmi důležité zajistit softwarově i časovou validitu cache.

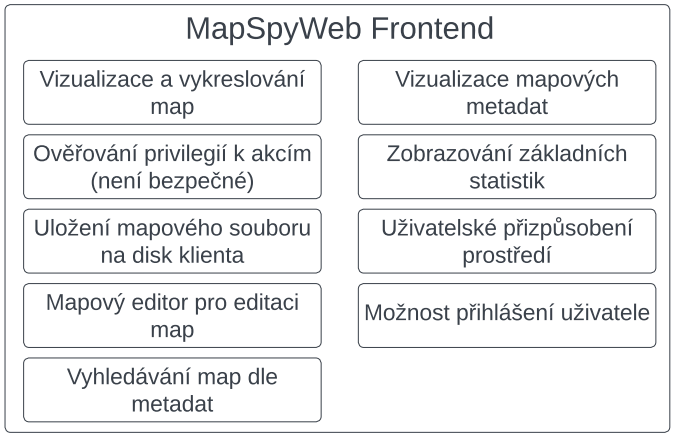
5.2 Návrh frontend část aplikace

V rámci diplomové práce byl proveden návrh frontend části aplikace MapSpyWeb. Úkolem frontend části aplikace je zobrazování a poskytování dat uživateli ve vhodném formátu. Mimo jiné má také umožnit interakci s těmito daty, jak již bylo zmíněno v úvodu kapitoly číslo 5. Pro komunikaci s backend částí aplikace MapSpyWeb bylo navrženo řešení s využitím REST API. Následující podkapitoly se věnují jak návrhu funkcionalit, za které bude frontend část aplikace zodpovědná, tak návrhu vhodného uživatelského rozhraní aplikace.

5.2.1 Funkcionality frontend části aplikace

Přehled všech navržených funkcionalit, které bude vykonávat frontend část vyvíjené aplikace, je shrnuta na obrázku 5.7. Data pro všechny tyto funkcionality poskytuje REST API backend části aplikace MapSpyWeb. Další odstavce této podkapitoly se věnují návrhu funkcionalit, které bude zajišťovat frontend část.

Dle návrhu bude frontend část aplikace zajišťovat následující akce. První z nich je vyhledávaní map dle specifických metadat. Proto bude aplikace obsahovat vhodné vyhledávací okno. Následně je třeba mapová metadata vizualizovat tak, jak specifikují požadavky na systém uvedené v kapitole 4. Kromě vizualizace metadat musí frontend také zajišťovat jeden z nejdůležitějších úkolů a tím je vizualizace map křemíkových desek. Vizualizace umožní interaktivitu s jednotlivými prvky mapy, vykreslování různých vrstev mapy a hlavně editaci těchto map. Uživateli bude také



Obr. 5.7: Návrh architektury frontend části systému MapSpyWeb.

dána možnost stahovat mapové soubory k sobě na disk ve formátu, ve kterém jsou uloženy v úložišti MapVault.

Uživatelské prostředí bude vyvíjeno tak, aby bylo co nejvíce přizpůsobitelné uživateli a jeho požadavkům. Zde bude třeba určit ty části aplikace, které svou možností přizpůsobení umožní uživateli jednodušší práci s aplikací.

5.2.2 Návrh uživatelského rozhraní

Na základě požadavků zadavatele specifikovaných v kapitole 4 byl proveden návrh uživatelského rozhraní aplikace MapSpyWeb. K návrhu uživatelského rozhraní se nabízely dva hlavní možné přístupy, a to buďto využití náčrtů na papír, nebo provádění návrhů za pomocí vhodného softwaru. Výhodou prototypování za pomocí softwaru je mnohem jednodušší provádění změn u již vytvořených návrhů. Pokud by například bylo potřeba vyzkoušet, jak bude nějaký prvek uživatelského prostředí vypadat na různých místech, je možné jej jednoduše v softwaru přesunout. V případě, že by však bylo využito návrhů na papír, jednoduché přesunutí samozřejmě není možné. Na základě těchto skutečností byl vybrán návrh uživatelského rozhraní za pomocí softwaru Justinmind [26] v jeho verzi 9.9.4. Tento software je přímo určen pro vytváření prototypů a návrhu designu webových stránek. Nástroj Justinmind již obsahuje hlavní HTML objekty jako jsou tabulky, tlačítka, textové vstupy a další. Kromě toho lze pro návrh uživatelského rozhraní využívat mnoho geometrických objektů.

Za pomoci aplikace Justinmind tedy byly vytvořeny návrhy základních stránek uživatelského rozhraní aplikace MapSpyWeb. První stránkou aplikace, která byla navržena, je stránka pro přihlášení uživatele do aplikace. Návrh je k nahlédnutí v obrázkové příloze A.1. Dále byl proveden návrh stránky aplikace sloužící pro vyhledávání, který je znázorněn v příloze na obrázku A.2. Návrh hlavního okna aplikace s vizualizací výsledků vyhledávaní je pak ke zhlédnutí v příloze na obrázku A.3.

5.2.3 Vizualizace map křemíkových desek

Na základě rešerše dostupných možností pro vykreslování grafiky na webu uvedené v kapitole 2, byly navrženy 3 možnosti pro vytvoření grafické vizualizace map křemíkových desek. Jedná se o vykreslovaní vizualizace pomoci vektorové grafiky SVG (kapitola 2.4.1) nebo s využitím bitmapového nástroje HTML5 Canvas (kapitola 2.4.2) či nástroje WebGL (kapitola 2.4.3), který využívá grafické akcelerace. Pro přehlednější implementaci je vhodné prozkoumat možnosti knihoven pracujících s SVG, HTML5 Canvas a WebGL. Využití knihoven se jeví jako vhodné primárně proto, aby byla v maximální míře omezena implementace funkcí, které samotné nástroje SVG, HTML5 Canvas a WebGL neposkytují, ale implementují je již některé knihovny, jež jsou k dispozici.

Z nástrojů vhodných pro vytváření vizualizací na webu zmíněných v předchozím odstavci se jako nejméně vhodný jeví SVG. Důvodem je, že škálovatelná vektorová grafika (SVG) obsahuje omezení, které se týká množství vykreslených elementů. Všechny tyto elementy jsou součástí DOM, což má zásadní vliv na výkon aplikace (viz sekce 2.4.1). Na základě požadavků na vizualizační systém je potřeba vykreslování map až do velikosti 1 000 x 1 000 prvků (viz kapitola 4.1.5). To znamená, že musí být najednou vykresleno 1 000 000 interaktivních prvků. U SVG se však objevují výkonnostní problémy již u tisíců vykreslených prvků. Proto bylo pro implementaci navrženo využití technologie HTML5 Canvas s tím, že pokud by z výkonnostních důvodů Canvas nedostačoval, pro vykreslování bude využita technologie WebGL.

6 Praktická realizace systému MapSpyWeb

Kapitola se zabývá vývojem a implementací zadané aplikace MapSpyWeb. Primárně je zde popsán postup vývoje frontend a backend části aplikace. Dále jsou uvedeny překážky, které při vývoji vznikly, a informace, jak se je podařilo vyřešit. Zmíněna je také zvolená struktura aplikace a mnoho dalších detailů ohledně softwaru aplikace.

Pro verzování zdrojového kódu při vývoji bylo využito nástroje GIT. K vývoji byly ve většině případů využívána dvě vývojová prostředí. Prvním z nich je IntelliJ IDEA [27] od společnosti JetBrains, které bylo využito primárně k vývoji v jazyce Java. Pro vytváření testovacích kódů vizualizace mapy křemíkových desek a pro realizaci zdrojového kódu frontend části aplikace bylo použito prostředí Visual Studio Code [28] od společnosti Microsoft vynikající svou rychlostí a množstvím doplňků.

6.1 Realizace frontend části aplikace

Tato sekce shrnuje praktickou implementaci frontend části vyvíjené aplikace MapSpyWeb. Pro softwarovou realizaci byl vybrán webový framework Angular [8], a to na základě diskuze ohledně jeho možností a vlastností. Zadání vyvíjené aplikace napovídá, že výsledná webová aplikace bude poměrně rozsáhlá. Dle [29] je pak pro rozsáhle aplikace s mnoha funkcemi vhodnější framework Angular než framework React [30]. Angular obsahuje ve srovnání s frameworkem React mnohem více vestavěných funkcionalit, a není proto nutné používat při vývoji mnoho externích knihoven.

Pro účely zjednodušení procesu tvorby uživatelského rozhraní bylo využito knihovny PrimeNG v13.4.1 [31]. Tato knihovna obsahuje přes 80 komponent uživatelského rozhraní určených pro Angular aplikace [32]. Jedná se o komponenty pro formuláře, zobrazování dat, dialogy, vyskakovací okna, menu, tlačítka, tabulky a mnoho dalšího. Díky tomu je možné snadno využít již existující komponenty, a tím zjednodušit zdrojový kód aplikace a zkrátit čas vývoje.

Pro ulehčení definování uživatelského rozhraní pomocí kaskádových stylů byla využita knihovna Bootstrap ve verzi 5.1.3 [33]. Ta obsahuje mnoho předdefinovaných stylů, které lze velmi jednoduše použít pro tvorbu uživatelského rozhraní webové aplikace. Knihovna Bootstrap byla vytvořena za účelem standardizace nástrojů sloužících k vytváření uživatelského rozhraní v rámci společnosti Twitter. Tento nástroj obsahuje mnoho prvků pro tvorbu responzivních webových stránek, jako jsou tabulky definované pomocí kaskádových stylů, tlačítka, menu a další prvky uživatelského rozhraní. [34]

Implementace aplikace se opírá o teoretický návrh uvedený v kapitole 5. Dle něj byly ve frontend části implementovány funkcionality pro vizualizaci map křemíkových desek, vizualizaci mapových metadat, vyhledávání map dle metadat, možnost

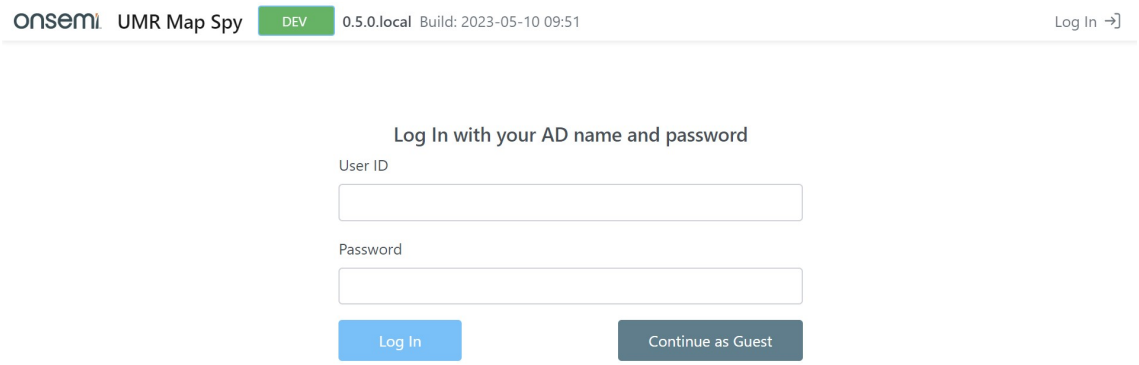
přihlášení uživatele a další. Vše je zmíněno v přehledu na obrázku 5.7. Pro zdrojový kód frontend části byla vytvořena dokumentace nacházející se v elektronické příloze (příloha F).

Frontend část aplikace byla za pomocí modulů rozdělena na 4 hlavní stránky. První z nich je stránka pro přihlášení uživatele. Další je stránka pro vyhledávání dle mapových metadat určená k nalezení požadovaných map a dodatečných informací o nich. Hlavní stránkou aplikace je souhrn výsledků vyhledávání, na kterém je zobrazen přehled mapových metadat a vizualizace map křemíkových desek. Poslední stránka zajišťuje možnost manuální editace map křemíkových desek. Jednotlivé stránky jsou detailněji rozebrány v následujících odstavcích.

6.1.1 Realizace úvodní stránky sloužící k přihlášení uživatele

Nejprve bude popsána stránka zajišťující přihlášení uživatele do aplikace MapSpy-Web. Tato stránka umožňuje přihlášení za pomocí uživatelského identifikátoru a hesla, které uživatel využívá ve firemním Active Directory. Na základě přihlášení je uživatel aplikace autentizován, a může tak provádět úkony, ke kterým má specifická oprávnění.

Popis uživatelského rozhraní stránky pro přihlášení



Obr. 6.1: Snímek úvodní stránky aplikace sloužící k přihlášení uživatele.

Vytvořené uživatelské rozhraní stránky pro přihlášení obsahuje formulář se dvěma textovými elementy. Ty slouží k zadání uživatelského identifikátoru a hesla. Pod textovými vstupy byla vytvořena dvě tlačítka. První z nich slouží k akci přihlášení a druhé k pokračování jako host. Vzhled úvodní stránky pro přihlášení lze vidět na obrázku číslo 6.1.

Pokud uživatel zvolí možnost pokračování jako host, může se během používaní aplikace dodatečně přihlásit pomocí dialogu. Skutečnost, zda je uživatel přihlášen, je vizualizována v pravé horní části aplikace uživatelským ID a ikonou uživatele. V případě, že uživatel přihlášen není, v pravé horní části se vyskytuje tlačítko „*Log In*“ sloužící k zobrazení již zmíněného přihlašovacího dialogu.

Popis implementace stránky pro přihlášení

V rámci frameworku Angular byla stránka pro přihlášením realizována v rámci modulu `login.module.ts`. Komponenta obsahující univerzální přihlašovací formulář s textovými vstupy a tlačítky byla umístěna do modulu `security.module.ts`. Tento univerzální formulář je pak využíván jak na stránce pro přihlášení, tak v přihlašovacím dialogu.

V modulu `security.module.ts` je také umístěna služba `authentication .service.ts`. Ta je zodpovědná za přihlašování pomocí POST požadavku na REST API backend části aplikace a vyhodnocení přihlašovacího požadavku. Dále zabezpečuje udržovaní informací o přihlášeném uživateli a zajišťuje možnosti korektního odhlášení uživatele. Služba mimo jiné obsahuje proměnnou typu `BehaviorSubject` z reaktivní knihovny RxJS [35]. Tato proměnná funguje na principu publikovatel – odběratel (angl. *Publisher – Subscriber*) a je schopná vytvářet události. Jiné části aplikace mohou události odebírat a na jejich základě konat požadované akce. Konkrétně služba `authentication.service.ts` obsahuje publikovatele, který zveřejňuje informace o tom, zda je přihlášen uživatel. Některé jiné části aplikace pak tuto událost odebírají a jsou informovány, zda a jaký uživatel je k aplikaci aktuálně přihlášen.

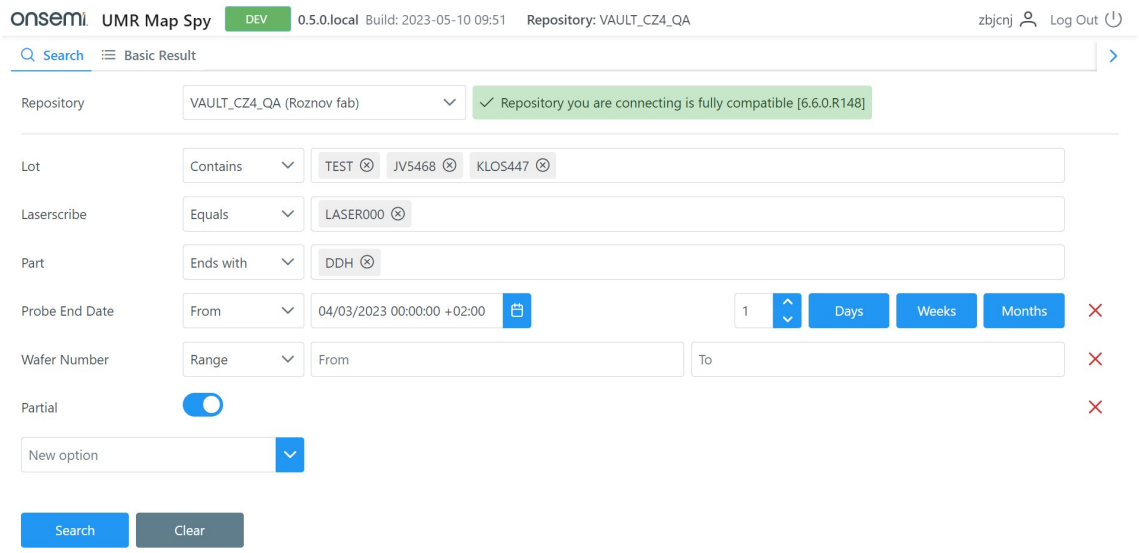
V souvislosti s přihlášením jsou ve službě `privileges.service.ts` načtena uživatelská oprávnění, a to za pomocí volání GET požadavku na REST API backend části, načež si služba uloží privilegia daného uživatele. Mimo tuto funkci obsahuje služba metody určené pro vyhodnocení, zda má uživatel oprávnění k uživatelským akcím. Více informací k autentizaci a autorizaci je uvedeno v podkapitole 6.3.

6.1.2 Realizace stránky pro vyhledávaní map křemíkových desek a jejich metadat

Stránka pro vyhledávání je druhou stránkou, na kterou uživatel aplikace přejde po přihlášení. Jejím úkolem je umožnit co nejjednodušší a nejrychlejší vyhledávání v mapových souborech a jejich metadatech.

Popis uživatelského rozhraní stránky pro vyhledávání

Vizuální vzhled uživatelského rozhraní stránky je zobrazen na obrázku 6.2. Stránka pro vyhledávání se skládá ze dvou částí. První z nich je horní část s rozbalovacím seznamem sloužícím k výběru repozitáře, se kterým chce uživatel pracovat. Pokud uživatel při výběru podrží kurzor myši na jakémkoli repozitáři v rozbalovacím seznamu déle než půl sekundy, zobrazí se bublina s dodatečnými informacemi. Po výběru repozitáře se vedle rozbalovacího seznamu zobrazí informace o jeho kompatibilitě se systémem MapSpyWeb.



Obr. 6.2: Snímek stránky určené pro vyhledávání map křemíkových desek a jejich metadat.

Druhou částí stránky pro vyhledávaní je samotný formulář sloužící k vyhledávání map křemíkových desek. Ve výchozím stavu obsahuje formulář tři základní vyhledávací řádky s poli pro vyhledávání. V levé části každého řádku je název vyhledávací položky. Vedle názvu se nachází rozevírací seznam s dostupnými vyhledávacími operátory a samotné vyhledávací pole, do kterého lze zadávat požadované hodnoty.

Položky, na základě kterých je možné vyhledávat, mají různé datové typy. Proto se vyhledávací pole přizpůsobuje danému datovému typu a zvolenému vyhledávacímu operátoru. Pro datový typ text (*string*) jsou dostupné operátory „*Contains*“, „*Equals*“, „*Start with*“ a „*End with*“. Do samotného vyhledávacího pole lze zadat více hodnot, a tím vyhledat více výsledků. Číselné datové typy umožňují zvolit operátory „*Equals*“, „*Less than*“, „*More than*“ a „*Range*“.

Datový typ datum nabízí operátory „*From*“ a „*Range*“. Operátor „*From*“ poskytuje možnost zadat počáteční datum buďto výběrem z kalendáře, nebo pomocí

tlačítek na pravé straně. Tato tlačítka umožňují nastavit počáteční datum na předešlých n dní/týdnů/měsíců od aktuálního data. Posledním podporovaným datovým typem je pravdivostní hodnota, u které lze jednoduchým přepínačem zvolit, jestli má být daná položka *true* nebo *false*.

Pod vyhledávacími poli se nachází rozbalovací seznam s názvem „*New option*“. Pomocí něj lze přidávat další metadata k vyhledání, a provádět tak složitější dotazy. Po výběru položky v seznamu je pod existující vyhledávací řádky přidán řádek nový. Pokud je následně potřeba přidaný řádek odstranit, je možné to učinit kliknutím na červený křížek v pravé části vyhledávacího řádku.

Ve spodní části stránky se nachází dvě tlačítka. Jedno slouží pro zahájení vyhledávání a druhé pro smazaní vyhledávacího formuláře.

Popis implementace stránky pro vyhledávání

Stránka pro vyhledávání je realizována v rámci modulu `map-spy.module.ts` Angular komponentou `search.component.ts`. Ta spolu s komponentu pro výběr repozitáře vytváří vyhledávací formulář. Komponenta pro výběr repozitáře je nazvána `repository-select.component.ts`. Obsahuje seznam dostupných repozitářů načtených z backend části aplikace a zajišťuje získání informací o dostupnosti a kompatibilitě zvoleného repozitáře. Součástí implementace je také služba `repository-select.service.ts` obsahující dvě proměnné typu `Subject` z knihovny RxJS [35]. Ty slouží k publikování událostí s informacemi o změně repozitáře. Kupříkladu stránka s výsledky vyhledávaní na základě dané události vymaže zobrazené výsledky vyhledávání.

Výpis 6.1: Ukázka úpravy existující PrimeNG komponenty pomocí ng-template.

```
<p-calendar [(ngModel)]="model" [showTime]="true" [showSeconds]="true">
    <ng-template pTemplate="footer">
        <div class="d-flex justify-content-center">
            <p-dropdown
                [options]="listOfTimezones" [(ngModel)]="timezone">
            </p-dropdown>
        </div>
    </ng-template>
</p-calendar>
```

Jednotlivé řádky vyhledávacího formuláře jsou tvořeny ze čtyř komponent pro různé datové typy, a to pro zadávaní textů, čísel, dat, a pravdivostních hodnot. Zmíněné komponenty jsou sdruženy abstraktní třídou `input-abstract.component.ts`, která jim udává společné funkcionality a rozhraní. Jedná se například o funkcionalitu mazání či inicializaci a odstranění vyhledávacího pole. Samotné uživatelské

vstupy jsou realizovány pomocí komponent knihovny PrimeNG [31]. Ve vyhledávacím formuláři je pro zadávání textu využita PrimeNG komponenta umožnující vložení více textových hodnot nazvaná `p-chips`. Pro zadávání data je použita komponenta `p-calendar`, která implementuje dialog s kalendářem, kde si může uživatel navolit požadované datum. V některých případech je však potřeba komponentu z knihovny PrimeNG upravit podle svých potřeb. Zde bylo nutné přidat do kalendáře volbu časového pásma, jelikož aplikaci budou používat uživatelé v různých částech planety. Pro to nabízí Angular ve spojení s knihovnou PrimeNG elegantní řešení, které spočívá v tom, že pomocí klíčového slova jazyka Angular `ng-template` je do existující komponenty vložen vlastní kód.

