Rozšíření aplikace pro monitorování procesu výroby firmy ON Semiconductor Czech Republic, s.r.o.

Marek Kantor

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Marek Kantor**
Osobní číslo:	**A19810**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Rozšíření aplikace pro monitorování procesu výroby firmy ON Semiconductor Czech Republic, s.r.o.**
Téma práce anglicky:	**Extension of ON Semiconductor Czech Republic, s.r.o. production process monitoring application**

Zásady pro vypracování

1. Popište současný stav technologií pro vývoj dané aplikace.
2. Zaměřte se na technologie .NET, ASP.NET a na integraci s Open Office SDK.
3. Navrhněte rozšíření aplikace o kontrolu velikosti měřených veličin, definuje funkční a nefunkční požadavky, případy užití a databázový model a vytvořte drátové modely uživatelského rozhraní.
4. Vytvořte prototyp rozšíření aplikace a popište jeho klíčové části.
5. Zhodnoťte dosažené výsledky a formulujte možnosti dalšího vývoje aplikace.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PRINCE, Mark. C# 8.0 and .NET Core 3.0: Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code. 4. Birmingham: Pack Publishing, 2019. ISBN 978-1788478120.
2. .NET documentation. Microsoft Docs [online]. Oficiálna dokumentácia firmy Microsoft Corporation. Dostupné z: https://docs.microsoft.com/en-us/dotnet/
3. ASP.NET | Open-source web framework for .NET. .NET | Free. Cross-platform. Open Source. [online]. Dostupné z: https://dotnet.microsoft.com/apps/aspnet
4. NAGEL, Christian. Professional C# 7 and .NET Core 2.0. 11th edition. Indianapolis: Wrox, a Wiley Brand, 2018. ISBN 978-1119449270.
5. ALBAHARI, Joseph a Ben ALBAHARI. C# 7.0 in a nutshell. 7th edition. Sebastopol: O´Reilly, 2018. ISBN 978--1491987650.

Vedoucí bakalářské práce: **Ing. Erik Král, Ph.D.**
Ústav počítačových a komunikačních systémů

Oponent bakalářské práce: **Ing. Bc. Pavel Vařacha, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **22. července 2022**
Termín odevzdání bakalářské práce: **19. srpna 2022**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 25. července 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis studenta

ABSTRAKT

Tato bakalářská práce se zabývá rozšířením aplikace pro monitorování procesu výroby. Teoretická část se zaměřuje na technologie .NET, ASP.NET, jak fungují a způsob, kterým se vyvíjely. V praktické části se autor bude věnovat návrhu a realizaci dané aplikace. Bude zpracována studie proveditelnosti a popsáno vytvořené řešení. Cílem práce bylo vytvořit funkční prototyp této aplikace.

Klíčová slova: .NET, ASP.NET

ABSTRACT

This bachelor thesis deals with the extension of a production process monitoring application. The theoretical part focuses on .NET, ASP.NET technologies, how they work, and how they have evolved. In the practical part, the author will focus on the design and implementation of the application. A feasibility study and description of the developed solution will be done. The aim of the work was to create a working prototype of this application.

Keywords: .NET, ASP.NET

Tímto bych chtěl velmi poděkovat mým konzultantům Mgr. Miroslavovi Holáňovi, Ing. Michalovi Musilovi a vedoucímu Ing. et Ing. Eriku Královi, Ph.D. za cenné rady a podmětné připomínky. Především děkuji za motivaci a trpělivost při psaní této bakalářské práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

S příchodem internetu, přišly i veliké změny, ať už se jedná o způsob přenosu informací, komunikace, nebo třeba bankovnictví, jednou ze změn jsou také určitě i webové aplikace, které v dnešní době pomalu zastiňují aplikace desktopové, proč je tomu tak? Hlavní výhoda těchto aplikací je zejména dostupnost, přístup k internetu je a funkční zařízení s internetem je to jediné co je potřeba.

Jelikož doba jde dopředu, a tak i firma onsemi se snaží své aplikace posouvat a webové aplikace již nějakým rokem používá. Ať už se jedná o aplikace pro výrobu nebo stránky pro zaměstnance. Jednou takovou aplikací byla Logbook. Tato aplikace má za úkol abstrahovat data z velké procesní databáze s názvem Promis a vyobrazovat je v uživatelsky přívětivé podobě. Aplikace Logbook v současném stavu neobsahuje záznamy o testovacích křemíkových deskách a jejich sledovaných hodnotách. Proto firma onsemi požádala autora o zpracování jejich evidence jako téma bakalářské práce.