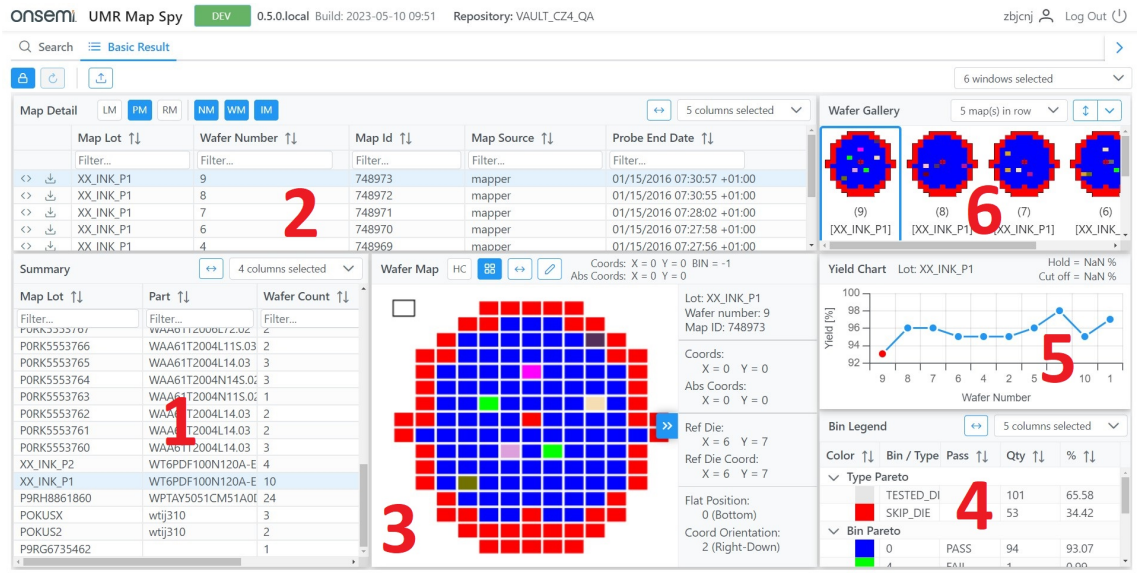
Ukázka takové úpravy komponenty je vidět ve výpisu 6.1. Na řádku číslo 2 jsou využita klíčová slova `ng-template pTemplate="footer">`, která říkají, že je do komponenty na pozici `footer` vložen kód na řádcích 4, 5 a 6. Dané řádky obsahují rozbalovací seznam s časovými pásmy. Takto jednoduše bylo provedeno rozšíření existující PrimeNG komponenty o rozbalovací seznam s časovými pásmy.

Při vyvolání akce vyhledávání je v komponentě `search.component.ts` nejdříve zjištěno, jaké položky jsou ve vyhledávacím formuláři vyplněny a zda jsou všechny zadané hodnoty validní (např. ověření, že v poli pro číselné hodnoty je zadáno opravdu číslo, a podobně). Pokud není identifikován žádný problém se zadanými daty, je proveden GET požadavek s vyhledávacím dotazem na backend část. Po úspěšném požadavku se uživateli zobrazí stránka s výsledky vyhledávání.

6.1.3 Realizace stránky pro zobrazení výsledků vyhledávání a vizualizaci map křemíkových desek

Stránka pro zobrazení výsledků vyhledávaní je realizována komponentou `basic-result.component.ts` v rámci modulu `map-spy.module.ts`. Zahrnuje několik oken, přičemž každé má svou specifickou funkci. Jedná se o okno obsahující tabulku shrnující nalezené mapy a metadata křemíkových desek podle dávky (angl. *lot*), tabulku s metadaty map v rámci vybraných dávek, vizualizaci mapy křemíkově desky včetně legendy, vizualizaci všech map v rámci dávky a graf zobrazující statistické parametry křemíkových desek. Vizuální podobu této stránky lze nalézt na obrázku 6.3 a ve větší verzi v příloze B.3.

Pro zajištění možnosti maximálního přizpůsobení aplikace byla v rámci rešerše nalezena knihovna pro responzivní, uživatelsky nastavitelnou mřížku *angular-gridster* [36]. Ta umožňuje vložit do stránky mřížku s okny, která lze pomocí myši zvětšovat, zmenšovat či vzájemně prohazovat. Využití mřížky dovoluje uživatelům přizpůsobit si pozici a velikost oken ve výsledcích vyhledávaní podle svých preferencí a požadavků. Funkcionalita zmíněné mřížky je implementována v komponentě



Obr. 6.3: Snímek stránky se souhrnem výsledků vyhledávání a vizualizací mapy křemíkové desky.

`basic-result.component.ts` a definována abstraktní třídou `gridster-abstract.component.ts`.

V levé horní části stránky s výsledky vyhledávání se nachází tři tlačítka. Tlačítko s ikonou zámku slouží k zamknutí mřížky, aby nebylo možné okna posouvat a zvětšovat. Vedle něj je tlačítko s ikonou šipky určené k obnovení výchozího rozložení a velikosti všech oken. Napravo je ještě jedno tlačítko oddělené svislou čárou, které slouží k ručnímu nahrání mapového souboru do repozitáře pomocí dialogového okna (viz obrázek B.2b v přílohách). V pravé části se nachází rozbalovací seznam, který umožňuje výběr oken, které mají být v responzivní mřížce zobrazeny. Zde byla implementována funkce zajišťující uložení nastavené velikosti a pozice oken pro přihlášené uživatele do úložiště prohlížeče `localStorage` [37]. Následně, po dalším přihlášení, tedy zůstane toto rozložení zachováno.

Popis uživatelského rozhraní okna se souhrnem dávek a okna s informacemi o mapách v rámci dávky

Prvním popisovaným oknem výsledků vyhledávání je okno se souhrnem nalezených dávek. Na obrázku 6.3 je označeno číslem 1. V tomto okně se nachází tabulka s metadaty nalezených dávek křemíkových desek. Data v tabulce lze jednoduše řadit kliknutím na název sloupce. Pomocí držení klávesy *Ctrl* a klikání myší je možné vybrat více sloupců, na základě kterých se mají data řadit. V levé horní části okna s tabulkou je rozbalovací seznam. Ten slouží k výběru sloupců, které mají být v ta-

bulce zobrazeny. Vedle něj se nachází tlačítko s vodorovnou šipkou určené k automatickému nastavení šířky sloupců. V okně se souhrnem nalezených dávek je možné klikat na jednotlivé řádky, čímž dojde ke zobrazení tabulky všech map patřících do dané dávky ve vedlejším okně.

Seznam všech map patřících do dávky zobrazuje okno s číslem 2 na obrázku 6.3. Zde jsou v tabulce zobrazena metadata z výroby patřící k jednotlivým mapám. Dané okno se na rozdíl od prvního (popisovaného v předchozím odstavci) liší tím, že hlavička tabulky obsahuje navíc přepínací tlačítka pro filtrování určitých typů map. Jedná se například o filtraci různých typů map, filtraci odstraněných map či zobrazení pouze poslední verze dané mapy v tabulce. Po kliknutí na mapu v této tabulce dojde k jejímu zobrazení v okně sloužícím k vizualizaci mapy.

Součástí tabulky je také kontextové menu, které umožňuje provádět některé další akce nad řádky v tabulce. Zmínit je možné například vybrání mapy k editaci, zobrazení originálního mapového souboru v dialogovém okně (viz obr. B.2a v přílohách), uložení mapového souboru na disk či zkopírování hodnoty v tabulce do schránky. Ukázka kontextového menu je zachycena na obrázku B.1c v přílohách.

Popis implementace okna se souhrnem dávek a okna s informacemi o mapách v rámci dávky

Společným znakem okna se souhrnem dávek a okna s informacemi o mapách v rámci dávky je zobrazení textových či číselných metadat v tabulce. Pro implementaci tabulky byla vybrána komponenta `p-table` ze sady komponent od PrimeNG [31]. Ta již ve výchozím stavu obsahuje mnoho užitečných funkcionalit. Lze zmínit například možnost výběru řádků, rolování a řazení zobrazených dat. Jelikož tato tabulka bude využívána pro více oken vyvíjené aplikace, byla vytvořena komponenta `map-spy-table.component.ts` a abstraktní třída `map-spy-table-abstract .component.ts`. Abstraktní třída definuje funkcionality společné pro všechny tabulky. Jedná se zejména o zpracování událostí jako kliknutí do tabulky, dvojkliku na tabulku, otevření kontextového menu nebo zajištění uložení zobrazených sloupců pro specifického uživatele do úložiště `LocalStorage` [37]. Komponenta s tabulkou `map-spy-table.component.ts` dědí zmíněnou abstraktní třídu a zajišťuje rozšíření funkcionalit tabulky z knihovny komponent PrimeNG.

Okno se souhrnným zobrazením nalezených dávek je implementováno za pomocí komponenty `lot-summary.component.ts` a je vykresleno v již popisované přizpůsobitelné mřížce *angular-gridster*. Pro okno s tabulkou obsahující metadata jednotlivých map z dávky byla vytvořena komponenta `map-detail.component.ts`. Ta je oproti komponentě se souhrnem dle dávek složitější a obsahuje více funkcionalit.

Komponenta `map-detail.component.ts` obsahuje implementaci kontextového

menu, které již bylo zmíněno výše. Některé z akcí kontextového menu jsou omezeny uživatelskými oprávněními a jsou povoleny pouze určitým uživatelům. Proto je při otevření kontextového menu kontrolováno, zda má uživatel k akcím oprávnění. Pokud je vyhodnoceno, že uživatel oprávnění nemá, není možné položku vybrat.

V tabulce obsahující mapy z dávky je možné výběrem položky „*Save map file*“ kontextového menu uložit XML mapový soubor na disk zařízení. Ke stejném účelu slouží i ikona s šipkou směřující dolů, která se nachází v levé části každého řádku tabulky. Stažení souboru zajišťuje služba `file-downloading.service.ts`. Ta za pomocí metody `showSaveFilePicker` zobrazí okno pro výběr umístění při stahování souboru. Následně je odeslán GET požadavek na REST API backend části, mapa je stažena do aplikace a pomocí `File System Access API` [38] je stažený soubor uložen na disk.

Položka kontextového menu „Open map file“ byla implementována k rychlému nahlédnutí do mapového souboru. Stejně jako u stahování je i zde u každého řádku tabulky ikona, tentokrát se symbolem „`<>`“. Po kliknutí na tento symbol je XML mapový soubor načten z backendu a zobrazen v dialogovém okně (obr. B.2a v přílohách). Díky této možnosti nemusí uživatel mapový soubor stahovat do zařízení, pokud si jej potřebuje jen zobrazit.

Popis uživatelského rozhraní okna s grafem výtěžnosti

Okno s grafem výtěžnosti slouží k vizualizaci výtěžnosti křemíkových desek v průběhu výrobního procesu a v rámci dávky. Na obrázku 6.3 se jedná o okno označené číslem 5. Výtěžnost se počítá dle rovnice

$$yield = \frac{n_{good}}{pdpw} 100 \qquad [\%], \tag{6.1}$$

kde n_{good} je počet funkčních prvků mapy a *pdpw* (angl. *potential die per wafer*) je potenciální maximální počet funkčních prvků mapy.

Hlavička okna obsahuje název dávky a napravo dvě hodnoty – *Hold* a *Cut off*. Jedná se o limitní hodnoty výtěžnosti v procentech. Ty slouží k vizuálnímu označení map, které mají nižší výtěžnost než stanovené limity. Hlavní částí okna je graf, který výtěžnost vizualizuje. Jedná se o interaktivní spojnicový graf se značkami. Jeho interaktivita spočívá v tom, že po kliknutí na bod v grafu je možné vybrat mapu zobrazenou ve vizualizaci stejně, jako by uživatel klikal na mapy v tabulce pro zobrazení map z dávky.

Popis implementace okna s grafem výtěžnosti

Okno s grafem výtěžnosti je implementováno jako komponenta `yield-chart .component.ts`. Pro výběr interaktivního grafu byla provedena rešerše dostup-

ných knihoven k vykreslování grafů. Zde přicházelo v úvahu uplatnění již využívané knihovny PrimeNG, avšak ta neposkytuje dostačující možnosti personalizace. Byla proto zvolena knihovna Chart.js [39], která poskytuje dostatečné možnosti přizpůsobení, jako je detailní nastavení barev, os, mřížky, nápovědy v bublině (angl. *tooltip*) a dalších.

Pro umožnění přidání grafických anotací do grafu byla knihovna Chart.js rozšířena pomocí jiné knihovny chartjs-plugin-annotation [40]. Ta dává možnost doplnit do grafu obrazce, text a další geometrické objekty. Dodatečná knihovna umožnila přidat do grafu text s hodnotami limitů výtěžnosti *Hold* a *Cut off* a také změnu barvy pozadí grafu pro hodnoty menší než limity výtěžnosti.

Popis uživatelského rozhraní okna s přehledem map křemíkových desek a popis jeho implementace

Dalším oknem na stránce s výsledky vyhledávání je okno zobrazující souhrnnou vizualizaci všech map křemíkových desek pro jednu nebo více dávek. Okno je znázorněno na obr. 6.3, kde je označeno číslicí 6 a také na obrázku B.6 v přílohách. Hlavní částí okna je mřížka, ve které jsou uspořádány obrázky s náhledy map křemíkových desek včetně jejich popisu. Okno je interaktivní, takže kliknutím na mapu je daná mapa vykreslena v okně pro plnohodnotnou vizualizaci křemíkové desky. V horní části okna je tlačítko, které umožňuje okno zvětšit buďto na půl stránky, nebo na téměř celou webovou stránku. Dalším prvkem rozhraní je rozbalovací seznam, který umožňuje měnit počet obrázků map zobrazených na řádku, a tím upravovat velikost a počet zobrazených map v okně.

Implementace okna s přehledem map křemíkových desek proběhla v rámci komponenty `wafer-gallery.component.ts`. Přehled zobrazuje obrázky map, které jsou generované na serveru, jelikož vykreslení tolika vizualizací by bylo výpočetně náročné. V rámci komponenty `wafer-gallery-image-grid.ts` pak byla implementována logika zajišťující vykreslení obrázků map do mřížky. Velikost mřížky je určena programem tak, aby se vměstnala do šířky okna, ve kterém je vykreslována. Dle toho se přizpůsobí i velikost obrázků map, aby byly vidět všechny obrázky v řádku, pokud však není překročena minimální velikost obrázků. V případně překročení minimální velikosti se část obrázků map schová a ve spodní části okna se zobrazí horizontální posuvník.

Popis uživatelského rozhraní okna s vizualizací mapy křemíkové desky

Stěžejní částí stránky s výsledky vyhledávání je okno s interaktivní vizualizací mapy křemíkové desky. To je na obrázku 6.3 označeno číslem 3. Největší část okna zabírá samotná vizualizace mapy, která je detailněji popsána v dalších částech diplomové

práce. Okno obsahuje hlavičku s několika tlačítky. První z nich slouží k aktivaci módu vysokého kontrastu zobrazení mapy, další zajišťuje zapnutí či vypnutí zobrazení mřížky v mapě. Nachází se zde i tlačítko pro vycentrování mapy do zobrazovací oblasti nebo tlačítko sloužící k zahájení editace mapy. Posledním prvkem hlavičky je zobrazení aktuálních souřadnic mapy.

U pravého okraje vizualizace mapy se nachází vysouvací menu obsahující doplňující informace o mapě, které využijí uživatelé aplikace. Jsou zde informace o zobrazené mapě, o jejím souřadnicovém systému, vybraném prvku či o tom, kde se nachází *flat*[1] křemíkové desky.

Implementace okna s vizualizací je provedena v rámci komponenty `map-window.component.ts` a postranní panel okna pak v rámci komponenty `map-window-sidebar.component.ts`. Samotná implementace vizualizace mapy je popsána v další části této práce, v podkapitole 6.1.6.

Popis uživatelského rozhraní a implementace okna s legendou mapy křemíkové desky

K oknu s interaktivní vizualizací mapy křemíkové desky se váže legenda mapy. Na obrázku 6.3 je označena číslem 4. Legenda je podstatnou součástí mapy, neboť uživateli aplikace pomáhá rozklíčovat, co vizualizace zobrazuje. Legenda je realizována jako tabulka a obsahuje dvě záložky. První z nich je záložka „*Type Pareto*“ obsahující vysvětlivky jednotlivých prvků mapy ve vizualizaci, které nebyly ve výrobě testovány či se na nich nenacházejí čipy. Druhou záložkou je „*Bin Pareto*“, zde se nachází prvky mapy, které byly testováním či jiným způsobem označeny jako funkční (*pass*), nebo naopak jako vyřazené (*fail*).

Každý řádek tabulky obsahuje 5 sloupců. První sloupec tabulky s názvem „Color“ obsahuje barvu, kterou je daný prvek mapy vizualizován. Druhým sloupcem je „Bin / Type“, ve kterém je uvedeno, jakému typu prvku mapy daná barva odpovídá. Typem může být například „*SKIP_DIE*“ označující prvky, které nemají být testovány, nebo „*UNTESTED_DIE*“ reprezentující prvky, které nebyly otestovány. Může se zde nacházet také číselná hodnota, která u testovaných prvků uvádí typ a příčinu selhání. Další sloupec s názvem „*PASS*“ informuje, zda byl testovaný prvek vyhodnocen jako funkční nebo nefunkční. Posledními dvěma sloupci jsou „*QTY*“ a „*%*“. Sloupec „*QTY*“ zobrazuje četnost prvků na mapě odpovídající danému řádku a sloupec „*%*“ jeho procentuální zastoupení.

Součástí legendy je i kontextové menu (obr. B.1b v přílohách), které obsahuje specifické funkcionality. První funkcionalitou je možnost skrytí prvku mapy. Po vy-

[1] Jedna strana křemíkové desky je zkosená a je označována jako *flat*. Flat slouží k identifikaci orientace křemíkové desky a pro přesné zarovnání během výroby

brání této možnosti dojde ke skrytí daného prvku ve vizualizaci. Druhou funkcionalitou je položka pro změnu barvy. Ta uživateli umožňuje změnit barvu daného prvku ve vizualizaci. Ke změně barvy slouží dialogové okno vyobrazené na obrázku B.2b v přílohách.

Implementace legendy k mapě křemíkové desky je zajištěna Angular komponentou `bin-legend.component.ts`. Vstupem této komponenty jsou mapová data definované rozhraním `WaferMapContentDto`. Z poskytnutých dat je vytvořena legenda a jsou vypočítány četnosti jednotlivých typů prvků mapy. Dialog pro uživatelskou změnu barev je realizován pomocí PrimeNG komponenty `p-dialog`. V dialogu je využita komponenta `chrome-picker` z knihovny ngx-color-picker [41] sloužící k interaktivnímu výběru barvy.

6.1.4 Realizace stránky pro ruční editaci map křemíkových desek

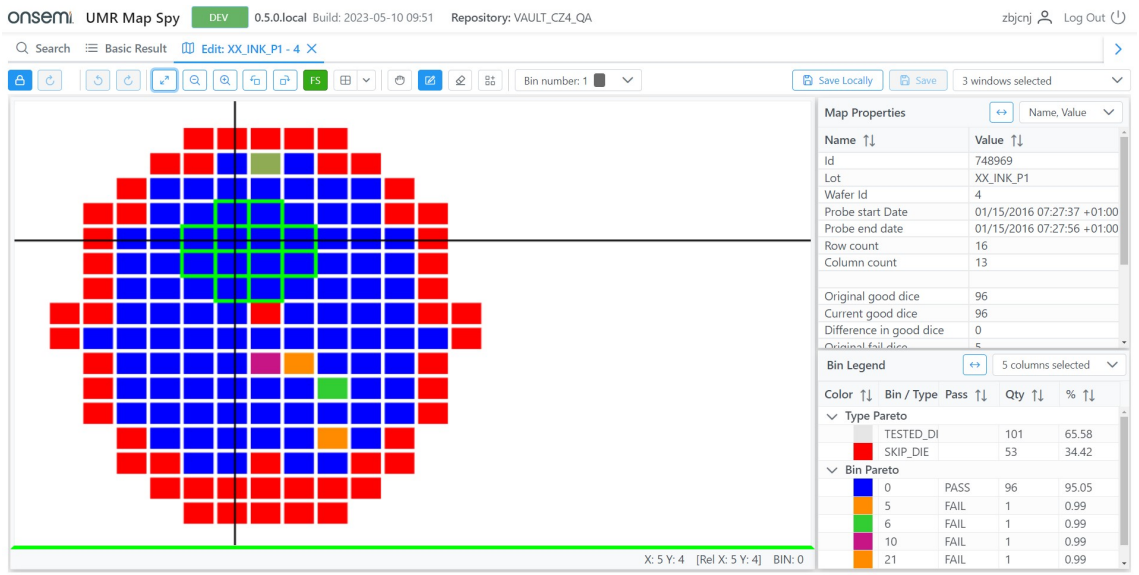
Podstatným prvkem aplikace je stránka mapového editoru sloužící k ruční anotaci map křemíkových desek. Stránka je implementována v rámci modulu `map-editor .module.ts` a komponenty `map-editor.component.ts`. Vizuální podoba stránky pro editaci map křemíkových desek je vidět na obrázku 6.4.

Stránka s editorem umožňuje provádět změny nad existující mapou křemíkové desky. Přejít na editační stránku lze ze stránky s výsledky vyhledávání. Po vybrání mapy k editaci je proveden GET požadavek na REST API aplikace, ze kterého jsou získána mapová data editované mapy. Uživatelské rozhraní je zde, stejně jako výsledky vyhledávání, umístěno do responzivní uživatelsky přizpůsobitelné mřížky *angular-gridster* [36].

Popis uživatelského rozhraní stránky pro ruční editaci map křemíkových desek

V horní části uživatelského rozhraní se nachází nástrojová lišta mapového editoru obsahující mnoho tlačítek. Jedná se například o tlačítka zpět a opakovat, pomocí kterých lze vrátit nebo opakovat akce v editoru. Dále je zde možné najít tlačítka pro otočení vizualizace mapy o 90° doprava, doleva či pro horizontální otočení. Součástí jsou také tlačítka umožnující nastavení samotné editace, a to konkrétně tlačítko pro pohyb na mapě, tlačítko pro kreslení do mapy ke označení prvků jako vyřazených (*fail*), tlačítko gumy a tlačítko pro rozšířeni skupiny vadných prvků. Nachází se zde i možnost pro vycentrování mapy do okna nebo nastavení mřížky.