Práce se věnuje úpravě již zmíněné aplikace o možnost přidání ručního záznamu, editaci ručního záznamu a auditaci změn. V části teoretické bude věnována pozornost technologiím použitých v aplikaci. Jedná se hlavně o frameworky .NET, ASP.NET, vysvětleno bude ale i mnoho dalších pojmů, které s práci bezprostředně souvisí.

V praktické části práce bude navrženo rozšíření aplikace o kontrolu velikosti měřených veličin, budou definovány funkční a nefunkční požadavky, případy užití a drátové modely uživatelského rozhraní. Na základě návrhu bude vytvořen prototyp rozšíření aplikace včetně databázového modelu a budou popsány jeho klíčové části. V závěru práce budou zhodnoceny dosažené výsledky a formulovány možnosti dalšího vývoje aplikace.

I. TEORETICKÁ ČÁST

1 SOUČASNÝ STAV TECHNOLOGIÍ PRO VÝVOJ WEBOVÝCH APLIKACÍ

Vývoj webových aplikací jde stále dopředu a lidé chtějí stále lepší a rychlejší aplikace. Pro různé druhy webových stránek se preferují různé webové frameworky. Podle Stack Overflow Developer Survey 2021 [1] se na prvním místě jako nejpoužívanější framework umístil React.js. V průzkumu hlasovali jak profesionální programátoři, tak lidé, kteří profesionálové nejsou, a tyto hlasy jsou rozdělené na dvě ankety. Práce se bude zabývat pouze anketou vytvořenou profesionály.

Jedná se o Javascriptový framework vyvíjený firmou Facebook a komunitou samotných vývojářů. React se soustředí na práci s rychle se měnícími daty a stará vyobrazovaní dat uživateli.

Na druhém místě se umístil framework jQuery, jedná se opět o Javascriptovou knihovnu. Tato knihovna je malá ovšem velice bohatá knihovna na funkce. Díky snadno použitelnému rozhraní API, které funguje ve velkém množství prohlížečů, jsou věci, jako je procházení a manipulace s dokumenty HTML, zpracování událostí, animace a používání technologie AJAX, mnohem jednodušší. Díky kombinaci všestrannosti a rozšiřitelnosti změnil jQuery způsob, jakým miliony lidí píšou JavaScript. [2]

Třetí místo si vybojoval Google se svým produktem jménem Angular. Jedná se o bezplatný multiplatformní framework opět založený na TypeScriptu.

Na čtvrtém místě se umístil první backendový framework jménem Express. Jedná se nástavbu pro softwarový systém Node.js.

ASP.NET Core, je framework, jemuž je v práci věnována velká pozornost. Tento framework se umístil těsně na pátém místě před Vue.js. Na místě sedmém se umístil starší verze tohoto frameworku a to ASP.NET, kdybychom je brali jako jeden framework, protože jsou si hodně podobné, tak by se umístili na druhém místě.

Na zbylých místech se umístili jak frameworky Javascriptu, tak frameworky serverového jazyka PHP, jedná se konkrétně o Spring, Flask, Django, Angular.js, Laravel, Ruby on Rails, Gatsby, Symfony, FastAPI, Svelte a Drupal. [1]

2 TECHNOLOGIE .NET

Následující kapitola se zaměřuje na historii vývoje a funkce technologií .NET.

2.1.1 Microsoft .Net framework

Koncem 20. století firma Microsoft začala pracovat na projektu nazvaný NGWS. Cílem bylo sjednotit všechny produkty Microsoftu a přidat koncovku .NET. To však nešlo podle plánu a Microsoft se rozhodl z NGWS vytvořit .NET jak ho dnes známe. Projekt přejmenovali a za pár let vyšel první .NET framework 1.0.

První beta verze .NET frameworku byla vydána na konci roku 2000 a 13. února 2002 byla vydána první verze .NET 1.0. Jeho hlavní funkcí bylo CLR, objektově orientovaný vývoj webových aplikací ASP.NET a desktopových aplikací WinForms.

Verze frameworku .NET 2.0 v roce 2005 přinesla spoustu nových funkcí pro ASP.NET, ale mimo jiné i generické kolekce, iterátory a nullable typy.

O rok později však vyšla nová verze a to .NET 3.0, která obsahovala WPF, WCF a WWF. WPF je architektura uživatelského rozhraní na vývoj aplikací pro stolní počítače. S pomocí WCF může odesílat data asynchronně, a to na koncové body hostované službou ISS nebo službou v aplikaci.