Největší část stránky zabírá okno určené k vizualizaci a editaci mapy. Dále se zde vyskytuje okno s doplňujícími informacemi o mapě, kde jsou podrobnosti k editované mapě křemíkové desky shrnuty v tabulce. Posledním oknem mapového editoru je legenda mapové vizualizace. Jedná se o totožnou legendu jako na stránce s výsledky vyhledávání (kapitola 6.1.3).



Obr. 6.4: Snímek stránky určené pro ruční editaci map křemíkových desek.

Popis implementace stránky pro ruční editaci map křemíkových desek

Nástrojová lišta mapového editoru se všemi tlačítky a ovládacími prvky byla implementována pomocí komponenty `map-editor-tool-bar.component.ts`. K indikování uživatelských akcí využívá nástrojová lišta službu `map-editor-tool-bar.service.ts`. V této službě jsou umístěny proměnné typu `Subject` z knihovny RxJS [35], které publikují události při uživatelských akcích s lištou.

Samotné okno s vizualizací a interaktivním editorem mapových dat je implementováno pomocí komponenty `map-editor-window.component.ts`. Daná komponenta odebírá události z nástrojové lišty a předává je vizualizační komponentě. Pro vizualizaci mapy je zde využita stejná komponenta jako ve výsledcích vyhledávání. Její implementace je popsána v podkapitole 6.1.6.

Okno a doplňujícími informacemi o mapě využívá již dříve popsané univerzální komponenty tabulky `map-spy-table.component.ts`, v níž vizualizuje informace o mapě. Také okno s legendou vizualizace mapy křemíkové desky na stránce mapového editoru je totožné s oknem legendy nacházejícím se na stránce s výsledky vyhledávaní. Oba jsou realizována totožnou komponentou `bin-legend.component.ts`.

6.1.5 Výběr softwarové implementace vizualizace map křemíkových desek

Pro mapy křemíkových desek byla v sekci 5.2.3 na základě průzkumu dostupných technologii vybrána dvě hlavní možná technologická řešení. Jedná se o vykreslovaní

mapy pomocí bitmapové rastrové grafiky s využitím nástroje HTML5 Canvas nebo WebGL. Následující odstavce se věnují implementaci, postupu a výsledkům testování zmíněných řešení.

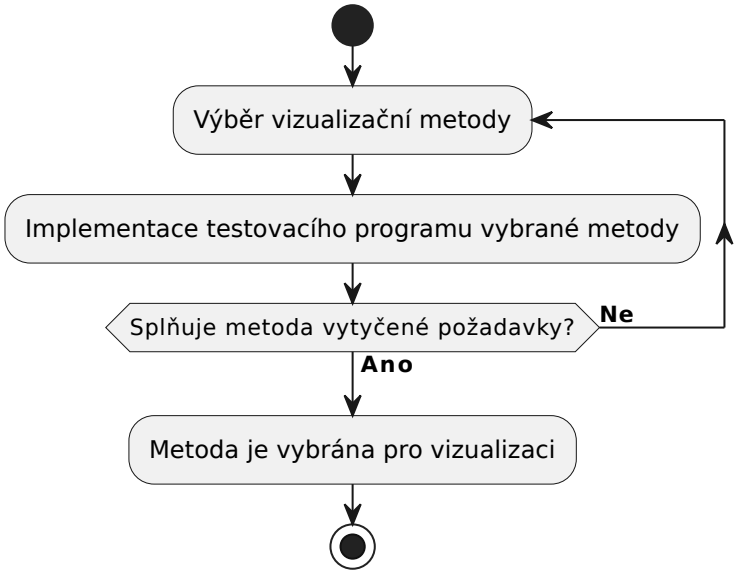
Při implementaci byl brán zřetel na to, aby zdrojový kód měl co největší vypovídající hodnotu o principu fungování a byl ideálně co nejkratší. Dodržování těchto cílů může zásadně zlepšit dlouhodobou udržitelnost zdrojových kódů programu. Zmíněné cíle vybízí k použití knihoven, které poskytují více vysokoúrovňový přístup k technologiím Canvas a WebGl a již implementují funkce, které by bylo jinak nutné programovat. Využitím knihoven je tedy možné se vyhnout psaní kódu funkcí, které jsou již přítomny v knihovnách a je možné je rovnou použít. Pro Canvas je lze využít knihovnu KonvaJS [42], pro WebGl pak například knihovnu PixiJS [17] určenou pro 2D grafiku a knihovnu nazvanou Three.js [43] pro 3D grafiku.

Pro výběr nejvhodnějšího řešení bylo vytvořeno několik testovacích programů v jazyce JavaScript. Tyto programy využívají různých metod vizualizace map a slouží primárně k otestování správnosti metod vybraných při návrhu systému. Základním požadavkem na vizualizaci a zároveň první implementovanou funkcionalitou v testovacích programech je vykreslení mapy obsahující mřížku až 1 000 x 1 000 prvků. Dalším požadavkem je možnost změny měřítka mapy a možnost pohybu po mapě pomocí myši. Na těchto funkcionalitách bude demonstrována schopnost technologie splnit podmínky dané zadavatelem. Testovací aplikace popsané v následujících odstavcích obsahují kód pro vykreslení náhodné mapy křemíkové desky. U mapy je následně v prohlížeči měřena rychlost vykreslovaní. Pro měření rychlosti bylo využito metody `performance.now()` z JavaScript Performance API určeného pro měření výkonnosti webových aplikací.

Výběr vhodné vizualizační metody pro implementaci

Pro výběr a implementaci vizualizační metody byl vytvořen následující postup, při kterém byly metody implementovány a testovány postupně v definovaném pořadí. Proces výběru metody probíhal následujícím způsobem. Nejprve byla vybrána technologie pro vizualizaci, která se zdála být nejméně složitá, a existoval tak předpoklad, že zdrojový kód implementace nebude rozsáhlý. Následně byla vizualizace mapy implementována za pomocí zvolené technologie. Poté proběhlo otestování vytvořeného programu a byly zhodnoceny výsledky. Pokud výsledný program nesplňoval požadavky na rychlost a plynulost, stejným postupem byla vybrána jiná metoda pro otestování. Celý zmíněný postup je graficky znázorněn v diagramu na obrázku 6.5.

Veškeré testování vizualizačních metod implementovaných v této části práce, pokud není v textu uvedeno jinak, bylo prováděno za využití hardware uvedeného v následujících odrážkách.



Obr. 6.5: Schéma procesu sloužícího k výběru nejvhodnější vizualizační metody.

- **Procesor:** Intel Core i5-8250U 1.6GHz (turbo 3.4GHz) Quad-Core
- **Operační paměť:** 12GB DDR4 2400MHz SODIMM
- **Grafický procesor:** Intel UHD Graphics 620

Software využitý k testování je vypsán v následujících odrážkách.

- **Operační systém:** Windows 11 Home 22H2 Build 22621.819 (64bit)
- **Webový prohlížeč:** Google Chrome 107.0.5304.123 (64bit)

Implementace za využití Canvas

První testovanou metodou vizualizace byla zvolena rastrová grafika využívající Canvas. Canvas byl vybrán primárně z toho důvodu, že je přímo podporován všemi hlavními prohlížeči, nevyžaduje grafickou akceleraci, jako je tomu například u WebGL, a navíc se zdá být pro daný problém poměrně jednoduchý na implementaci.

Testovací program s Canvas knihovnou Konva.js

První testovací program byl vytvořen za pomocí Canvas knihovny nazvané Konva.js [42]. Knihovna byla použita proto, že již obsahuje mnoho implementovaných řešení, která budou pro vizualizaci potřeba. Jedná se například o možnost využití scény `Konva.Stage`, do které lze vykreslovat požadované objekty. Vlastnost scény `draggable` přímo umožňuje vykreslený obraz libovolně posouvat pomocí myši. Dále je možné jednoduše programově nastavovat měřítko pomocí funkce `Stage.scale()`. Díky knihovně Konva.js rovněž není nutné obraz manuálně překreslovat při každé

změně vizualizace, neboť to je prováděno automaticky. Zdrojový kód testovacího programu je k nahlédnutí v souboru `testovaci_program_konva.html` v elektronické příloze (příloha F).

Výsledky implementace za použití knihovny Konva.js však nebyly uspokojivé. Zdrojový kód byl sice jednoduchý a přehledný, ale výkon aplikace nebyl dostatečný. Již při využití mapové mřížky o rozměru 250x250, což odpovídá celkem 62 500 vykresleným prvkům se objevily problémy s plynulostí a odezvou. Posouvaní mapy bylo značně trhané a u změny měřítka mapy se objevovala značná latence. Ve zmíněném případě se doba vykreslování pohybovala průměrně okolo hodnoty 175 ms. Mapa velikosti 1 000 x 1 000 prvků se dala vykreslit pouze jednou a při následném překreslovaní webový prohlížeč zobrazil chybovou hlášku o nedostatku operační paměti. Paměťové nároky této knihovny byly obrovské. Využití operační paměti testovací aplikací se pohybovalo okolo 4,1 GB pro mapu velikosti 1 000 x 1 000. Takové výsledky jsou pro požadovanou aplikaci naprosto nedostatečné. Proto bylo přistoupeno k otestování dalšího přístupu.

Testovací program Canvas

Další možností pro vizualizaci mapy, která byla implementována a testována, je využití přímo Canvas API webového prohlížeče. Výhoda v tomto případě spočívá v eliminaci nutnosti používat knihovny třetích stran. Nevýhodou je však to, že zde existuje pouze nízkoúrovňové Canvas API. Důsledkem je náročnější implementace složitějších funkcionalit při použití Canvas API než v případě využití knihoven. S využitím technologie Canvas bylo očekáváno zvýšení výkonu, neboť zde odpadá režie spojená s využitím knihovny. Pro dosažení dalšího zrychlení byl použit odlišný přístup k pohybu po mapě. Změna spočívá tom, že mapa je ve vizualizaci vykreslována za pomocí Canvas API v požadovaném měřítku pokaždé celá, ale na obrazovce se uživateli zobrazí pouze její požadovaná část. Zbytek mapy přeteče mimo zobrazovací oblast definovanou v HTML a na okrajích zobrazovací oblasti se objeví posuvníky. Pomocí posuvníku je pak možné se v mapě pohybovat, aniž by ji bylo nutné při pohybu znovu překreslovat. Pro přiblížení je však stále nutné mapu překreslit. Testovací program pro Canvas je v souboru `testovaci_program_canvas.html` v elektronické příloze (příloha F).

U testovacího programu, který využívá Canvas API webového prohlížeče, bylo dosaženo lepších výsledků než v případě předchozího přístupu, kde je využita knihovna Konva.js. Testování s mapou velikosti 1 000 x 1 000 vykazovalo následující výsledky. Doba potřebná pro překreslení mapy při změně měřítka mapy se pohybovala okolo 1,7 vteřiny. Tato hodnota sice neumožňuje plynulé přiblížení a oddálení mapy, ale uživatelsky je již toto řešení omezeně použitelné. Excelentních výsledků však bylo

dosaženo při pohybu po mapě. Jelikož se mapa vykresluje vždy celá a v zobrazovací oblasti prohlížeče se nachází pouze zvolená část mapy, je pohyb po mapě za pomocí posuvníků okamžitý, plynulý a bez citelné odezvy. Také paměťové nároky pro dané řešení nejsou velké. Vizualizace využívala při vykreslování mapy velkosti 1 000 x 1 000 pouze okolo 32 MB paměti RAM a 455 MB grafické paměti. Výsledky tohoto přístupu jsou slibné, avšak testování nadále pokračovalo pro nalezení přístupu, který bude rychlostně výkonnější ve vykreslování vizualizace při změně měřítka.

Implementace za využití WebGL

Druhou metodou vhodnou pro vykreslovaní grafiky a vizualizací na webu po technologii Canvas je technologie WebGL. Výhoda WebGL spočívá ve využití grafického procesoru pro vykreslování, díky čemuž dokáže pracovat rychleji než Canvas využívající pouze CPU. WebGL je velmi nízkoúrovňová technologie, která umožňuje vykreslovat pouze body, čáry a trojúhelníky [43]. Aby bylo možné vytvořit složitější obraz, je obvykle potřeba poměrně velké množství zdrojového kódu. Tento nedostatek však řeší knihovny. Pro implementaci bylo uvažováno využití jedné ze dvou knihoven, které jsou aktivně vyvíjeny a mají velkou uživatelskou základnu vývojářů. Jedná se o knihovny PixiJS [17] a Three.js [43].

PixiJS je knihovna zaměřena a uzpůsobena primárně pro práci s 2D grafikou. Knihovna Three.js je určena zejména pro vykreslování trojrozměrné grafiky. Při použití PixiJS je tedy ve většině případů práce s dvojrozměrnou grafikou jednodušší a jeví se jako vhodnější pro implementaci vizualizace mapy křemíkové desky.

Testovací program s WebGL knihovnou PixiJS

Třetí testovací program byl implementován za pomoci WebGL knihovny PixiJS. Knihovna poskytuje vysokoúrovňový přístup k vykreslování, proto je možné za použití krátkého zdrojového kódu pracovat s poměrně složitou dvojrozměrnou grafikou. V kombinaci s knihovnou PixiJS je možné využít doplněk nazvaný `pixi-viewport` [44], který poskytuje pro knihovnu funkci 2D kamery. Díky tomu je možné se ve vykreslené scéně pohybovat pomocí myši, provádět přiblížení pohledu kamery či přidávat omezení možného rozsahu pohybu kamery a měřítka. Všechny zmíněné funkce, které knihovna PixiJS a doplněk pixi-viewport obsahují, jsou velmi dobře využitelné pro vytvoření požadované vizualizace křemíkových desek. Testovací program pro PixiJS je v souboru `testovaci_program_pixijs.html` v elektronické příloze (příloha F).

Pomocí testovacího programu využívajícího knihovnu PixiJS a doplněk `pixi-viewport` bylo dosaženo velmi nadějných výsledků. S mapou o velkosti 1 000 x 1 000

vykreslovaných prvků bylo dosaženo velmi plynulé změna měřítka mapy pomocí kolečka myši s časem překreslení okolo 22 ms. Také pohyb po mapě pomocí kliknutí myši byl u této metody vykreslování velmi plynulý a pro uživatele bez znatelné odezvy. Čas překreslení se u pohybu mapy vyskytoval v podobném rozmezí jako u přiblížení, a to okolo 22 ms. Aplikace využívající PixiJS byla navíc otestována při běhu na výkonnější grafické kartě AMD Radeon RX 560 při vykreslování mapy velikosti 2 000 x 2 000. Taková velikost mapy odpovídá 4 000 000 vykresleným objektům. I v tomto případě se za použití karty AMD Radeon RX 560 podařilo dosáhnout plynulého pohybu po mapě a plynulých změn měřítka. Paměťové nároky zde však byly vyšší než při řešení pomocí Canvas API. Využití paměti RAM se v případě mapy velikosti 1 000 x 1 000 pohybovalo okolo 885 MB. Kromě paměti, kterou upotřebila samotná aplikace, je potřeba připočítat i operační paměť využitou procesem GPU, tedy dalších přibližně 670 MB. Celkově se sice jedná o vyšší hodnotu, ale dnešní pracovní stanice a notebooky obsahují dostatek paměti RAM pro zvládnutí této vizualizace. Využití knihovny PixiJS splnilo vytyčené požadavky, a proto již nebylo potřeba vytvářet další testovací program využívající jinou technologii zobrazování.

Výsledky a srovnání testovaných vizualizačních metod

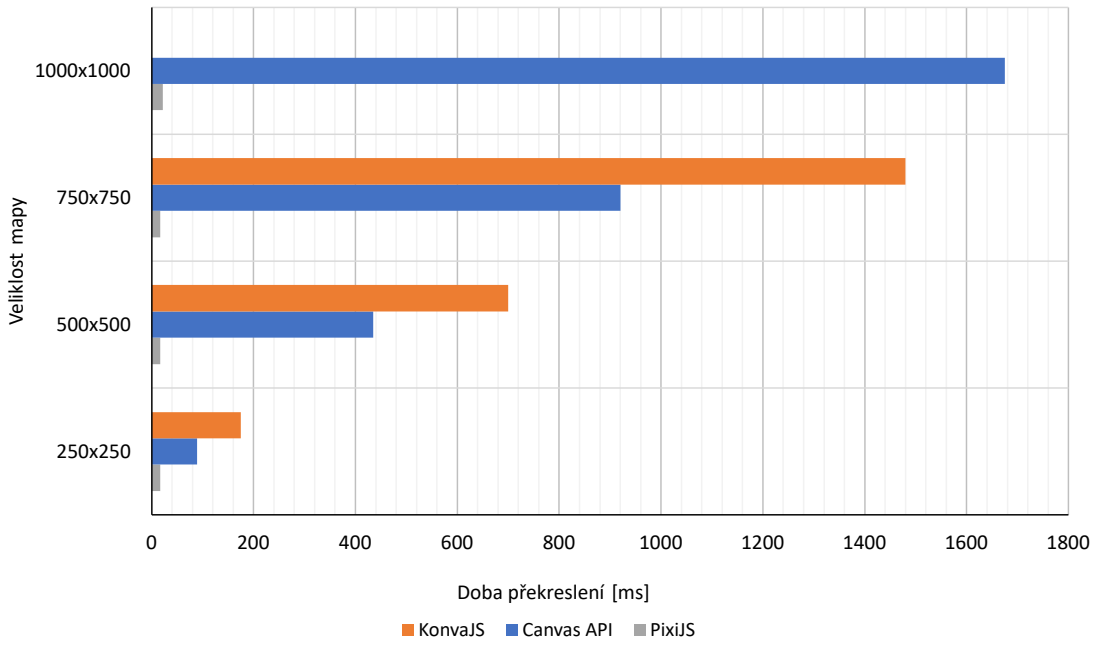
Tab. 6.1: Přehled výsledků jednotlivých testovaných metod pro vizualizaci map křemíkových desek.

		Velikost mapy			
Typ zpětné vazby	**Vizualizační metoda**	250x250	500x500	750x750	1000x1000
Doba překreslení [ms]	Konva.js	175	700	1480	-
	Canvas API	89	435	920	1675
	PixiJS	17	17	17	22
Paměť RAM [MB]	KonvaJs	363	1280	2580	4090
	Canvas API	26	26	27	32
	PixiJS	75	256	513	885
Paměť GPU [MB]	KonvaJs	71	69	70	94
	Canvas API	89	158	282	455
	PixiJS	196	285	430	670

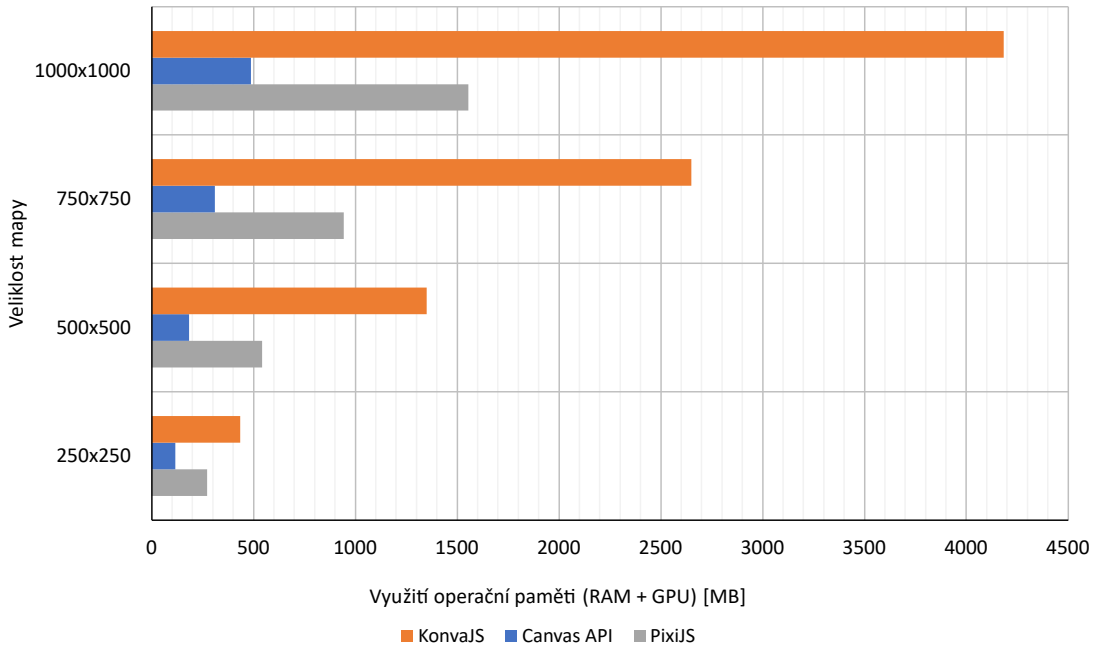
Všechny dosažené výsledky jsou přehledně shrnuty v tabulce 6.1. Srovnání rychlosti vykreslování jednotlivých použitých technologií je také znázorněno pomocí grafu na obrázku 6.6. Porovnání paměťových nároků je pak vyobrazeno na grafu na obrázku 6.7.

Z výsledků je zřejmé, že pro vytvoření vizualizace není možné využít knihovnu KonvaJS. Hlavním důvodem je skutečnost, že mapu požadované velikosti nelze za pomocí této knihovny překreslovat, a navíc její paměťové nároky jsou velmi vysoké. Také použití Canvas API se nezdá jako vhodné řešení, a to zejména proto, že doba překreslení pro změnu měřítka vizualizace je poměrně vysoká. Avšak, co se týče využití paměti, dosahuje Canvas API nejlepších výsledků ze všech tří testovaných možností. Pro ověření kompatibility byly testovací programy vyzkoušeny nejen na webovém prohlížeči Google Chrome, ale také na prohlížečích Microsoft Edge a Mozilla Firefox. Na obou dodatečně testovaných prohlížečích byly výsledky velmi podobné, a to jak z pohledu rychlosti vykreslovaní, tak z pohledu využití paměti.

Na základě testovacích programů byla pro implementaci vybrána metoda vizualizace mapy s využitím WebGL knihovny PixiJS. Tento přístup poskytuje dostatečný výkon, plynulost i rychlost. Knihovna dokonce překonává požadavky na navrhovaný systém. Na výkonnějších grafických procesorech zvládá testovací aplikace pracovat i s mapou o velikosti 2 000 x 2 000 prvků. To je výhodné z důvodu, že ve výsledné aplikaci nebude nutné vizualizaci mapy pouze zobrazovat, ale bude potřeba do vizualizace přidávat i další interaktivní prvky pro práci s touto mapou. Při implementaci dalších funkcí ve vizualizaci mapy tedy nebude vývoj narážet na limity vizualizační metody.



Obr. 6.6: Srovnání rychlosti vykreslování implementovaných testovacích metod vizualizace.

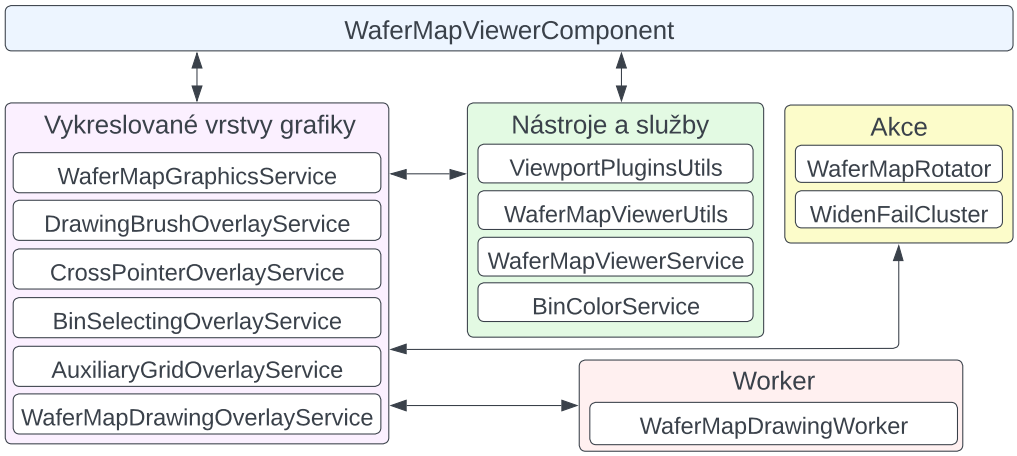


Obr. 6.7: Srovnání využití paměti u implementovaných testovacích metod vizualizace.

6.1.6 Výsledná implementace vizualizace mapy polovodičových součástek za pomoci knihovny PixiJS

Implementace vizualizace mapy křemíkové desky proběhla v samostatném modulu `waferm-map.module.ts` a komponentě `wafer-map-viewer.component.ts`. Volba umístění vizualizační komponenty do samostatného modulu byla důležitá, protože implementace byla prováděna v souladu s možností zdrojový kód vizualizace v budoucnu osamostatnit do samostatné knihovny. S tím souvisí fakt, že zde popisovaná část aplikace je jen minimálně závislá na implementaci ostatních částí aplikace MapSpyWeb. Architektura vytvořeného vizualizačního modulu je pro přehlednost schématicky vyobrazena na obrázku 6.8.

Samotná komponenta `wafer-map-viewer.component.ts` zajišťuje inicializaci vykreslovacího plátna PixiJS a inicializaci *pixi-viewport* s funkcí 2D kamery. Ukázka inicializace PixiJS je uvedena ve výpisu C.1 a ukázka inicializace *pixi-viewport* kontejneru ve výpisu C.2 v rámci příloh diplomové práce. Dále komponenta zpracovává vstupy a výstupy, tedy například vstupní mapová data, mód, ve kterém vizualizace běží (editace, nebo pouze zobrazení), události klávesnice a myši provedené nad vizualizací, změny nastavení vizualizace, nastavení módu editace, nastavení mřížky a mnoha dalších parametrů. Popis softwarového rozhraní komponenty vizualizace je uveden v příloze E. Komponenta následně na základě vstupů pracuje s něko-



Obr. 6.8: Schéma architektury modulu obsahujícího implementaci vizualizace mapy křemíkové desky.

lika dalšími službami a nástroji, které zajišťují různé funkcionality vizualizace map křemíkových desek. V dalších odstavcích jsou popsány hlavní služby a nástroje vizualizačního modulu.

Vykreslování mapy křemíkové desky s polovodičovými součástkami

Vykreslování vizualizace mapy křemíkové desky do Canvas elementu vizualizace za pomocí knihovny PixiJS je realizováno ve službě `wafer-map-graphics.service.ts`. Tato služba zajišťuje vytvoření objektu typu `Graphics` knihovny PixiJS, ve kterém je dále vytvářena grafika vizualizace mapy.

Mapová data jsou v rámci frontend části aplikace reprezentována dvourozměrným polem čísel. Během vytváření grafiky vizualizace jsou procházeny hodnoty tohoto dvourozměrného pole mapových dat `for` cyklem. V rámci cyklu jsou následně na odpovídajících souřadnicích v prostoru vytvářeny obdélníky definované velikosti znázorňující prvky mapy křemíkové desky. Samotné vložení obdélníku probíhá pomocí PixiJS funkce `Graphics.drawRect`. Nakonec, poté, co je zpracována celá mapa, je objekt s grafikou mapy `Graphics` přidán do *pixi-viewport*. Tím je mapa automaticky vykreslena do odpovídajícího elementu v prohlížeči. Ukázka kódu sloužícího k vytvoření grafiky mapy je uvedena ve výpisu C.3 v přílohách.

Velikost obdélníkového prvku mapy je určena reálnou velikostí součástky na křemíkové desce. Pro stanovení velikosti slouží metoda `getWaferMapBinSize` ze služby `wafer-map-viewer.service.ts`. Daná metoda zjišťuje z mapových metadat reálnou výšku a šířku polovodičové součástky na křemíkové desce v milimetrech. Následně je tato velikost normalizována a vrácena ve formátu vhodném pro použití

při vytváření vizualizace. Kromě určování velikosti prvků křemíkové desky obsahuje zmíněná služba `wafer-map-viewer.service.ts` také kód zajišťující omezení maximálního přiblížení a oddálení vizualizace mapy pomocí kolečka myši.

Každý prvek mapy křemíkové desky je ve vizualizaci reprezentován určitou barvou. Pro určení barvy a typu prvků mapy slouží služba `bin-color.service.ts`. Ta obsahuje metody zajišťující získání barvy pro odpovídající číselnou hodnotu prvku z mapových dat. Výchozí barvy pro vizualizaci jsou definovány v JSON souboru `bin-colors.json`. Při inicializaci služby `bin-color.service.ts` jsou barvy z daného JSON souboru načteny a následně s nimi služba pracuje. Barvy jsou uživatelsky měnitelné, proto je zde obsažen i kód umožňující změnu barvy prvků ve vizualizaci. Jak již bylo zmíněno dříve v této práci, službu `bin-color.service.ts` využívá také komponenta pro zobrazování legendy vizualizace mapy, která obsahuje i dialogové okno pro změnu barvy.

Editace mapy křemíkových desek

První službou, která zajišťuje funkcionality spojené s editací mapy, je služba `drawing-brush-overlay.service.ts`. Ta zabezpečuje vykreslování štětce sloužícího k editaci map. Štětec je tvořen obdélníky zelené barvy vizualizujícími oblast, se kterou uživatel při editaci pracuje. Služba zajišťuje zobrazení, skrytí a aktualizaci polohy štětce na mapě. Dále je zde uložena definice 4 velikostí štětců. Z těch si uživatel může vybírat kliknutím pravého tlačítka myši na vizualizaci při editaci mapy (obr. B.1a v přílohách).

Samotná editace mapy probíhá ve službě `wafer-map-drawing-overlay.service.ts`. Nejprve byl implementován přístup, kde při editaci docházelo k aktualizaci vrstvy grafiky s původní mapou. Zde se však objevily problémy s výkonem při překreslování celé grafiky původní mapy křemíkové desky. Bylo tedy zřejmé, že tímto směrem se není možné ubírat. Proto byl navržen a implementován přístup, kde ve vizualizaci probíhá editace mapy tak, že jsou nad vrstvu grafiky s původní mapou vkládány další vrstvy grafiky obsahující editované prvky mapy. Tento přístup byl ve výsledku mnohem výkonnější a také usnadnil implementaci ostatních funkcionalit, jako je například možnost vrátit provedené akce zpět nebo je opakovat.

Editace mapy probíhá následovně. Při výběru možnosti kreslení do mapy (štětec) nebo mazání změněných prvků (guma) je vytvořena nad mapou nová vrstva grafiky. Při kliknutí a tahu po mapě jsou změněné prvky vykreslovány do této nové vrstvy. Pro každý další tah štětcem či gumou je vytvořena nová PixiJS vrstva a vložená do zásobníku nakreslených vrstev. Díky tomu jsou vytvářeny nové vrstvy s menším počtem prvků nad původní mapou a není nutné aktualizovat vrstvu s původní mapou.

Při využití tohoto přístupu byla velmi jednoduše implementována funkcionalita zpět. Ta probíhá pouhým odebíráním nakreslených vrstev grafiky ze zásobníku a jejich odstraňováním z PixiJS vizualizace. Funkcionalita opakovat je řešená přidáním druhého zásobníku pro vrstvy k opakování. Pokud je vybrána možnost zpět, poslední nakreslená vrstva grafiky je odstraněna z vizualizace, vyjmuta ze zásobníku nakreslených vrstev a vložena do zásobníku pro opakování. Pokud je následně požadováno opakovat akci, poslední vrstva se jednoduše přesune ze zásobníku pro opakování do zásobníku vykreslených vrstev a vykreslí se do vizualizace. Další funkcionalitou pro správné fungování akcí zpět a opakovat je zajištění toho, aby byl zásobník pro opakování vyprázdněn pokaždé, když uživatel provede novou editaci mapy.

Při samotném kreslení štětcem ve vizualizaci mapy se objevil problém s tím, že při rychlejším pohybu kurzoru byly přeskočeny některé prvky mapy. Daný problém nastal proto, že poloha kurzoru nad vizualizací je získávána v diskrétních okamžicích, a dochází tedy ke skokové změně souřadnic. Pro vyřešení problému bylo provedeno uložení posledních souřadnic kurzoru. Poslední a aktuální souřadnice kurzoru jsou následně v programu proloženy přímkou, na jejímž základě jsou dopočítány přeskočené prvky. Odvození vztahů využitých v programu pro proložení dvou bodů je uvedeno v příloze D. Smyčka programu zajišťující kreslení poté projde všechny prvky protínající zmiňovanou přímku, nedojde tak k vynechání žádného prvku. Uvažováno bylo také využití složitějších metod proložení jako jsou spline, polynomy vyšších řádů a další možnosti. V rámci testování proběhlo několik různých pokusů, ze kterých však bylo zjištěno, že proložení přímkou je naprosto dostačující a dosahuje dobrých výsledků.

Během editace je v programu udržováno dvojrozměrné pole obsahující aktuální mapová data odpovídající stavu mapy, který je zobrazen ve vizualizaci. Během procesu kreslení či používaní gumy jsou tato data průběžně aktualizována, a udržována tak aktuální. To však neplatí pro akce zpět a opakovat, kde jsou pouze odebírány nebo přidávány vrstvy grafiky ze zásobníků. Tento problém má dvě potenciální řešení. První řešení spočívá v ukládání nejen vrstvy s grafikou PixiJS, ale i kopií aktuálních mapových dat do zásobníku. Tím by se problém vyřešil a bylo by možné udržovat aktuální mapová data odpovídající vizualizaci i pro akce zpět a opakovat. Popsané řešení by však bylo z hlediska využití paměti velmi náročné. Proto bylo přistoupeno k druhému řešení, kdy se po akci zpět nebo opakovat projdou všechny vrstvy grafiky v zásobníku vykreslených vrstev a na základě toho jsou vytvořena aktuální mapová data. U toho řešení se však objevil problém s výkonem, neboť přepočítaní mapových dat u velkých map trvalo příliš dlouho, a znamenalo by tak snížení uživatelské přívětivosti. Proto bylo přistoupeno k řešení využívající více vláken procesoru. Na webu je pro využití více vláken nutné použití *Web Worker* [6, 45]. Dané řešení zajistí, aby aplikace takzvaně „nezamrzla“ z důvodu plného vytížení hlavního

vlákna. Přepočítání aktuálních mapových dat na základě vykreslených vrstev grafiky totiž proběhne v separátním vlákně. Zrychlení je však možné očekávat pouze na zařízeních, která mají k dispozici alespoň dvě vlákna CPU. Samotná implementace *Web Worker* kódu pro vytvoření aktuálních mapových dat proběhla v rámci třídy `wafer-map-drawing.worker.ts`

Rotace mapy křemíkových desek ve vizualizaci

Pro usnadnění práce s editorem byla přidána funkcionalita sloužící k otáčení mapy s krokem 90° a možnost mapu horizontálně převrátit. Funkce otáčení mapy zajistí, aby si mohl operátor ve výrobě mapu ve vizualizaci otočit tak, jak má ve výrobě fyzicky před sebou křemíkovou desku. To mu ulehčí zakreslování vadných prvků. Uživatel může také požadovat horizontální převrácení mapy, a to v případě, že se dívá na spodní stranu reálné křemíkové desky.

Implementace kódu zajištujícího otáčení a horizontální převrácení mapy je ve třídě `wafer-map-rotator.ts`. Byly zde implementovány metody pro otočení mapových dat o 90° ve směru hodinových ručiček, o 90° proti směru hodinových ručiček, o 180° a pro horizontální převrácení mapy. Ukázka kódu sloužícího k otočení mapy o 90° doprava je uvedena ve výpisu C.4 v přílohách.

Obalování oblastí vyřazených prvků na mapě

Funkcionalita pro obalení vyřazených (*fail*) prvků (angl. *widen fail cluster*) slouží k urychlení práce uživatele při editaci. Jedná se o editační akci požadovanou uživateli, která umožňuje rychle rozšiřovat oblasti vyřazených prvků mapy křemíkové desky. Obalení prvků spočívá ve výběru vyřazeného prvku na mapě uživatelem. Poté je po okraji oblasti vyřazených prvků, ve které se nachází vybraný prvek, přidána vrstva nových vyřazených prvků. Dojde tak k obalení původní oblasti další vrstvou vyřazených prvků. Ukázka takového obalení je vidět na obrázku B.5 v přílohách.

Nejprve bylo obalování vyřazených prvků řešeno rekurzivním algoritmem. Ten procházel osmiokolí prvku mapy, na který uživatel kliknul. Prvky okolí, u kterých bylo zjištěno, že jsou vyřazené (*fail*), byly zařazeny do pole již zkontrolovaných prvků a byl na ně rekurzivně aplikován zde popisovaný algoritmus. Prvky okolí, které jsou funkční (*pass*), byly uloženy do pole prvků určených pro vytvoření požadovaného obalu oblasti. U tohoto algoritmu se však objevil problém v souvislosti s velkými oblastmi, kde rekurze způsobila vyčerpání zásobníku sloužícího k předávání parametrů a návratových adres podprogramu.

Řešení zmíněného problému s rekurzivním algoritmem bylo inspirováno semínkovým vyplňováním [46]. Místo rekurze je využito `while` cyklu a zásobníku, do kterého

se ukládají prvky, které je třeba projít. Díky tomu nemůže dojít k vyčerpání zásobníku pro předávání parametrů a návratových adres podprogramu. Algoritmus se liší v tom, že prvky v osmiokolí, u kterých bylo zjištěno, že jsou vyřazené (*fail*), jsou zařazeny do zásobníku prvků, které mají být zkontrolovány při některé z dalších iterací algoritmu. Zásobník prvků určených ke zkontrolovaní je procházen dříve zmíněným `while` cyklem, dokud nejsou ověřeny všechny prvky oblasti. Výsledný postup je shrnut algoritmem nacházejícím se ve výpisu 6.2.

Výpis 6.2: Algoritmus funkce pro obalení oblasti mapy další vrstvou prvků.

```
1. Vytvoř zásobník, pole zkontrolovaných prvků a pole prvků k vyřazení.
2. Vlož souřadnice [x, y] výchozího prvku mapy do zásobníku.
3. Prováděj, dokud není zásobník prázdný.
    (a) Vyjmi prvek se souřadnicemi [x, y] ze zásobníku.
    (b) Procházej postupně prvek po prvku osmiokolí prvku.
          i. Pokud se prvek o souřadnicích [x, y] nachází mimo mapu,
             přejdi do další iterace cyklu.
         ii. Pokud se prvek již nachází v poli zkontrolovaných prvků nebo
             prvků k vyřazení, přejdi do další iterace cyklu.
        iii. Pokud je prvek "fail", vlož prvek do pole zkontrolovaných
             prvků a do zásobníku.
         iv. Pokud je prvek "pass", vlož prvek do pole prvků k vyřazení.
4. Pole prvků k vyřazení obsahuje prvky obalující původní oblast.
```

Další funkcionality vizualizace map křemíkových desek

Mezi další funkcionality vizualizace map patří výběr prvku mapy. Implementace výběru prvku je provedena ve službě `bin-selecting-overlay.service.ts` a její funkce je taková, že pokud uživatel klikne myší na prvek mapy, je daný prvek zvýrazněn a je publikována událost obsahující informace o daném prvku. Mimo výběr myší je možné využít také klávesnici a pohybovat se po mapě za pomoci šipek doprava, doleva nahoru a dolů.

Vizualizace křemíkových desek obsahuje také mřížku. Ta je implementována v rámci služby `auxiliary-grid-overlay.service.ts`. Mřížka je vykreslována jako vrstva grafiky PixiJS přes mapu, a proto je její vykreslení velmi rychlé. Mřížka umožňuje také základní nastavení. Lze zvolit velikost buněk mřížky a její počáteční pozici v ose x a ose y. Dále byl pro jednodušší orientaci v mapě při editaci do vizualizace vložen kříž zobrazující aktuální polohu kurzoru. Implementace kurzoru je provedena v rámci služby `cross-pointer-overlay.service.ts`. Kurzor je ve vizualizaci zobrazován pouze v režimu editace, a pokud se ukazatel myši nachází nad vizualizací.

6.2 Realizace backend části aplikace

Backend část aplikace byla implementována na základě teoretického návrhu popsaného v kapitole 5.2. Pro realizaci byl vybrán programovací jazyk Java verze 17. V jazyce Java je ve společnosti onsemi napsána většina existujících aplikací z oblasti testování polovodičů. Aby tedy bylo možné využít již existující knihovny používané pro správu mapových souborů a jejích metadat, bylo nutné využít jazyka Java. Dále bylo v kapitole s teoretickým návrhem prezentováno, že frontend bude s backend částí aplikace komunikovat za pomocí REST API. Pro implementaci aplikace bylo na základě teoretické rešerše rozhodnuto o využití Spring Boot frameworku, aby byl vývoj co nejrychlejší a implementace co nejjednodušší. Tento framework totiž umožňuje vytvářet velice jednoduše REST API za pomoci anotací. Pro zdrojový kód celé backend části byla vytvořena dokumentace nacházející se v elektronické příloze (příloha F).

Architektura backend části aplikace byla rozvržena do dvou vrstev. První vrstva aplikace obsahuje kontroléry definující REST API rozhraní určené pro komunikaci s frontend části aplikace. Druhá vrstva obsahuje služby poskytující implementaci funkcionalit backend části a kód zajišťující komunikaci se systémy MapVault a MapSpyAccesspoint.

6.2.1 Vrstva obsahující služby

Služby byly umístěny do balíčku `service` a zajišťují implementaci logiky backend části aplikace. Zmíněný balíček obsahuje celkem čtyři služby, které jsou dále detailněji rozebrány.

Služba AccessPointService

První službou je `AccessPointService` určená k práci se systémem MapSpyAccessPoint. Pro komunikaci s REST API systému MapSpyAccessPoint a MapVault je využit reaktivní HTTP klient `WebClient` ze Spring balíčku `spring-webflux`. Jedním z úkolů služby je zajištění načítání dostupných mapových repozitářů. K této službě se váže funkcionalita sloužící k ukládání repozitářů do cache pro případ, že by bylo REST API systému MapSpyAccessPoint nedostupné. Pokud tato situace nastane, data jsou načtena z cache uložené na disku. Samotná cache byla naprogramována v rámci balíčku `cache` za pomocí třídy `InFileCaching`. Implementace byla provedena za použití generických metod, takže je možné danou cache využít pro jakýkoliv model dat. Jedinou podmínkou je, aby data určena ke vložení do cache implementovala rozhraní `Serializable`, neboť data jsou v při vytváření cache serializována. Při následném ukládání serializovaných dat na disk je uložen i čas vytvoření

cache. Díky tomu je možné zajistit, aby cache byla validní jen po definovaný časový interval.

Služba MapVaultService

Služba `MapVaultService` slouží k práci se systémem MapVault. Nachází se zde implementace funkcionalit zajišťujících vyhledávání a získaní mapových metadat na základě vyhledávacího dotazu vytvořeného frontend částí aplikace. Získaná mapová metadata jsou zde převáděna na formát vhodný pro frontend část. Dále služba obsahuje kód pro získaní mapových souborů na základě identifikátoru mapy, kód určený pro manuální nahrání mapového XML souboru do repozitáře či kód pro nahrání dat z mapového editoru frontend části aplikace. Kromě funkcí pro zpracování a vyhledávaní mapových souborů jsou zde metody pro dynamické získávání dostupných metadat, na základě kterých je možné v systému MapVault vyhledávat, dále metody pro ověření kompatibility repozitáře MapVault s aplikací MapSpyWeb či metody pro získání náhledových obrázků vizualizací map křemíkových desek.

Pro práci s mapovými soubory využívá služba `MapVaultService` jinou službu definovanou třídou s názvem `WaferMapService`, která je popsána v následujících odstavcích.

Služba WaferMapService

V rámci služby `WaferMapService` je implementována logika sloužící k práci s mapovými XML soubory křemíkových desek. V první řadě tato služba obsahuje kód pro dekódovaní komprimovaného mapového souboru z formátu Base64 a následné rozbalení ZIP souboru pro získání XML mapového souboru. Dále se zde nachází funkce zajišťující načtení XML souboru pomocí knihovny WaferMapJr, která je určena pro práci s mapovými soubory. Služba také zajišťuje úpravu a přepočítání metadat originálních mapových souborů například po editaci mapy ve frontend části aplikace. Součástí `WaferMapService` jsou i metody sloužící ke zpětné komprimaci mapového souboru do formátu ZIP a pro transformaci mapového XML souboru na formát pole typu `byte`.