Další verze .NET byla 3.5 v roce 2007, která obsahovala funkce na podporu jazyku AJAX, LINQ nebo také ASP.NET MVC viz 2.2. Microsoft se také rozhodl že zdrojové kódy knihoven budou přístupné pod licencí Microsoft Reference Software License.

Do roku 2014 Microsoft přišel ještě s pár dalšími verzemi začínající číslem 4, které obsahovaly třeba MEF, což je framework pro rozšiřovaní projektu bez nutnosti konfigurace, vývojáři se díky němu lehce vyhnou pevným závislostem. Také zlepšili výkon, ladění a vytvořili Razor view engine. [3][4]

2.1.2 Microsoft .NET Core

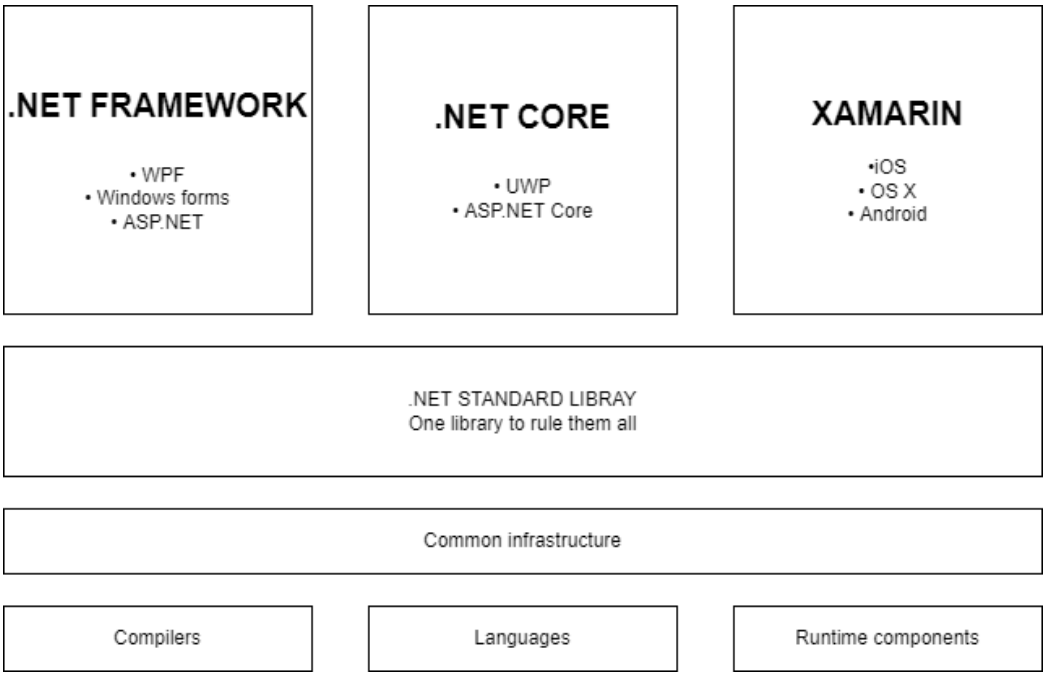
V roce 2014 Microsoft oznámil vývoj systému Microsoft .NET Core. Microsoft předělal vrstvu pod programovacími jazyky a to CLR, to se přejmenovalo na CCLR a přestali podporovat staré funkce, které už nebyly potřebné. Celý systém vyšel v roce 2016 a představoval vyšší výkon a stal se také multiplatformní. .NET Core zůstává zhruba z 97% kompatibilní s .Net frameworkem. Díky novému systému můžeme vyvíjet nově i mobilní

aplikace nebo aplikace pro Linux a jiné platformy. Některé platformy však mají stále své omezení, to že se stal .NET multiplatformní ještě neznamená, že třeba na Linuxu poběží Windows Forms, ty jsou totiž závislé na operačním systému Windows. [5][3]

Po verzi .NET Core 3, byla verze .NET Core 4 přeskočena a Microsoft se rozhodl že změní název celého systému pouze na .NET, nejedná se tak o žádnou velkou změnu a vše funguje stejně jako to bylo předtím.

2.1.3 Microsoft .NET Standard

Microsoft .NET Standard představuje interface pro knihovny základních tříd jednotlivých platforem .Net. Znamená to že .NetFramework, .Net Core, Xamarin, Silverlight, Unity atd. mohou sdílet jednotlivé knihovny. [16]



Obrázek 1 - .NET Standart (vypracováno autorem dle [16])

2.1.4 Jazyky platformy .NET

Aplikace .NET mohou být psány v C#, F# nebo Visual Basic. Každý jazyk, který podporuje .NET, musí dodržovat společný standard, musí vydávat a připojovat metadata ke každému binárnímu nebo přenosnému spustitelnému souboru .pe. Metadata zahrnují typy, objekty, členy a odkazy.