Podstatnou částí služby `WaferMapService` jsou také funkcionality zajišťující transformaci mapových dat z XML mapového souboru do formátu vhodného pro přenos do frontend části aplikace. Zde bylo uvažováno, jaký formát bude velikostně nejúspornější pro přenos mapy do frontend části aplikace, jestliže je pro komunikaci využíváno REST API a formátu JSON. Na základě získaných informací byl pro přenos mapových dat zvolen formát dvourozměrného pole celočíselných hodnot, kde každý specifický prvek mapy křemíkové desky odpovídá jedné numerické hodnotě.

6.2.2 Vrstva definující REST API aplikace

Rozhraní REST API aplikace MapSpyWeb bylo podrobně zdokumentováno. Vytvořená dokumentace se nachází v elektronické příloze (příloha F).

Pro definici REST API rozhraní byl v backend části aplikace vytvořen balíček `controller`. Samotné vytvoření REST API proběhlo velmi jednoduše za pomoci Spring anotací (viz kapitola 2.7 a výpis 2.4). V rámci balíčku bylo vytvořeno několik tříd rozdělujících REST API podle funkcionality. Jedná se o třídu `MainController`, která definuje GET metody sloužící k získání seznamu dostupných repozitářů MapVault a pro jejich následné ověření kompatibility. Další třídou je `SystemController` obsahující metody k získání informací o verzi aplikace MapSpyWeb.

Třída `MetadataController` je určena pro práci s metadaty. Obsahuje GET metody pro vyhledávání a získávání potřebných mapových metadat. `MapContent Controller` je třída s implementací GET metod sloužících k získávání mapových souborů jak ve formátu vhodném pro vizualizaci, tak ve formátu pole `byte` určeném k uložení souboru na disku. Nachází se zde i POST metody sloužící k ručnímu nahrání mapového souboru do úložiště MapVault a ke zpracování editované mapy z frontend části a její následné uložení do systému MapVault.

Autentizaci uživatele pomocí id a hesla zajišťuje třída `AuthorizationController`. Kromě autentizace je zde také GET metoda pro získaní privilegií uživatele. Více informací k autentizace a autorizaci je uvedeno v následující podkapitole 6.3.

Pro ulehčení vývoje frontend části aplikace byl v rámci rešerše nalezen a následně využit nástroj OpenApi Generotor [47]. Jedná se o generátor HTTP klienta na základě OpenApi specifikace REST API. Tento generátor byl přidán do aplikace a je schopen vygenerovat kód služeb frontend části aplikace zajišťujících provádění HTTP požadavků na REST API backend části včetně zabezpečení předání potřebných parametrů.

6.3 Autentizace a autorizace uživatele

Pro zajištění autentizace uživatele při práci s REST API aplikace bylo využito Spring funkcionalit nazvaných Spring Security. Zabezpečení aplikace bylo implementováno z toho důvodu, aby bylo uživatelům zabráněno provádět neoprávněné akce. Nejprve je v této podkapitole probráno fungování autentizace a autorizace z pohledu backend části a následné také z pohledu frontend části aplikace.

6.3.1 Autentizace a autorizace backend části aplikace

Pro autorizaci uživatele pomocí uživatelského ID a hesla slouží POST REST API metoda umístěna ve třídě `AuthorizationController`. Přihlašovací údaje jsou vy-

hodnoceny ve službě `AuthorizationService` pomocí Active Directory, a pokud jsou správné, dojde k autorizaci uživatele. Autorizace spočívá ve vygenerovánní JWT (angl. *JSON Web Token*) pro uživatele [48]. JWT je bezpečnostní token, který je možné použít k autorizaci uživatele vůči backend části aplikace MapSpyWeb. Vygenerovaný JWT obsahuje ID uživatele, pro kterého byl vydán, čas, kdy byl token vytvořen, a dobu jeho platnosti. Celý token je následně podepsán algoritmem HMAC SHA-256 za pomocí privátního klíče. Díky tomu je vždy možné ověřit, zda byl token vygenerován pomocí této aplikace a zda nebyl pozměněn. Zmíněný JWT je pak navrácen v těle odpovědi na autentizační HTTP požadavek.

Některé HTTP metody REST API následně vyžadují přítomnost autorizačního tokenu v hlavičce pro jejich úspěšné volání. Konfigurace nastavující, které metody vyžadují tento token, je definována ve třídě `WebSecurityConfig`. Samotné vyhodnocení platnosti tokenu na základě jeho podpisu a času expirace zajišťuje třída `JwtFilter`.

Vyhodnocení uživatelských privilegií k akcím zajišťují metody ve službách backend části aplikace. Těmto metodám je z JWT tokenu předáno uživatelské ID a na základě toho je následně prováděno vyhodnocení privilegii k akci pro daného uživatele. Toto vyhodnocení uživatelských privilegii na backend části lze považovat za bezpečné, neboť není možné jej jednoduše obejít.

6.3.2 Zabezpečení frontend části aplikace

Při přihlášení do aplikace je z její frontend části zavolán autentizační HTTP požadavek na backend část, načež je uložen JWT token určený k autorizaci. Následně jsou dalším HTTP požadavkem s využitím autentizačního tokenu z backend části získána uživatelská oprávnění pro provádění akcí v aplikaci.

JWT token je poté vkládán do hlavičky `Authorization` HTTP požadavků. Za tímto účelem byla ve frontend části aplikace vytvořena funkcionalita `auth-interceptor.ts`, která, pokud je uživatel přihlášen, zajistí automatické vložení JWT tokenu do hlavičky každého HTTP požadavku. Tímto je provedena autorizace uživatele vůči backend části aplikace MapSpyWeb.

Pro vyhodnocovaní uživatelských oprávnění k akcím ve frontend části slouží metoda ve službě `privileges.service.ts`. Tato metoda je volána pokaždé, když chce uživatel provést akci, ke které jsou potřeba speciální oprávnění. Již z principu však toto vyhodnocení oprávnění není bezpečné. Jeho benefitem je pouze zvýšení uživatelské přívětivosti a informování uživatele o jeho oprávněních. Vyhodnocení na frontend části je totiž možné obejít úpravou JavaScript kódu aplikace či vlastním voláním REST API. Bezpečné vyhodnocení oprávnění k akcím tedy zajišťuje, jak již bylo zmíněno dříve, výhradně backend část aplikace.

7 Testování aplikace a ověření funkčnosti

Pro zajištění kvality a správné funkčnosti aplikace je nutné provádět její testování. Pro správné fungování kritických částí aplikace se využívaly unit testy ověřující častí kódu, u kterých se mohla pravděpodobně vyskytnou chyba. Dále probíhalo manuální testování funkcionalit aplikace přímo přes vyvíjené uživatelské rozhraní. Posledním využitým typem ověřování bylo testování frontend části za pomocí *mock* REST API. Všechny zmíněné druhy testování jsou blíže popsány v následujících odstavcích.

7.1 Testování softwaru aplikace

Pro otestování softwarové implementace byly využity unit testy. Jak již bylo uvedeno, jejich cílem je otestovat kritický kód, u kterého je pravděpodobné, že by se mohla objevit chyba v implementaci, a tuto chybu nebylo možné jednoduše odhalit ručním testováním. Příkladem můžou být unit testy ve frontend části aplikace, které byly napsány za využití testů frameworku Angular [8].

Jednou z důležitých funkcionalit frontend části je vyhodnocování privilegií uživatele ve službě `privileges.service.ts`. Privilegia pro uživatele však mohou být velmi specifická a při jejich vyhodnocování je kontrolováno mnoho metadat mapových souborů různých datových typů (text, pravdivostní hodnota, číslo a datum). Privilegia mohou obsahovat mnoho požadavků na metadata definovaných logickými výrazy AND a OR. Kromě toho obsahuje každý takový požadavek operátor. Pro datový typ string obsahující text jsou to například operátory rovná se, nerovná se, obsahuje, neobsahuje a začíná. U čísel je to pak rovná se, nerovná se, větší a menší. Ze zde uvedených informací je jasné, že funkce sloužící pro vyhodnocení privilegii zpracovává různé datové typy, operátory a logické výrazy. Z toho důvodu není možné jednoduše otestovat všechny kombinace vstupních parametrů ručně. V takovém případě je velmi účinně využití automatizovaných testů.

Proto byl vytvořen test `privileges.service.spec.ts`. Ten obsahuje kód a definici testovacích dat, která vygeneruje celkem 186 testovacích případů s různými kombinacemi datový typů a operátorů. Bylo tak možné důkladně ověřit funkčnost metody sloužící pro vyhodnocovaní privilegii. Vyskytlo se několik případů, ve kterých vyhodnocování neprobíhalo správně, a díky popsanému testu byly dané případy jednoduše identifikovány a opraveny.

Dalším příkladem může být test `map-detail-filtering.service.spec.ts`. Ten slouží pro ověření správné funkčnosti filtrů podle různých typů map a jejich metadat. Jedná se například o filtr odstraněných map, posledních verzí map či map se specifickým příznakem. I zde byla díky testu opravena funkcionalita jednoho filtru, který nefungoval korektně.

7.2 Manuální testování aplikace

Manuální testování je nejčastěji zastoupeným typem testování vyvíjené aplikace. Pro manuální testování frontend části bylo hojně využíváno *mock* REST API popsané v následující podkapitole. Manuální testování spočívalo v otestovaní každé nové funkcionality na vytvořených testovacích datech.

Aplikace byla vyvíjena postupně s tím, že všechny funkcionality byly implementovány po malých celistvých částech. Pro každou část pak bylo prováděno testování. První manuální testování se uskutečnilo přímo při vývoji nové funkcionality aplikace. Byly vytyčeny požadavky, které má funkcionalita splňovat, a ty byly řádně otestovány. Zároveň zde byl kladen důraz na to, aby proběhlo otestovaní všech stavů a možných vstupů i výstupů, které daná funkcionalita může nabývat. Většina vzniklých chyb byla odhalena již v této fázi.

Poté byla funkcionalita předána konzultantovi z firmy onsemi na další manuální otestování, a proběhlo tak následující kolo testů dané funkcionality. Skutečnost, že manuální testování probíhalo minimálně dvakrát, skýtá výhodu, neboť při testováním konzultantem ze společnosti onsemi bylo zajištěno otestování dalších možných stavů, které nemusely být předchozími testy pokryty.

Pro testování REST API backend části aplikace byl využit softwarový nástroj Postman [49]. Ten obsahuje HTTP klienta, který zprostředkovává testování a ladění REST API aplikací. Umožňuje také vytvářet všechny typy HTTP požadavků včetně jednoduchého nastavovaní jejich parametrů a hlaviček. Nástrojem Postman bylo tedy testováno REST API a funkcionality backend části aplikace.

7.3 Testování frontend části pomocí mock REST API

Pro řádné otestování frontend části aplikace bylo potřeba funkční REST API backend části, které však není při vývoji vždy k dispozici, a proto se přistoupilo k využití *mock* REST API. *Mock* REST API vytvoří simulované API s ručně předdefinovanými HTTP metodami a odpověďmi na požadavky. Díky tomu je možné testovat frontend část i bez běžící backend části.

Pro vytvoření *mock* REST API byl využit nástroj Mockoon [50]. Ten umožňuje jednoduché vytváření *mock* REST API za pomocí uživatelského rozhraní. Stačí určit cestu, typ REST metody a definovat, jaká data a hlavičky má metoda vrátit. Výhoda tohoto nástroje spočívá v možnosti vytvářet pravidla a na základě parametrů HTTP požadavku vracet různá data. Tím lze zabezpečit, aby pro specifické parametry byla vrácena odpovídající data.

8 Zpětná vazba a připomínky zadavatele

Aby bylo možné dodat zadavateli kvalitní software, je potřeba shromažďovat zpětnou vazbu k používání vytvořené aplikace. Pro tyto účely byla sbírána zpětná vazba od konzultanta ze společnosti onsemi a od budoucích uživatelů aplikace MapSpyWeb. A to proto, aby byly do aplikace zapracovány připomínky co nejvíce uživatelů a bylo možné aplikaci vyladit k dosažení maximálního přínosu pro uživatele.

8.1 Zpětná vazba od konzultanta

Bylo již zmíněno, že důležitá část zpětné vazby byla získávána od konzultanta. Konzultant má zkušenosti s reálnými požadavky na aplikaci MapSpyWeb mezi uživateli, a navíc je sám uživatelem této aplikace. Proto při vývoji probíhalo časté předvádění funkcionalit konzultantovi, aby bylo zajištěno, že je vše implementováno opravdu tak, jak je požadováno. Dále byl vytvořen testovací sever, na který byla během vývoje každý týden nahrána verze aplikace obsahující aktuálně dokončené funkcionality. Ty konzultant pravidelně zkoušel a poskytoval k nim připomínky a návrhy na zlepšení. Všechny připomínky pak byly zapracovány do aplikace a znovu prezentovány zadavateli.

8.2 Zpětná vazba od budoucích uživatelů

Dále byla sbírána zpětná vazba od zaměstnanců, kteří budou aplikaci využívat. Vždy, když bylo do aplikace implementováno několik nových funkcionalit, budoucí uživatelé aplikace obdrželi e-mail s oznámením, že byla na testovací server nasazena verze aplikace MapSpyWeb obsahující nové funkcionality. E-mail zahrnoval žádost o otestování aktuální verze a také o zaslání svých připomínek, případně nápadů na vylepšení. Takto získaná zpětná vazba s připomínkami je shrnuta v přehledové tabulce 8.1. Hodnocení a připomínky jsou zde rozdělena do několika kategorii a u některých je v tabulce 8.1 uvedena poznámka, jak je možné konkrétní připomínky řešit, případně jak byly vyřešeny. Celkem byla během vývoje získána zpětná vazba od 10 uživatelů aplikace, přičemž někteří se vyjadřovali v průběhu vývoje vícekrát.

Za největší přínos nové aplikace uživatelé označují rychlost aplikace a také to, že není potřeba instalovat specifickou verzi Java JRE pro její spuštění. Dále uživatelé zmiňovali jako velký benefit integrovaný mapový editor, díky němuž nemusí spouštět pro editaci map jinou aplikaci. Prvek, který dle uživatelů dokáže ušetřit mnoho času při vyhledávání map, je pokročilé vyhledávání umožnující hledat podle mnoha různých parametrů. Velmi kladnou zpětnou vazbu získala také funkcionalita, pomocí které lze zobrazit mapový soubor bez nutnosti stažení na disk. Někteří uživatelé

Tab. 8.1: Tabulka obsahující souhrn získané zpětné vazby od uživatelů aplikace MapSpyWeb. Zeleně jsou označeny funkcionality zapracované do aplikace a vyřešené připomínky. Žlutě je označená zpětná vazba navržená k budoucímu zapracování.

Typ zpětné vazby	Počet	Popis zpětné vazby	Poznámka tvůrce diplomové práce a aplikace MapSpyWeb
Uživatelské rozhraní	1	Přidat název dávky a číslo desky do okna s vizualizací, aby bylo jasné, jaká mapa je vizualizována.	Přínosem je lepší přehled při práci s aplikací. Bylo implementováno.
	1	Přejmenovat okno "Basic result" na "Search result".	Přínosem je lepší porozumění funkce okna s výsledky vyhledávání.
	1	Překlep v názvu tlačítka.	Název byl opraven.
	1	Udělat výraznější indikátor horizontálního převrácení mapy.	Přínosem je možné zmenšení chybovosti při práci s mapovým editorem.
Používání aplikace	1	Automatické přihlášení uživatele.	Lze řešit uložením přihlašovacích údajů ve webovém prohlížeči.
	1	Vylepšit zpracování chyb, aby aplikace dávala více informací, proč selhalo nahrávání mapy.	Přínosem je informovanost uživatele o problémech.
	1	Uložení posledního zvoleného repozitáře i do další relace.	Přínosem je zrychlení práce s aplikací. Odpadá nutnost pokaždé vybírat repozitář.
	1	Větší štětec v mapovém editoru pro kreslení na velkých mapách.	Lze řešit přidáním štětce s uživatelsky nastavitelnou velikostí.
Výkon aplikace	1	Pomalé načítání dat z repozitářů ve výrobních místech v Asii.	Způsobeno tím, že testovací server je umístěn v USA. Budou přidány servery v EU a Asii.
Chyby v aplikaci	1	Čísla v tabulkách se neřadí numericky, ale abecedně.	Vyřešeno vytvořením vlastní řadící funkce místo využívaní funkce poskytnuté tabulkou.
	1	Souřadnice bodů na mapě by neměly být záporné pro jiné souřadnicové systémy.	Je potřeba upravit tak, aby zobrazená data souhlasila s reprezentací využívanou v onsemi.
Návrh nových funkcí	1	Přidat pokročilé vyhledávání podle různých metadat.	Pokročilé vyhledávání implementováno.
	1	Umožnit vyhledávání více hodnot pro jednu položku.	Bylo implementováno.
	1	Přidat do tabulek řazení podle více sloupců.	Bylo implementováno.
	1	Přidat okno s přehledem vizualizací všech map v rámci dávky.	Okno bylo implementováno.
	1	Přidat okno s grafem výtěžnosti křemíkových desek.	Okno bylo implementováno.
	1	Přidat filtry do všech sloupců tabulek, aby bylo možné jednoduše zobrazené výsledky filtrovat.	Filtry byly implementovány.
	1	Přidat do vizualizace další data z mapového souboru, specificky počet měření každého prvku mapy.	Lze vyřešit přidáním vrstvy, ve které by byly čísla s počtem měření pro každý prvek mapy.
	1	Umožnit změnu barev v mapě i v přehledu vizualizací všech map v rámci dávky.	Přínosem je možnost většího uživatelského přizpůsobení.
	2	Umožnit skládat mapy na sebe a zvýrazňovat stejné defekty v těchto mapách.	Přínosem je možnost zvýraznění opakujících se chyb na křemíkových deskách.
Přínos aplikace	3	Rychlost aplikace je velmi dobrá, mnohem rychlejší než práce se starými aplikacemi.	-
	3	Není potřeba instalovat Java JRE.	-
	1	Integrovaný mapový editor.	-
	2	Pokročilé vyhledávání zjednodušuje práci.	-
	1	Obalení prvků mapy další vrstvou se velmi hodí a funguje i na opravdu velkých mapách.	-
	1	Zobrazení mapy bez nutnosti stažení na disk, doteď musel uživatel stahovat obrovské množství map na disk, aby zkontroloval data v souboru.	-
Nepochopení funkcionality	1	Uživatel nevěděl, že zde existuje tlačítko na restartování rozvržení oken na stránce.	Uživateli bylo vysvětleno, jak restartovat rozvržení oken v aplikaci.
	1	Uživatel nevěděl, jak použít ve vyhledávání operátor "Range" pro datový typ datum.	Uživateli bylo vysvětleno, jak pracovat s pokročilým vyhledáváním.
	1	Uživatel nevěděl, jak zajistit, aby do příští relace zůstaly uloženy sloupce v tabulkách a rozložení UI.	Uživateli bylo vysvětleno, že uživatelské předvolby se ukládají pouze, pokud je přihlášen.

díky tomu nebudou muset stahovat obrovské množství mapových souborů na disk jen proto, aby v nich něco zkontrolovali.

Objevila se i výtka k výkonu aplikace, konkretně k rychlosti načítaní dat u uživatelů aplikace z Asie. Tento problém však nesouvisí s aplikací samotnou, nýbrž s jejím nasazením na server v USA. Pokud se uživatel v Asii snažil vyhledávat, požadavek na vyhledávání byl odeslán z Asie na server do USA a ze serveru v USA byl prohledán mapový repozitář v Asii. Následně byla nalezená data odeslána zpět na server do USA a z něj nakonec do Asie k uživateli. Pro vyřešení problému bylo navrženo vytvoření 3 instancí aplikace MapSpyWeb, a to jedné v USA, druhé v EU a třetí v Asii.

V oblasti používání uživatelského rozhraní bylo zaznamenáno několik návrhů na zlepšení. Řadí se mezi ně návrh na přejmenování několika prvků aplikace či přidání více informací o vizualizované mapě. Na základě zpětné vazby byl také zvýrazněn indikátor horizontálního převrácení mapy (sytě červeným nebo zeleným podbarvením), a změněna barva ukazatele pozice *flat* (zkosené strany křemíkové desky). Ukázka podbarvení zmíněných indikátorů je znázorněna v příloze na obrázku B.8.

Ohledně používaní aplikace se objevil návrh na automatické přihlášení. Uživatel však může využít uložení přihlašovacích údajů ve webovém prohlížecí nebo správci hesel. To je vhodné samozřejmě jen za předpokladu, že zaměstnanec má pracovní počítač, který není sdílený. Dalším přínosným návrhem je uložení posledního zvoleného repozitáře do dalšího spuštění aplikace. Tato připomínka byla navržena k zapracovaní a technicky by byla řešena uložením repozitáře do úložiště LocalStorage.

Jinou kategorií zpětné vazby jsou návrhy nových funkcionalit. Uživatelé zde navrhli hned několik funkcionalit, které by chtěli do aplikace přidat. Několik z nich již bylo do aplikace zapracováno. Jedná se o okno s přehledem vizualizací všech map v rámci dávky a okno s grafem výtěžnosti křemíkových desek. Další realizovanou funkcí je přidání filtrů do všech sloupců tabulek, aby bylo možné jednoduše zobrazené výsledky filtrovat. Kromě toho uživatelé navrhovali přidat do vizualizace mapy další vrstvu s daty mapového souboru nebo také umožnit skládání map na sebe se zvýrazňováním stejných defektů na těchto mapách. Tyto i další návrhy byly navrženy k zapracování do aplikace. Změnu podoby u okna s výsledky vyhledávání před a po zapracování zpětné vazby je možné vidět na obrázku B.4 v přílohách.

V rámci zpětné vazby došlo i ke třem situacím, kdy nebylo zcela pochopeno uživatelského rozhraní aplikace. U jednoho uživatele se jednalo o neporozumění nastavení pokročilého vyhledávání dle rozsahů datového typu datum. V dalších dvou případech se jednalo o nevědomost, že v aplikaci existuje tlačítko na obnovení výchozího rozložení oken, a o neznalost principu uložení nastaveného rozložení oken. Těmto uživatelů byla nepochopená funkcionalita vysvětlena a zároveň byl na základě jejich zpětné vazby vhodně upraven interní uživatelský manuál k aplikaci.

9 Demonstrace použití aplikace ve výrobě

V rámci této kapitoly je rozebrána demonstrace funkčnosti aplikace MySpyWeb na produkčních datech z výroby. Dále je zde popsáno, jak je aplikace nasazena na server pro produkční využití. Poslední část kapitoly je věnována popisu typických případů použití této aplikace.

Testování funkčnosti aplikace na datech ve výrobě prokázala, že testovací data využita při vývoji dostatečně pokryla scénáře a formát produkčních dat ve výrobě. Při používání aplikace MapSpyWeb uživateli ve výrobě se totiž neobjevily žádné problémy s její funkčností.

Pro produkční účely je ze zdrojového kódu aplikace MapSpyWeb sestaven WAR soubor obsahující jak backend, tak frontend část aplikace. Soubor WAR je obdobou souboru JAR, který vzniká sestavením Java aplikace, s tím rozdílem, že WAR soubor je určen pro nasazení na webový server. Definici pro sestavení aplikace obsahuje soubor `pom.xml`. Samotné sestavení celé aplikace probíhá za pomocí Apache Maven [51]. Sestavený WAR soubor je následně nasazen na webový server Apache Tomcat [52].

Webový server Apache Tomcat byl na produkčním serveru společnosti onsemi nakonfigurován tak, aby využíval šifrované komunikace pomocí protokolu HTTPS (angl. *Hypertext Transfer Protocol Secure*). Díky tomu je zajištěn zabezpečený přenos dat a přihlašovacích údajů mezi klientem a serverem. Jako komunikační protokol pro přenos dat mezi webovým serverem a klientem bylo zvoleno HTTP/2.

9.1 Využití aplikace uživatelem

Následující odstavce se zabývají typickými scénáři, při kterých najde aplikace MapSpyWeb uplatnění ve výrobě. Uživatelé této aplikace spadají do tří hlavních skupin. Jedná se o podporu výroby, operátory ve výrobě a inženýry výroby. Pro každou skupinu je dále uveden příklad použití aplikace MapSpyWeb.

Využití aplikace podporou výroby

Skupina zaměstnanců podpory výroby zajišťuje podporu při výrobě čipů na křemíkových deskách pro operátory a inženýry výroby.

Tato skupina uživatelů aplikace použije primárně pokročilé vyhledávaní, ve kterém je možné vyhledávat podle specifických metadat map křemíkových desek. Uživatel má tedy možnost vyhledat přesně ty mapy, které potřebuje. Zaměstnanci podpory výroby využijí také tabulky pro zobrazení metadat jednotlivých dávek a map. Zde jednoduše zjistí informace, které potřebují k řešenému problému. Například,

kdo desku vyrobil, kdy bylo provedeno jaké testování a mnoho dalších informací. Dále, pokud například při distribuci desky do jiného výrobního místa nesouhlasí počet funkčních čipů na desce, může podpora výroby dohledat, zda se opravdu jedná o správnou fyzickou desku a nedošlo například při transportu k záměně za jinou desku. Podpora také může analyzovat, jak se v čase vyvíjely mapy jedné křemíkové desky v průběhu výroby. K tomu uživatel využije okno s přehledem vizualizací map.

Využití aplikace operátorem ve výrobě

Další skupinou zaměstnanců, která bude aplikaci MapSpyWeby využívat, jsou operátoři ve výrobě. Ti pracují se stroji, které zajišťující výrobu polovodičových součástek, a přicházejí tak s křemíkovými deskami fyzicky do kontaktu.

Operátor typicky zajišťuje přenos křemíkových desek do dalšího stádia výroby pomocí speciálního nástroje. Při tom vizuálně přímo, nebo pomocí mikroskopu prohlédne křemíkové desky, u kterých má podezření, že by mohly být poškozeny mechanicky či obsahovat jiný defekt. Pokud na desce operátor najde vadu, vyhledá si mapu v aplikaci MapSpyWeb podle názvu dávky nebo jiného parametru. Následně otevře mapu v mapovém editoru a ručně zde zakreslí poškozenou část desky, čímž dané součástky (prvky mapy) označí jako vyřazené. Operátor tady využije funkce pro otáčení mapy a horizontální převrácení, aby si v editoru mapu otočil do stejné pozice, v jaké má křemíkovou desku fyzicky před sebou. To mu značně usnadní zakreslování vadných prvků do mapy křemíkové desky.

Využití aplikace inženýrem výroby

Poslední skupinou uživatelů jsou inženýři výroby. Ti mají na starost výrobu určitého produktu (specifického typu součástek). Úkolem inženýra výroby je ladění výroby pro zajištění maximální efektivity výroby čipů, které má na starost.

Inženýra výroby typicky nejvíce zajímá grafická reprezentace dat, ať už jde o vizualizaci mapy, souhrn vizualizací map v rámci dávky nebo vývoj výtěžnosti čipů. Pracuje často s vizualizací mapy, kde hledá určité chybové vzory. Využívá legendu mapy, díky které může zjistit, jaký typ chyby se v jaké části mapy vyskytuje. Na základě toho může následně optimalizovat výrobní proces. V okně s grafem, který graficky reprezentuje výtěžnost, může inženýr výroby zjišťovat, jak se vyvíjela výtěžnost specifických křemíkových desek v průběhu výrobního procesu. Velmi užitečným nástrojem pro inženýry výroby je také okno s přehledem vizualizací map křemíkových desek. Zde je možné si zobrazit všechny desky z jedné či více dávek najednou. Díky tomu mohou inženýři na deskách hledat opakující se vzory, které se vyskytují nad různými mapami. A následně hledat příčinu, proč při výrobě vznikají vadné čipy, a v závislosti na tom provádět optimalizaci výroby.

Závěr

Výsledkem této diplomové práce je, na základě provedené literární rešerše, realizovaný teoreticky návrh a následná implementace webového systému pro vizualizaci a anotaci dat z výroby polovodičových součástek. Systém byl nazván MapSpyWeb. Společnost onsemi, pro kterou je systém určen, se zbývá výrobou polovodičových součástek. Během výroby jsou tyto součástky na křemíkových deskách mnoha způsoby měřeny a kontrolovány, zda splňují požadované parametry. Získaná data jsou ukládána do map defektů křemíkových desek. Systém MapSpyWeb zajišťuje prohledávání, vizualizaci a editací těchto map a jejich metadat.

První kapitola diplomové práce se zabývá společností onsemi a vysvětlením základních principů výroby a testování polovodičových součástek. Kapitola číslo 2 pak rozebírá technologie vhodné k realizaci anotačních a vizualizačních systému na webu. Popis stávajících řešení ve společnosti onsemi, se kterými bude nová vizualizační aplikace spolupracovat, shrnuje kapitola číslo 3. Jedná se především o systémy MapVault a MapSpyAccessPoint.

Na základě konzultací se zadavatelem byly specifikovány detailní požadavky na vizualizační a anotační systém. Jedná se o požadavky na funkcionalitu, kompatibilitu s již existujícími systémy ve společnosti a v neposlední řadě také požadavky na webové prohlížeče, ve kterých musí aplikace správně fungovat. Základem je, aby aplikace byla webová a byla v ní umožněna interaktivní vizualizace a anotace map křemíkových desek o velikosti až 1 000 x 1 000 prvků. Detailně dané požadavky popisuje kapitola 4.

Během návrhu a následné implementace systému bylo dbáno na dodržování firemních zásad správného vývoje a udržování zdrojových kódů. Dále byl kladen důraz na využití moderních technologií a na možnost snadného dalšího rozšíření aplikace. Pro frontend, backend i REST API aplikace byla vytvořena dokumentace umístěná v elektronické příloze (příloha F). Implementace řešení, včetně jeho návrhu a testování, byla provedena během 1 100 hodin v rámci 11 kalendářních měsíců.

Teoretickým návrhem celého řešení se zabývá kapitola 5. Na základě posouzení požadavků zadavatele bylo navrženo řešení, kdy je aplikace rozdělena na backend část nasazenou na serveru a frontend část běžící u uživatele. Backend část aplikace zabezpečí primárně zpracování dat pro frontend část a funkcionality pro komunikaci s ostatními systémy, jako je například úložiště map křemíkových desek. Schéma navržených funkcionalit pro backend aplikace je znázorněno na obrázku 5.3. Frontend aplikace má dle návrhu zajišťovat vizualizaci všech požadovaných dat s možností jejich editace a zpracování uživatelských akcí (viz schéma na obrázku 5.7). V rámci kapitoly 5 byl následně proveden i návrh uživatelského rozhraní webové části aplikace.

Praktická část diplomové práce je popsána v kapitole 6. Celá praktická realizace vychází z prezentovaného teoretického návrhu. Pro realizaci frontend části aplikace byl využit framework Angular 13. Popis vytvořeného uživatelského rozhraní a jeho implementace je uveden v kapitole 6.1. Výsledné uživatelské rozhraní aplikace se skládá ze 4 oken. Nachází se zde okno pro přihlášení (obr. 6.1) uživatele, okno pro vyhledávání dle různých parametrů (obr. 6.2), okno se souhrnem výsledků vyhledávání (obr. 6.3) a okno sloužící k anotaci map křemíkových desek (obr. 6.4).

V rámci praktické části práce proběhla i implementace vizualizace map a jejich editoru. Na základě testovacích programů a praktického testovaní různých technologii vizualizace map křemíkových desek (viz kapitola 6.1.5) byla pro implementaci vybrána technologie WebGL a knihovna PixiJS. Tato metoda dosahovala u mapy velikosti 1 000 x 1 000 prvků průměrného času překreslení 22 ms, využití operační paměti 885 MB a využití grafické paměti 670 MB. Výsledná vizualizace mapy je naprosto plynulá a bez jakékoli znatelné odezvy na uživatelské akce. Výsledky testování metod vizualizace jsou shrnuty v tabulce 6.1 a zpracovány do grafů na obrázcích 6.6 a 6.7.

Vizualizační část aplikace byla implementována jako samostatný modul, který zajišťuje všechny požadované funkcionality vizualizace. Mapa křemíkové desky je vykreslována jako vrstva grafiky pomocí knihovny PixiJS. Pro editaci bylo ve vizualizaci z výkonnostních důvodů využito vykreslování editovaných prvků mapy křemíkové desky do nových grafických vrstev nad původní mapu. Díky tomu bylo také možné za pomoci zásobníků snadno implementovat funkcionality zpět a znovu. Došlo k zakomponování i dalších funkcionalit, lze mínit například možnost rotace map nebo obalování oblastí mapy křemíkové desky dalšími prvky (viz str. 77).

Backend část aplikace byla vytvořena za využití programovacího jazyka Java 17 a Spring Boot frameworku. Obsahuje REST API vrstvu pro komunikaci s frontend částí, služby zajištující komunikaci s ostatními systémy a provádění funkcionalit, které má backend na starost dle diagramu 5.3 (viz str. 46). Jedná se primárně o funkcionality související s mapovými soubory a jejich metadaty (název dávky, ID, výtěžnost a jiné) a dále o funkcionality sloužící k načítaní a zpracování uživatelských oprávnění.

Pro zabezpečení REST API aplikace je využito Spring Security. Autorizace uživatele vůči backend části probíhá pomocí bezpečnostního JWT [48] tokenu podepsaného algoritmem HMAC SHA-256. Vyhodnocení uživatelských oprávnění se uskutečňuje jak na frontend části z důvodu informování uživatele o jeho právech, tak na backend části z důvodu bezpečnosti. Zabezpečení aplikace detailně popisuje kapitola 6.3.

Celá aplikace a všechny její funkcionality byly pravidelně testovány z důvodu zajištění kvality a správné funkčnosti. Testování probíhalo za využití unit testů, ma-

nuálního testování a testování pomocí mock REST API. Detailní informace o testování jsou shrnuty v kapitole 7. Výhodou využitých unit testů je možnost jejich automatického spouštění v rámci průběžné integrace (angl. *continuous integration*) při každém sestavení aplikace, čímž lze zaručit její správnou funkčnost i v budoucnu.

Aby bylo zajištěno, že bude zadavateli dodán kvalitní software, který splňuje všechny jeho požadavky, bylo třeba průběžně shromažďovat zpětnou vazbu. Zpětná vazba byla sbíraná od konzultanta a budoucích uživatelů aplikace. K aplikaci se vyjádřilo celkem 10 uživatelů. Jejich připomínky jsou shrnuty v kapitole 8 a graficky seřazeny dle několika kategorií v tabulce 8.1. Mnoho podnětů ze zpětné vazby bylo do aplikace zapracováno a další byly zadavateli navrženy k zapracování v budoucnu. Uživatelé s aplikací nebyli předem seznámeni, i přesto se v ní byli schopni intuitivně orientovat. Objevily se pouze 3 otázky pramenící z nepochopení funkcionality (viz tab. 8.1).

Poslední kapitola diplomové práce číslo 9 rozebírá využití aplikace ve výrobě. Existuje velké množství potenciálních uživatelů mezi inženýry výroby, podporou výroby a operátory. Aplikace uživatelům urychlí práci ve srovnání se stávajícím řešením. Dále aplikace umožní odhalovat a analyzovat chyby vznikající při výrobě a na jejich základě optimalizovat výrobní procesy, zlepšovat spolehlivost vyrobených součástek a zvyšovat ziskovost. Ukázku fungování aplikace je možné shlédnou na videu v elektronické příloze (příloha F)

V rámci diplomové práce bylo pracováno na všech vytyčených cílech, to však neznamená, že u aplikace není prostor pro další vylepšení a funkcionality. Z toho důvodu bylo kromě získaných návrhů na zdokonalení od uživatelů (kapitola 8) navrženo ještě několik možných dalších zlepšení aplikace.

Prvním návrhem je přidání dialogového okna pro automatické hlášení chyb, problémů či připomínek. Princip by měl spočívat v tom, že pokud se v aplikaci objeví chybové hlášení, uživatel bude moci kliknout na tlačítko „nahlásit chybu“, případně ručně zvolit možnost „nahlásit připomínku“. Poté se mu otevře dialog, kde lze vložit svou poznámku a snímek obrazovky. Po odsouhlasení dojde k odeslání chybového hlášení na e-mail správce aplikace. Toto řešení by napomohlo sbírání informací o chybách v aplikaci, problémech a dalších nápadech na vylepšení.

Druhý návrh na vylepšení se týká vizualizace map křemíkových desek. Zde se může objevit situace, kdy uživatel ve vizualizaci pracuje s velkou křemíkovou deskou, přičemž využije možnosti přiblížení mapy. Po této akci uvidí pouze detail malé části mapy. Problémem tedy může být, že se uživatel nebude schopen v mapě orientovat a nebude vědět, v jaké části mapy se nachází. V této situaci by mohla pomoci minimapa, umístěná v rohu vizualizace. Zde by uživatel viděl náhled mapy křemíkové desky a v ní obdélník, který by znázorňoval, jakou část mapy zrovna ve vizualizaci vidí.

Poslední navrhované zlepšení se týká souhrnné vizualizace všech map křemíkových desek pro vybrané dávky. Aktuálně jsou v tomto přehledu zobrazeny obrázky obsahující vizualizaci křemíkových desek, avšak pro přehled není možné zobrazit legendu. Pokud tedy chce uživatel zjistit, jaké typy vad se na desce vyskytují, musí si desku zobrazit pomocí plnohodnotné vizualizace. Pokud by do přehledu byla přidána legenda obsahující prvky ze všech vyobrazených map, mohl by uživatel jednoduše zjistit, o jaký typ prvku na mapách se jedná, aniž by musel otevírat plnohodnotnou vizualizaci.

Literatura

[1] CROWDER, T. J. *JavaScript: the new toys*. Hoboken, NJ: Wrox, 2020. ISBN 978-1-119-36797-6.

[2] VANDERKAM, Dan. *Effective TypeScript: 62 specific ways to improve your TypeScript*. Sebastopol, CA: O'Reilly Media, 2019. ISBN 978-1-492-05374-3.

[3] ONSEMI. O společnosti. *Onsemi v České republice* [online]. [cit. 2022-12-29]. Dostupné z: `https://kariera-onsemi.cz/o-nas/`

[4] ONSEMI. IT centrum sdílených služeb. *Onsemi v České republice* [online]. [cit. 2023-05-10]. Dostupné z: `https://kariera-onsemi.cz/it-centrum-roznov-o-nas/`

[5] MORENO-LIZARANZU, Manuel a Federico CUESTA. Improving Electronic Sensor Reliability by Robust Outlier Screening. *Sensors* [online]. 2013, **13**(10), 13521-13542 [cit. 2023-05-03]. ISSN 1424-8220. Dostupné z: doi:10.3390/s131013521

[6] FLANAGAN, David. *JavaScript: the definitive guide*. Seventh edition. Sebastopol: O´Reilly, 2020. ISBN 978-1-491-95202-3.

[7] CHERNY, Boris. *Programming TypeScript: making your JavaScript applications scale*. Sebastopol, CA: O'Reilly Media, 2019. ISBN 978-1-492-03765-1.

[8] *Angular 13: documentation* [online]. c2010-2022 [cit. 2023-03-17]. Dostupné z: `https://v13.angular.io/docs`

[9] BAMPAKOS, Aristeidis a Pablo DEELEMAN. *Learning Angular*. Third edition. Birmingham: Packt Publishing, 2020. ISBN 978-1-83921-066-2.

[10] Ahead-of-time (AOT) compilation. *Angular* [online]. c2010-2022 [cit. 2022-11-05]. Dostupné z: `https://v13.angular.io/guide/aot-compiler`

[11] DEAN, John. *Web programming with HTML5, CSS, and JavaScript*. Burlington, [2019]. ISBN 978-128-4091-793.

[12] LARSEN, Rob. *Mastering SVG: Ace web animations, visualizations, and vector graphics with HTML, CSS, and JavaScript*. Birmingham: Packt Publishing, 2018. ISBN 978-1-78862-674-3.

[13] FULTON, Steve a Jeff FULTON. *HTML5 Canvas*. Second Edition. Sebastopol: O-Reilly Media, 2013. ISBN 978-1-449-33498-7.

[14] MATSUDA, Kouichi a Rodger LEA. *WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL*. Michigan: Addison-Wesley Professional, 2013. ISBN 978-0-321-90292-4.

[15] DIEGO, Cantor a Jones BRANDON. *WebGL Beginner's Guide: Become a master of 3D web programming in WebGL and JavaScript*. Birmingham: Packt Publishing, 2012. ISBN 978-1-84969-172-7.

[16] SPUY, Rex van der. *Learn Pixi.js: Create Great Interactive Graphics for Games and the Web*. California: Apress, 2015. ISBN 978-1-4842-1095-6.

[17] PixiJS Guides: Architecture Overview. *PixiJS* [online]. c2022 [cit. 2022-12-29]. Dostupné z: `https://pixijs.io/guides/basics/architecture-overview.html`

[18] PATNI, Sanjay. *Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS*. California: Apress, 2017. ISBN 978-1-4842-2664-3.

[19] MASSÉ, Mark. *REST API Design Rulebook*. Sebastopol: O'Reilly, 2012. ISBN 978-1-449-31050-9.

[20] GILLIS, Alexander. REST API (RESTful API). *TechTarget* [online]. c2019 - 2023, September 2020 [cit. 2023-05-02]. Dostupné z: `https://www.techtarget.com/searchapparchitecture/definition/RESTful-API`

[21] COSMINA, Iuliana. *Java 17 for Absolute Beginners: Learn the Fundamentals of Java Programming*. Second edition. Edinburgh: Apress, 2022. ISBN 978-1-4842-7079-0.

[22] COSMINA, Iuliana. *Java for Absolute Beginners: Learn to Program the Fundamentals the Java 9+ Way*. Edinburgh: Apress, 2018. ISBN 978-1-4842-3777-9.

[23] GUTIERREZ, Felipe. *Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices*. Second edition. Albuquerque: Apress, 2019. ISBN 978-1-4842-3675-8.

[24] WALLS, Craig. *Spring Boot in Action*. New York: Manning Publications, 2016. ISBN 978-1617292545.

[25] *Spring Framework Documentation: 5.3.9* [online]. Pivotal, c2002—2021, aktualizováno 14.7.2021 [cit. 2023-03-14]. Dostupné z: `https://docs.spring.io/spring-framework/docs/5.3.9/reference/html/`

[26] JUSTINMIND. *Justinmind 9.9.4* [software]. c2022 [cit. 2022-12-18]. Dostupné z: `https://www.justinmind.com/`

[27] JETBRAINS. IntelliJ IDEA: the Leading Java and Kotlin IDE. *JetBrains* [software]. c2000-2023 [cit. 2023-04-20]. Dostupné z: https://www.jetbrains.com/idea/download/

[28] MICROSOFT. *Visual Studio Code* [software]. c2023 [cit. 2023-04-20]. Dostupné z: https://code.visualstudio.com/Download

[29] HARTMAN, James. React vs Angular — Difference Between Them. *Guru99* [online]. c2023, March 11, 2023 [cit. 2023-03-17]. Dostupné z: https://www.guru99.com/react-vs-angular-key-difference.html

[30] *React: The library for web and native user interfaces* [online]. c2023 [cit. 2023-03-17]. Dostupné z: https://react.dev/

[31] PrimeNG v13.4.1: PrimeNG is a rich set of open source native Angular UI components. *PrimeNG* [online]. [cit. 2023-03-25]. Dostupné z: https://www.primefaces.org/primeng-v13-lts/#/setup

[32] JONNA, Sudheer a Oleg VARAKSIN. *Angular UI Development with PrimeNG: Build rich and compelling Angular web applications using PrimeNG*. Birmingham: Packt Publishing, 2017. ISBN 978-1-78829-957-2.

[33] Bootstrap v5.1.3. *Bootsrap: Build fast, responsive sites with Bootstrap* [online]. [cit. 2023-03-27]. Dostupné z: https://getbootstrap.com/docs/5.1/getting-started/introduction/

[34] SPURLOCK, Jake. *Bootstrap*. Sebastopol: O'Reilly, 2013. ISBN 978-1-449-34391-0.