Standart se jmenuje CLI. Specifikuje a definuje prostředí, jenž umožňuje používání více vysokoúrovňových programovacích jazyků. Aniž by bylo nutné přepisovat jejich

překladače. Obsahuje také sadu CTS, jedná se o sadu, která definuje jaké typy jsou deklarovány, používány a spravovány v CLR. Systém poskytuje objektově orientovaný model, ten .NET používá v několika jazycích. Další sada co patří do podmnožiny sady CTS se jmenuje CLS. V této sadě se nacházejí přísnější pravidla než v samotném CTS. Například vícenásobná dědičnost nebo a zrušení pointerů. [6]

Samotné jazyky se generují zmíněné metada, se kterýma pracuje samotné CLR.

Jazyk C#

Je jednoduchý, moderní, objektově orientovaný a bezpečný programovací jazyk.

Následující kód představuje ukázku jazyka C#:

```
var names = new[]
{
   "Anicka",
   "Filip",
   "Ema"
};

foreach (var name in names)
{
   Console.WriteLine($"Hello {name}");
}
```

Jazyk F#

Jedná programovací jazyk zaměřený především funkcionální programování, který usnadňuje psaní stručného, robustního a výkonného kódu. Následující ukázka představuje stejný program jako v předcházejícím odstavci, ale tentokrát zapsaný v jazyce F#:

```
let names = [ " Anicka"; " Filip"; " Ema" ]

for name in names do printfn
$"Hello {name}"
```

Visual Basic

Je jazyk s jednoduchou syntaxí pro vytváření bezpečných, objektově orientovaných aplikací. Následující ukázka představuje stejný program jako v předcházejícím odstavci, ale tentokrát zapsaný v jazyce Visual Basic:

```
Dim names As New List(Of String)({
   "Anicka",
   " Filip",
   " Ema"
})


For Each name In names
   Console.WriteLine($"Hello {name}")

Next
```

[3]

2.1.5 NuGet

V každém programovacím jazyce potřebujeme nástroj, pomocí kterého by mohli vývojáři vytvářet a sdílet svůj kód s okolím. V různých jazycích se setkáváme s knihovnami, ať se jedná o jazyk C nebo třeba Javu. V platformě .NET se nacházejí tzv. balíky NuGet.

Jedná se o soubor s příponou .nupkg, který obsahuje zkompilovaný kód dll, další soubory související s tímto kódem a soubor s informacemi o daném nugetu. NuGety mohou být veřejně přístupné a můžeme si je stáhnout přímo ve vývojovém prostředí, nebo také privátní. V projektu jsou používány jak Nugety veřejné, tak Nugety privátní, firemní, které definují a standardizují třeba přihlašovací systém.[3]