[35] RxJS 7.5.2: Reactive Extensions Library for JavaScript. *RxJS* [online]. [cit. 2023-03-20]. Dostupné z: https://rxjs.dev/guide/overview

[36] ZULD, Tiberiu. Angular-gridster2: Angular implementation of angular-gridster. *Github* [online]. 2023 [cit. 2023-03-27]. Dostupné z: https://github.com/tiberiuzuld/angular-gridster2

[37] Window.localStorage. *MDN Web Docs* [online]. c1998—2023, Mar 16, 2023 [cit. 2023-03-27]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

[38] File System Access API. *MDN Web Docs* [online]. c1998—2023, Feb 27, 2023 [cit. 2023-03-29]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/File_System_Access_API

[39] Chart.js 3.9.1. *Chart.js: Simple yet flexible JavaScript charting library for the modern web* [online]. c2014-2023 [cit. 2023-03-30]. Dostupné z: https://www.chartjs.org/docs/3.9.1/

[40] Chartjs-plugin-annotation 2.0.1. *Chart.js: Annotations for Chart.js* [online]. c2016-2021 [cit. 2023-03-30]. Dostupné z: https://www.chartjs.org/chartjs-plugin-annotation/2.0.1/

[41] *Ngx-color-picker: Pure Angular color picker library* [online]. [cit. 2023-04-01]. Dostupné z: https://pivan.github.io/ngx-color-picker/

[42] *Konva.js: HTML5 2d canvas js library for desktop and mobile applications* [online]. c2022 [cit. 2022-12-03]. Dostupné z: https://konvajs.org/docs/

[43] Fundamentals. *Three.js: JavaScript 3D Library* [online]. [cit. 2022-12-31]. Dostupné z: https://threejs.org/manual/#en/fundamentals

[44] FIGATNER, David. Pixi-viewport API Documentation. *Pixi-viewport* [online]. c2021 [cit. 2022-12-29]. Dostupné z: https://davidfig.github.io/pixi-viewport/jsdoc/

[45] Web Workers API. *MDN Web Docs* [online]. c1998—2023, Feb 26, 2023 [cit. 2023-04-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers

[46] ŽÁRA, Jiří, Bedřich BENEŠ a Petr FELKEL. *Moderní počítačová grafika*. Praha: Computer Press, 1998. ISBN 80-722-6049-9.

[47] Typescript-angular Generator. *OpenAPI Generator: Generate clients, servers, and documentation from OpenAPI 2.0/3.x documents* [online]. c2023 [cit. 2023-04-13]. Dostupné z: https://openapi-generator.tech/docs/generators/typescript-angular

[48] JONES, Michael, John BRADLEY a Nat SAKIMURA. *JSON Web Token (JWT)*. May 2015. Dostupné z: doi:10.17487/RFC7519

[49] POSTMAN. Postman: API Platform. *Postman* [software]. c2023 [cit. 2023-04-14]. Dostupné z: https://www.postman.com/downloads/

[50] Mockoon v1.23.0: Create mock APIs in seconds. *Mockoon* [software]. c2017-2023 [cit. 2023-04-14]. Dostupné z: https://mockoon.com/download/

[51] APACHE SOFTWARE FOUNDATION. *Apache Maven 3.8.5* [online]. c2002—2023 [cit. 2023-04-18]. Dostupné z: https://maven.apache.org/download.cgi

[52] APACHE SOFTWARE FOUNDATION. *Apache Tomcat 9.0.73* [online]. c1999-2023 [cit. 2023-04-15]. Dostupné z: `https://tomcat.apache.org/download-90.cgi`

Seznam symbolů a zkratek

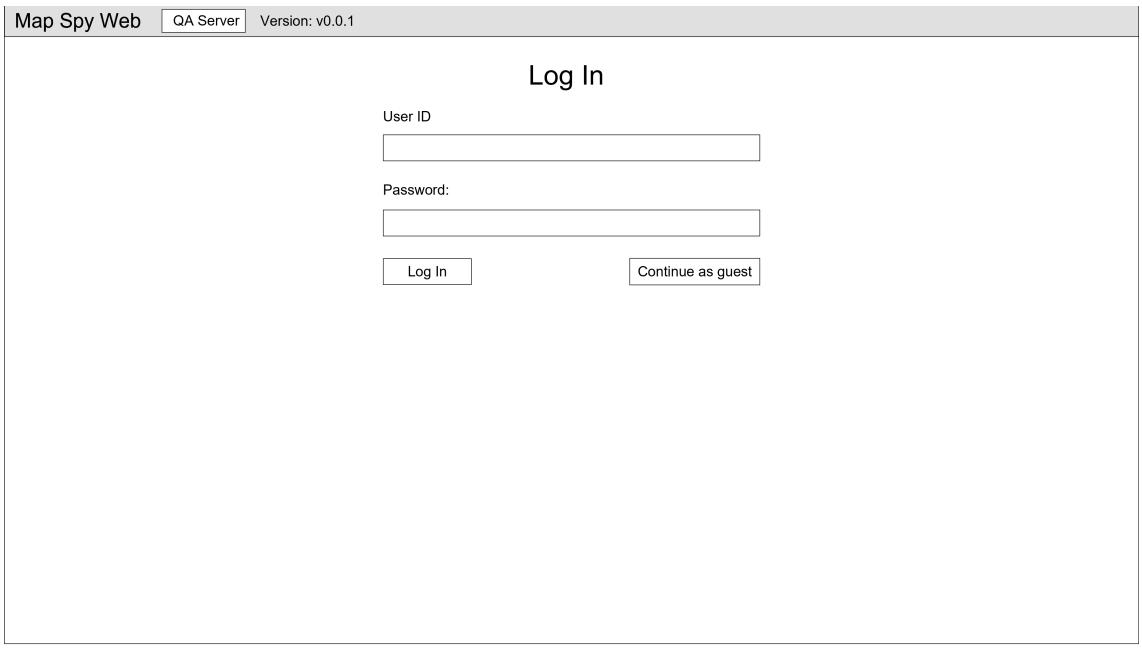
HTML	Značkovací jazyk používaný na webu (angl. *Hypertext Markup Language*)
CSS	Kaskádové styly sloužící k definovaní stylu webových stránek (angl. *Cascading Style Sheets*)
DOM	Reprezentace HTML dokumentu založená na objektech (angl. *Document Object Model*)
UI	Uživatelské rozhraní (angl. *User Interface*)
JavaScript	Vysokoúrovňový netypový programovací jazyk, který využívá většina webů
TypeScript	Typová nadmnožina programovacího jazyka JavaScript
Java	Vysokoúrovňový, objektově orientovaný, striktně typový programovací jazyk
JVM	Abstraktní výpočetní jednotka sloužící ke spuštění kódu napsaného v jazyce Java (angl. *Java Virtual Machine*)
JDK	Kit obsahující nástroje potřebné pro vývoj v jazyce Java (angl. *Java Development Kit*)
JRE	Softwarová vrstva potřebná ke spuštění Java programu (angl. *Java Runtime Environment*)
JAR	Formát souboru sloužící k distribucí programů a knihoven (angl. *Java Archive*)
WAR	Formát souboru sloužící k distribuci programů určených k nasazení na webový server (angl. *Web Application Archive*
GC	Slouží k automatické správě paměti (angl. *Garbage Collection*)
Frontend	Obvykle aplikace běžící přímo u klienta
Backend	Obvykle aplikace běžící na serveru
Dependency Injection	Technika vkládání závislostí v programu (angl. *Dependency Injection*)

XML	Značkovací jazyk (angl. *Extensible Markup Language*)
JSON	Způsob zápisu dat, objektů a polí (angl. *JavaScript Object Notation*)
API	Aplikační programové rozhraní (angl. *Application Programming Interface*)
REST	Architektura rozhraní (angl. *Representational State Transfer*)
SOAP	Protokol pro výměnu objektů (angl. *Simple Object Access Protocol*)
HTTPS	Protokol pro zabezpečenou komunikaci na webu (angl. *Hypertext Transfer Protocol Secure*)
JWT	Průmyslový standard RFC 7519 pro bezpečnou reprezentaci nároků mezi dvěma stranami (angl. *JSON Web Token*)
SVG	Škálovatelná vektorová grafika (angl. *Scalable Vector Graphics*)
Canvas	HTML5 technologie pro zobrazování dvourozměrné bitmapové rastrové grafiky na webu
WebGL	Technologie pro vykreslování 3D grafiky na webu s využitím grafického procesoru
Cloud	Rozsáhlá síť serverů, ke kterým je přistupováno z internetu
PROD	Produkční účely (angl. *Production*)
QA	Zajištění jakosti (angl. *Quality Assurance*)
DEV	Vývojové účely (angl. *Development*)
UMR	Tým ve společnosti onsemi vybývající se vývojem a správou systémů pro mapové repozitaře (angl. *Universal Map Repository*)
Lot	Sada nebo dávka waferů ve výrobě, výroba polovodičů je totiž výrobou dávkovou

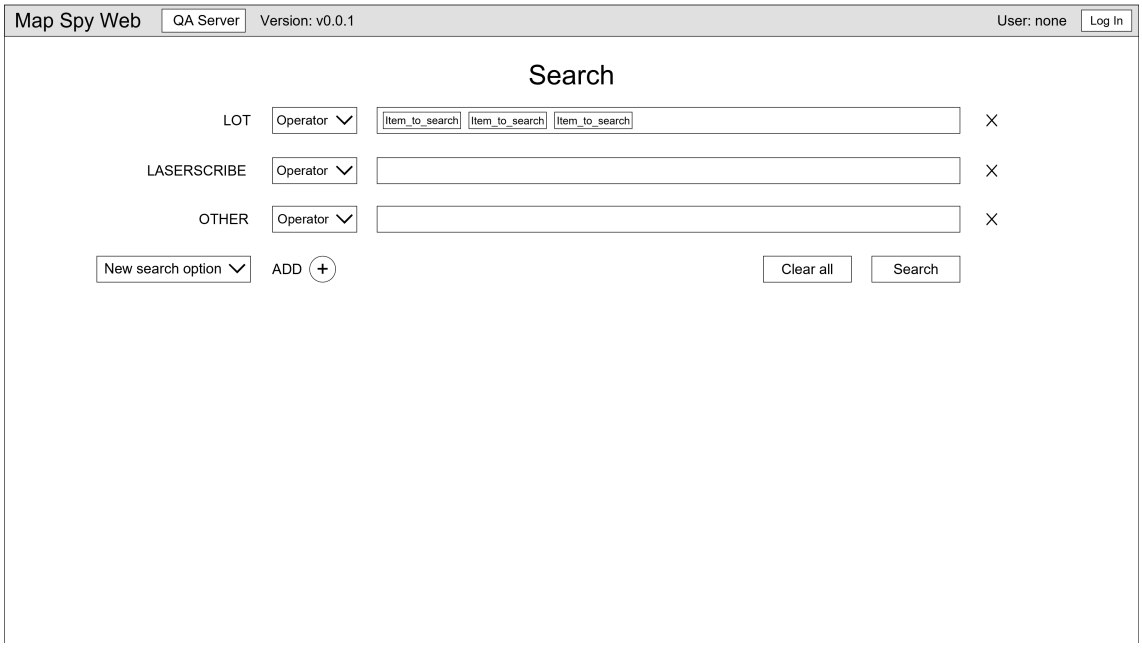
Wafer mapa	Mapa křemíkové desky obsahující data z výroby polovodičů
Flat	Zkosená část křemíkové desky sloužící k identifikaci její orientace
CPU	Centrální procesorová jednotka (angl. *Central Processing Unit*)
GPU	Grafický procesor (angl. *Graphics Processing Unit*)
PrimeNG	Open source knihovna Angular UI komponent
Spring	Framework pro vývoj Java aplikací
RxJS	Knihovna pro vytváření asynchronních a na událostech založených aplikací

Seznam příloh

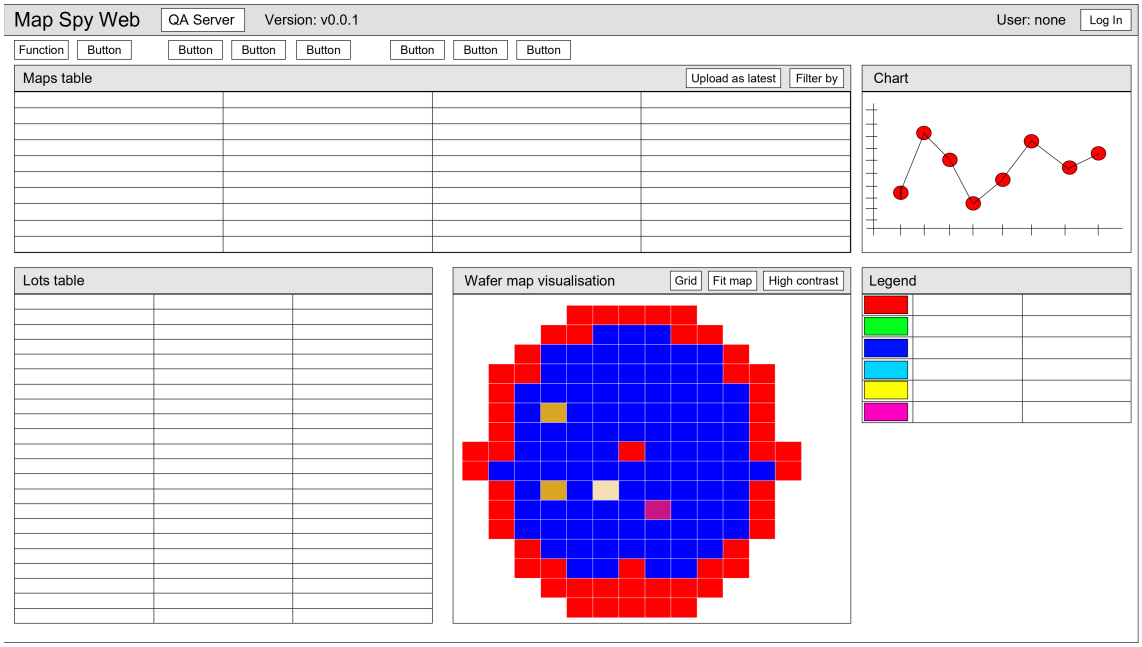
A Návrhy uživatelského rozhraní aplikace



Obr. A.1: Skica s návrhem stránky pro přihlášení uživatele.

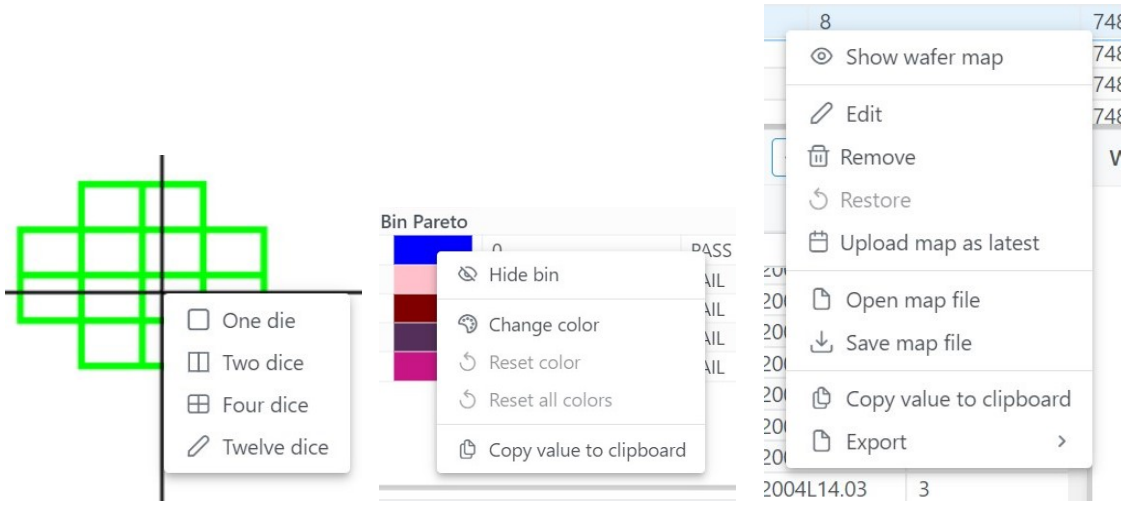


Obr. A.2: Skica s návrhem stránky pro vyhledávání map.



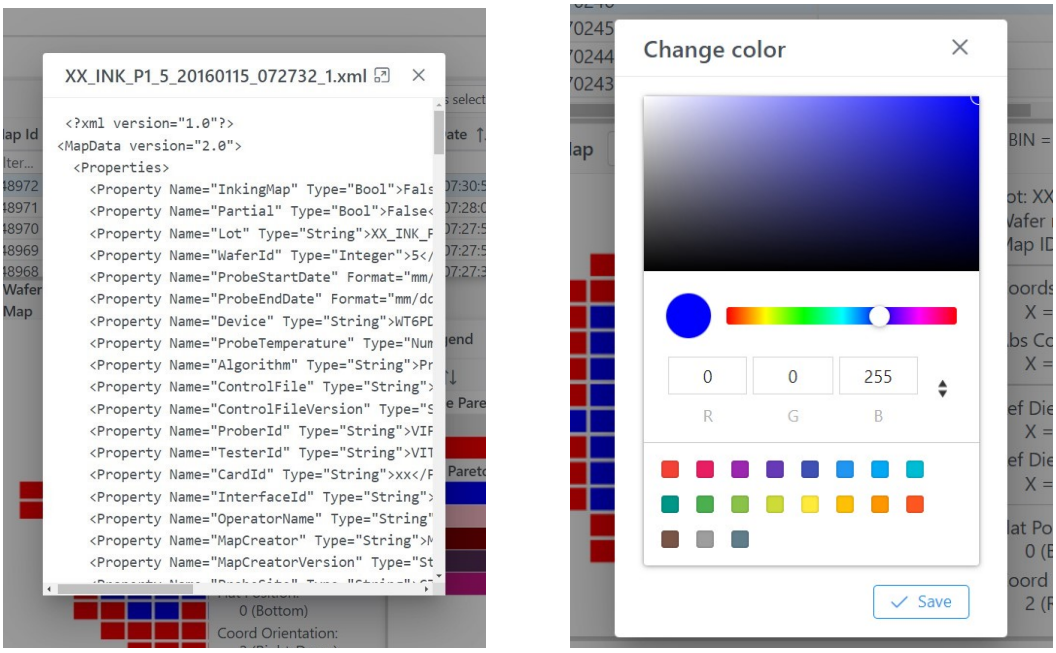
Obr. A.3: Skica s návrhem stránky pro zobrazení výsledků vyhledávání.

B Snímky z výsledné aplikace



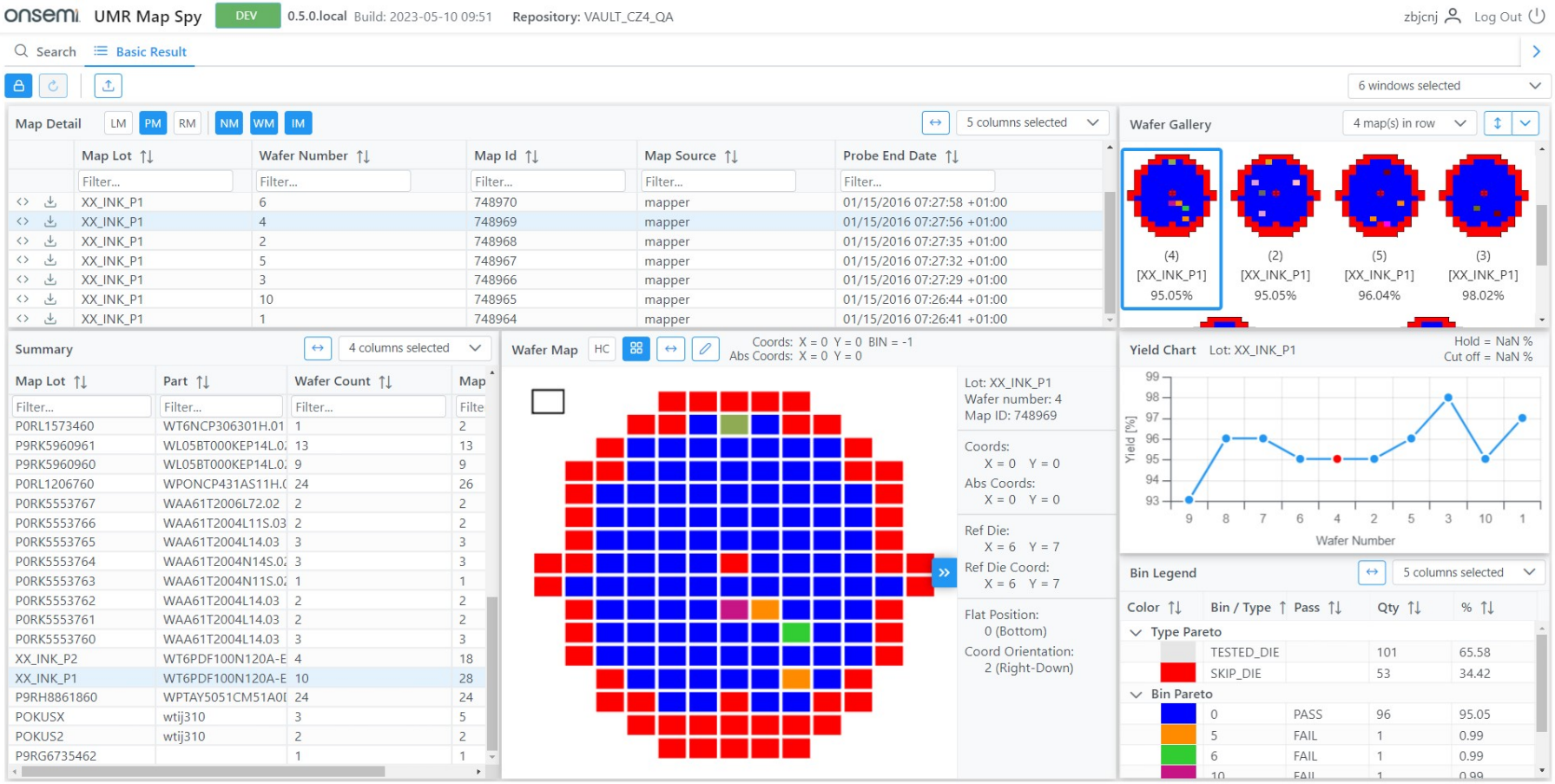
(a) Menu editoru. (b) Menu legendy vizualizace. (c) Menu tabulek metadat.

Obr. B.1: Ukázka kontextových menu aplikace MapSpyWeb.

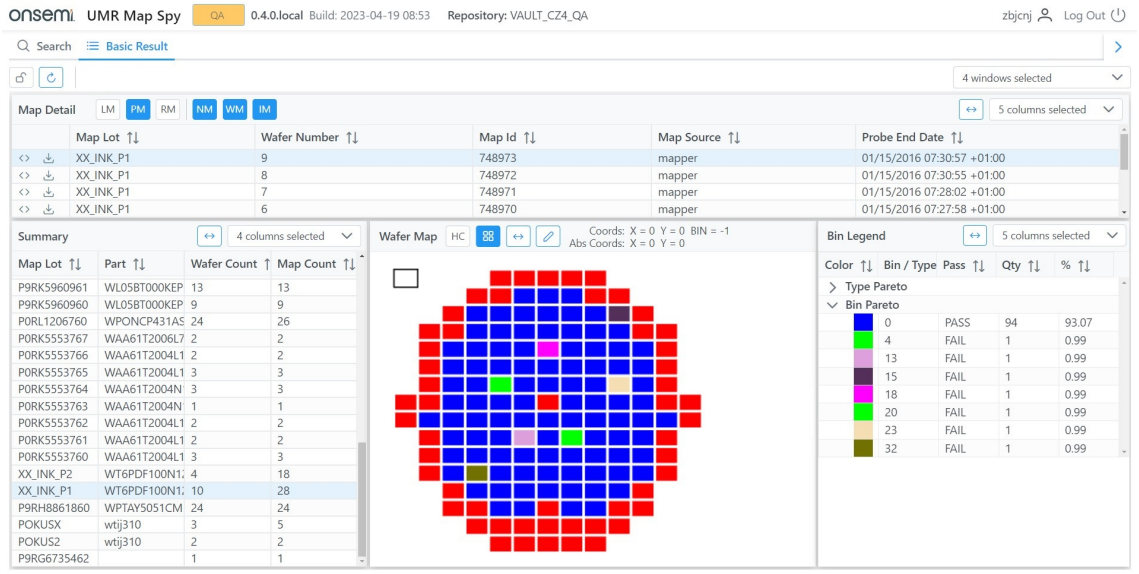


(a) Dialog pro zobrazení mapového souboru. (b) Dialog pro změnu barvy.

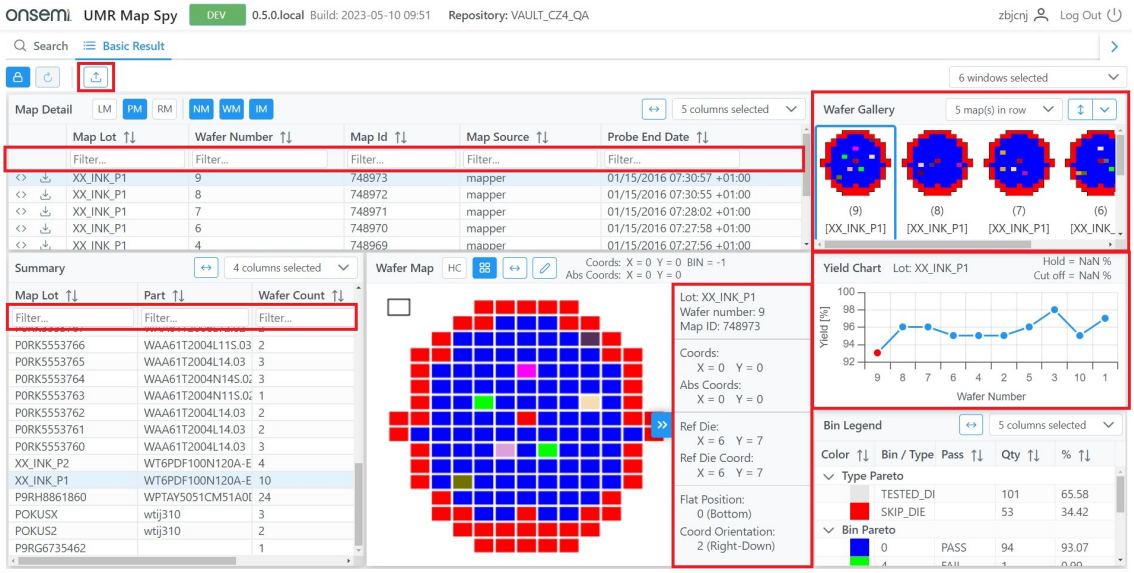
Obr. B.2: Ukázka dialogových oken aplikace MapSpyWeb. Obrázek (a) obsahuje dialog sloužící k zobrazení obsahu mapového souboru bez nutnosti jeho stažení na disk. Na obrázku (b) je snímek dialogu, který umožňuje změnu barvy prvku mapy ve vizualizaci křemíkové desky.



Obr. B.3: Snímek stránky se souhrnem výsledků vyhledávání a vizualizací mapy křemíkové desky (velký).

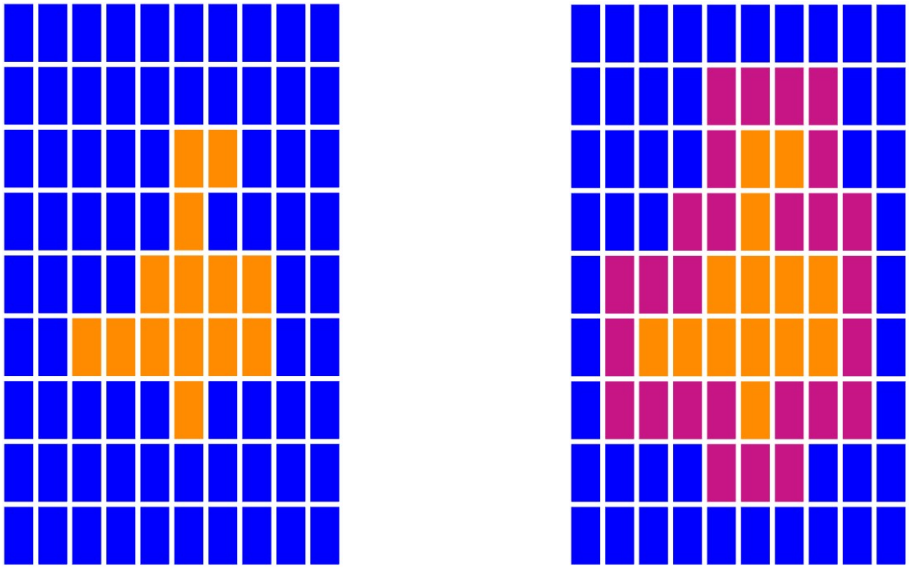


(a) Před zapracováním zpětné vazby.



(b) Po zapracování zpětné vazby.

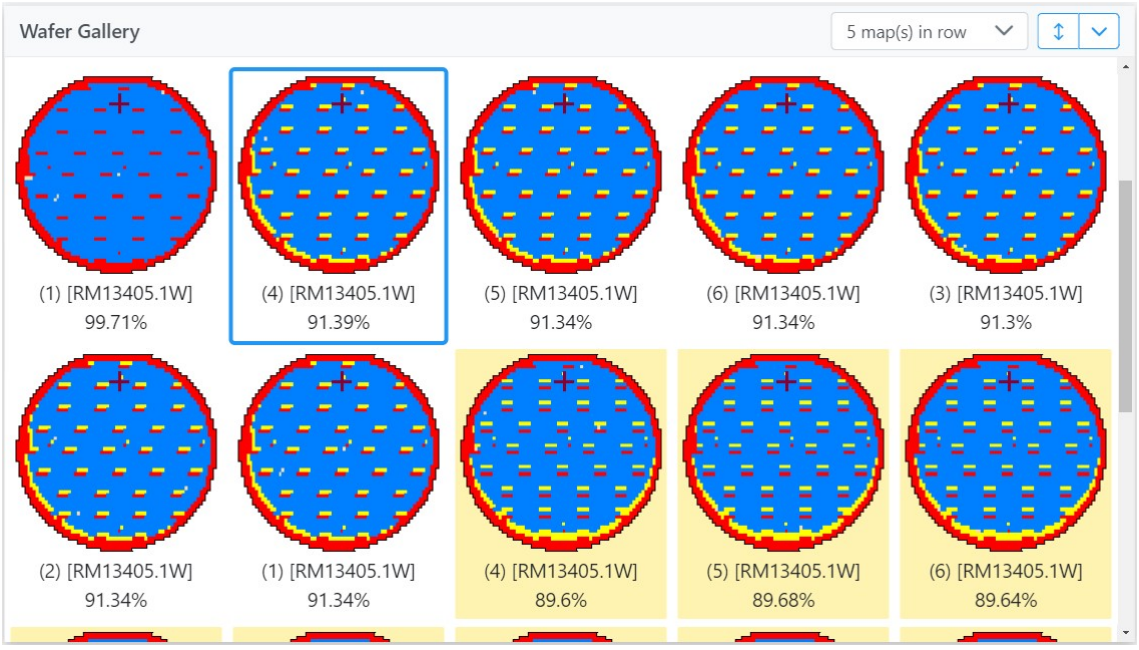
Obr. B.4: Snímky obrazovky zobrazující rozdíl vzhledu stránky s výsledky vyhledávání před a po zapracování zpětné vazby od uživatelů. Snímek (a) zobrazuje vzhled stránky před zapracováním zpětné vazby, snímek (b) vzhled po zapracování zpětné vazby (označeno červenými rámečky).



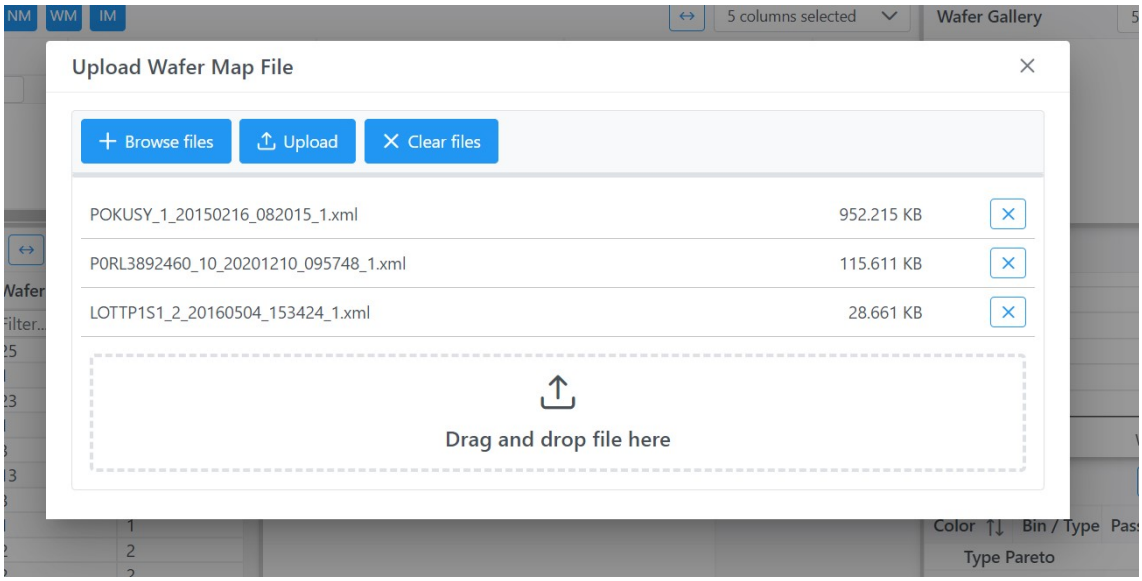
(a) Před obalením oblasti.

(b) Po obalení oblasti.

Obr. B.5: Ukázka použití funkce pro obalení oblasti mapy další vrstvou prvků. Obrázek (a) zobrazuje oblast prvků před aplikací funkce pro obalení prvků. Obrázek (b) zobrazuje stejnou oblast po aplikaci funkce pro obalení.



Obr. B.6: Detailní snímek okna zobrazujícího souhrnnou vizualizaci všech map křemíkových desek pro jednu nebo více dávek.



Obr. B.7: Snímek dialogu, který umožňuje ruční nahrání mapového souboru.

(a) Přední pohled na mapu.

(b) Zadní pohled na mapu.

Obr. B.8: Ukázka indikace horizontálního převrácení mapy pomocí podbarvení tlačítka pro převrácení a změnou barvy ukazatele pozice flat. V obrázku (a) je indikován zelenou barvou tlačítka a ukazatele orientace mapy přední pohled na mapu. V obrázku (b) je indikováno červenou barvou horizontální převrácení mapy na zadní stranu.

C Ukázky zdrojových kódů

Výpis C.1: Metoda pro framework Angular, která umožňuje vytvořit PixiJS vykreslovací plátno.

```
private initPixi(ngZone: NgZone, elementRef: ElementRef): Application {
    let app: Application;
    ngZone.runOutsideAngular(() => {
        app = new Application({
            backgroundColor: 0xffffff,
            resolution: 1,
            antialias: true,
            resizeTo: elementRef.nativeElement.parentElement
        });
    });
    elementRef.nativeElement.appendChild(app.view);
    return app;
}
```

Výpis C.2: Ukázka kódu, který umožnuje vytvořit pixi-viewport kontejner reagující na kliknutí a umožnující interaktivní pohyb pomocí změny polohy myši a přiblížení pomocí kolečka myši.

```
private initPixiViewport(app: Application): Viewport {
    const viewportContainer = app.stage.addChild(new Viewport({
        interaction: app.renderer.plugins.interaction,
        passiveWheel: false,
        stopPropagation: true
    }));

    viewportContainer
        .drag()
        .wheel({smooth: 3, trackpadPinch: true, wheelZoom: true}
        .pinch()
        .on('clicked', event => onMouseClick(event));

    return viewportContainer;
}

private onMouseClick(event): void {
    // kód, který se má vykonat po kliknutí do vizualizace
}
```

Výpis C.3: Ukázka zjednodušené metody sloužící k vytvoření PixiJS grafiky vizualizované mapy a její přidání do pixi-viewport kontejneru pro vykreslení.

```
private drawWaferMap(mapData: number[][], viewport: Viewport,
                     waferMapDieSize: WaferMapDieSize): Graphics {

    const waferMapGraphics: Graphics = new Graphics();
    for (let [rowNumber] of binValues.entries()) {
        for (let [colNumber, binVal] of binValues[rowNumber].entries()) {
            const color = getBinColor(binVal, isPassBin(binVal));

            waferMapGraphics.beginFill(color);
            waferMapGraphics.drawRect(
                colNumber * waferMapDieSize.width,
                rowNumber * waferMapDieSize.height,
                waferMapDieSize.width,
                waferMapDieSize.height
            );
        }
    }
    viewport.addChild(waferMapGraphics);
    return waferMapGraphics;
}
```

Výpis C.4: Ukázka kódu sloužícího k rotaci libovolné mapy ve formě dvourozměrného pole o 90° doprava.

```
private static rotateRightMap<T>(oldMapData: T[][]): T[][] {
    if (oldMapData == null || oldMapData[0] == null) {
        return null;
    }
    const oldRows = oldMapData.length;
    const oldCols = oldMapData[0].length;
    const newMapData: T[][] =[...Array(oldCols)].map(()=> Array(oldRows));
    for(let i = 0; i < oldRows; ++i) {
        for(let j = 0; j < oldCols; ++j) {
            newMapData[j][oldRows - i - 1] = oldMapData[i][j];
        }
    }
    return newMapData;
}
```

D Odvození vzorců využitých pro dopočítaní bodů na přímce

Tato příloha se zabývá odvozením parametrů a, b a c obecné rovnice přímky. Obecná rovnice přímky je definována jako

$$ax + by + c = 0. \tag{D.1}$$

Pro přímku procházející dvěma body v dvourozměrném prostoru o souřadnicích $[x_1, y_1]$ a $[x_2, y_2]$ platí

$$ax_1 + by_1 = -c, \tag{D.2}$$

$$ax_2 + by_2 = -c. \tag{D.3}$$

Po dosazení rovnice D.2 za $-c$ v rovnici D.3 je získán vztah

$$ax_1 + by_1 = ax_2 + by_2. \tag{D.4}$$

Po vytknutí a a b ze vztahu D.4 získáme:

$$a(x_1 - x_2) = b(y_2 - y_1). \tag{D.5}$$

Dále je předpokládáno, že musí platit vztah $ab = ba$. Lze tedy psát, že současně platí

$$a = y_2 - y_1, \tag{D.6}$$

$$b = x_1 - x_2. \tag{D.7}$$

Parametr c je následně určen z rovnice D.2 po dosazení za a a b

$$c = -(ax_1 + by_1). \tag{D.8}$$

Neznámou souřadnici x_n nebo y_n bodu na přímce pak pro odpovídající známou souřadnici y_z nebo x_z dopočítáme pomocí následujících vzorců, které byly vyjádřeny z rovnice D.1

$$x_n = \frac{-c - by_z}{a}, \tag{D.9}$$

$$y_n = \frac{-c - ax_z}{b}. \tag{D.10}$$

E Rozhraní vizualizačního modulu aplikace MapSpyWeb

- **Vstupy Angular komponenty vizualizace**
  - `[auxiliaryGridEnabled]` – Zobrazení nastavitelné mapové mřížky.
  - `[auxiliaryGridProperties]` – Parametry nastavitelné mřížky.
  - `[crossPointerEnabled]` – Zobrazení indikátoru polohy kurzoru.
  - `[drawingBrush]` – Nastavení velikosti štětce.
  - `[drawingFailBin]` – Nastavení typu prvku pro zakreslování.
  - `[markAsFailEnabled]` – Spuštění režimu editace.
  - `[eraserEnabled]` – Spuštění režimu gumy.
  - `[widenFailClusterEnabled]` – Spuštění režimu obalování vadných prvků.
  - `[flipMapHorizontally]` – Horizontální převrácení mapy.
  - `[grid]` – Zobrazení základní bílé mřížky.
  - `[hiddenBins]` – Seznam prvků, které mají byt skryty ve vizualizaci.
  - `[highContrast]` – Režim vysokého kontrastu.
  - `[waferMapContent]` – Vstupní mapová data.
  - `[waferMapViewerMode]` – Nastavení módu vizualizace na zobrazovací nebo editační.
- **Veřejné metody Angular komponenty vizualizace**
  - `void fitMapToViewport()` – Metoda pro vycentrování mapy do zobrazovací oblasti.
  - `boolean isMapChanged()` – Metoda k zjištění, zda byla nad mapou provedena nějaká změna.
  - `void mapEditRedo()` – Metoda k vyvolání akce znovu.
  - `void mapEditUndo()` – Metoda k vyvolání akce zpět.
- **Výstupy Angular komponenty vizualizace**
  - `(contextMenuOpenEvent)` – Událost indikující pokus o otevření kontextového menu nad vizualizací.
  - `(editorWaferMapContentChangedEvent)` – Událost indikující provedenou editaci mapy uživatelem.
  - `(errorOccurredEvent)` – Událost indikující vznik chyby v komponentě.
  - `(selectedBinAtMapPositionEvent)` – Událost, která indikuje kliknutí na prvek mapy.
  - `(undoAvailableEvent)` – Událost informující o dostupnosti možnosti vrátit akci zpět.
  - `(redoAvailableEvent)` – Událost informující o dostupnosti možnosti provést akci znovu.

F Obsah elektronické přílohy

- `/` — Kořenový adresář paměťového média
  - `readme.pdf` — Popis elektronické přílohy
  - `2023_DP_Turon_Rudolf_211188.pdf` — Text diplomové práce
  - `obrazky_z_DP` — Obrázky z diplomové práce
    - `1-1.jpg`
    - `2-1.pdf`
    - `2-2.png`
    - `...`
  - `ukazkova_videa` — Ukázková videa z používání aplikace
    - `pouzivani.mp4`
  - `dokumentace` — Dokumentace výsledné aplikace
    - `dokumentace_backend` — Dokumentace backend části
      - `index.html` — Otevřít dokumentaci ve webovém prohlížeči
      - `...`
    - `dokumentace_rest_api` — Dokumentace REST API backend části
      - `index.html` — Otevřít dokumentaci ve webovém prohlížeči
      - `...`
    - `dokumentace_frontend` — Dokumentace frontend části
      - `index.html` — Otevřít dokumentaci ve webovém prohlížeči
      - `...`
  - `program` — Zdrojový kód výsledné aplikace MapSpyWeb
    - `mapspy-web` — Adresář obsahující projekt aplikace MapSpyWeb
      - `src` — Kód backend části aplikace
      - `ui` — Kód frontend části aplikace
      - `pom.xml` — Definice sestavení aplikace pro Apache Maven
      - `...`
  - `demo_program` — Demo program pro demonstraci aplikace
    - `mock_rest_api` — Mock REST API backend části aplikace
      - `mapspyweb.json` — Soubor s definicí Mock REST API pro Mockoon
      - `...`
    - `testovaci_frontend` — Soubory sestavené frontend části aplikace
  - `testovaci_programy_vizualizace` — Lze spustit ve webovém prohlížeči
    - `testovaci_program_canvas.html`
    - `testovaci_program_konva.html`
    - `testovaci_program_pixijs.html`

F.1 Návod na spuštění demo programu z elektronické přílohy

Návod se vztahuje k operačnímu systému Microsoft Windows 10, či 11.

1. Zkopírovat adresář `./demo_program` z elektronické přílohy do zařízení, kde bude demo program spuštěn
2. Spuštění mock REST API
   (a) Stáhnout a nainstalovat Mockoon v1.23.0 z `https://github.com/mockoon/mockoon/releases/download/v1.23.0/mockoon.setup.1.23.0.exe`.
   (b) Spustit aplikaci Mockoon
   (c) V horní liště aplikace Mockoon zvolit *File* -> *Open Environment*
   (d) Vybrat soubor `./demo_program/mock_rest_api/mapspyweb.json`
   (e) Spustit mock REST API pomocí tlačítka se zelenou šipkou („Start server“) v horní části aplikace Mockoon
3. Spuštění frontend části aplikace MapSpyWeb
   (a) Stáhnout a nainstalovat Node.Js v16.14.2 z `https://nodejs.org/download/release/v16.14.2/node-v16.14.2-x64.msi`
   (b) Otevřít Windows příkazový řádek v adresáři `./demo_program/testovaci_frontend`
   (c) Provést příkaz pro instalaci Node.js web serveru (pozor na mezery při kopírování z pdf)

   ```
   npm install http-server -g
   ```

   (d) Provést příkaz pro spuštění web serveru s aplikací MapSpyWeb (pozor na mezery při kopírování z pdf)

   ```
   http-server . -p 8082 -o /mapspy-web/
   ```

4. Aplikace MapSpyWeb je spuštěna (pro přihlášení do aplikace je možné zadat jakékoli User ID a heslo, například User ID: test1, heslo: test1)