2.1.6 Common Language Runtime

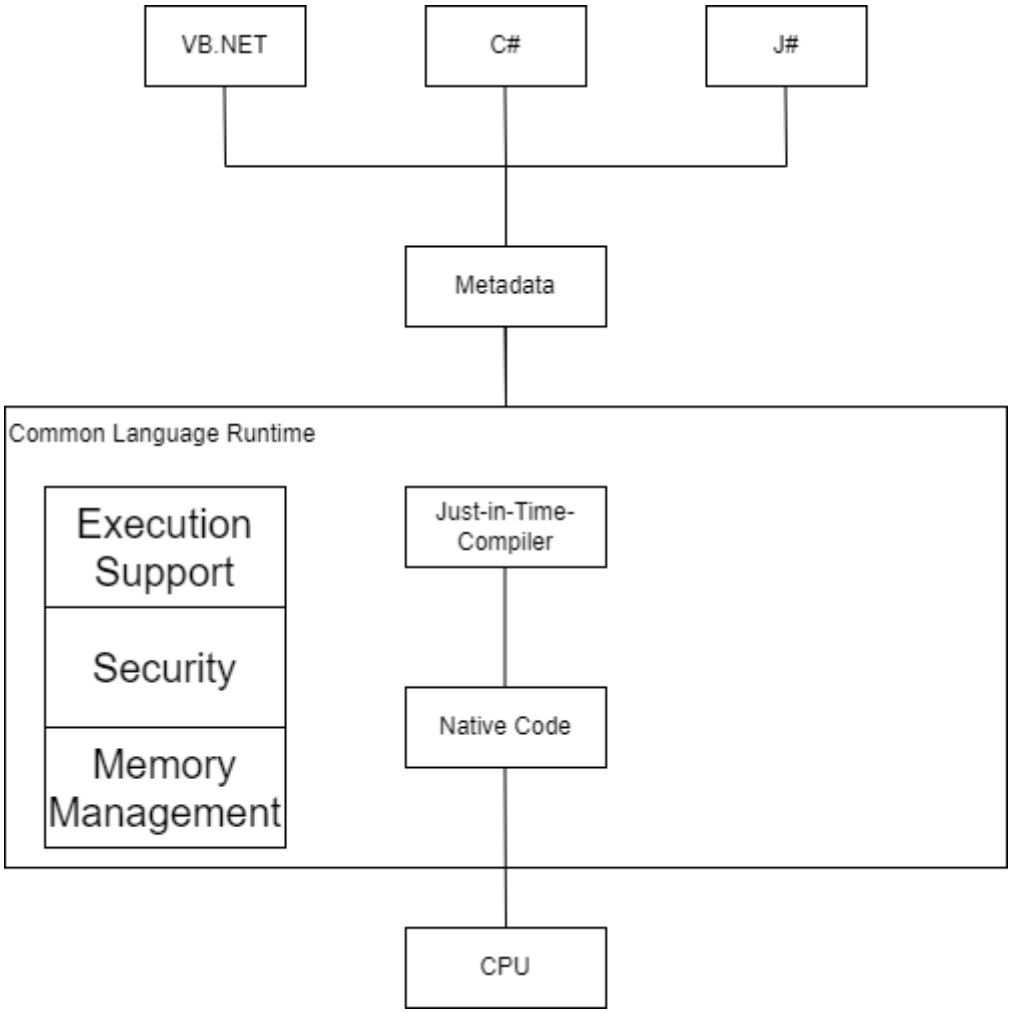
Common Language Runtime je nedílnou součástí frameworku .NET a jedná se o systém, jež je společný pro všechny jazyky tohoto frameworku. Každý jazyk vygeneruje metada data, které mají stejný formát, tyto medata poté zpracovává tento modul. Používá je k vyhledání a načítání tříd, rozmístění instancí v paměti a vyvolávání metod.

Díky tomuto modulu je možné také třeba definovat třídu a potom použít jiný jazyk abychom metodu zavolali. Můžeme předat instanci třídy metodě třídy napsané v jiném jazyce.

Zdroj [3] uvádí: "Tato integrace mezi jazyky je možná, protože kompilátory a nástroje jazyka, které cílí na modul runtime, používají společný systém typů definovaný modulem runtime a dodržují pravidla modulu runtime pro definování nových typů a také pro vytváření, používání, zachování a vazbu na typy."

V modulu existuje také Garbage collector. Spravuje uvolňování a přidělování paměti pro aplikaci. Vývojářům to usnadní mnoho práce a nemusí psát a dlouhé programy pro správu paměti. Garbage collector dokáže i sám od sebe eliminovat nějaké běžné chyby programátora, a to například zapomenout uvolnit objekt, nevracení paměti nebo pokus o přístup k paměti pro objekt, který už byl uvolněn.

Modul pracuje se zabezpečením, vlákny, výjimkami, knihovnami nebo třeba debugem. Obsahuje také compiler JIT jenž se stará o převod daných metadat do strojového kódu. Strojový kód je posloupnost strojových instrukcí pro procesor počítače. [3][6]



Obrázek 2 - Common Language Rutime (Vytvořeno autorem dle [17])

2.2 ASP.NET

ASP.NET je bezplatná webová technologie pro vytváření webů a webových aplikací pomocí HTML, CSS a JavaScriptu. Umožnuje také vytvářet webová rozhraní API a používat technologie v reálném čase, jako jsou webové sokety. Technologie je multiplatformní. Poprvé byla vydána v roce 2002 v .NET frameworku. ASP.NET se stal nástupcem technologie ASP, která byla taky vyvíjená společností Microsoft nová verze byla daleko rychlejší a ve všech ohledech lepší. Díky velikému množství knihoven a rozsáhlému výběru ovládacích prvků se vývoj aplikací zrychlil. Schopnost využití paměti Cache uvolňuje server od většího zatížení, samotná technologie ASP, tyto funkce neměla.

Existují tři přístupy k vytváření moderních webových aplikací či stránek.

- Aplikace, které pomocí serveru vykreslují uživatelské rozhraní
- Aplikace, které vykreslují uživatelské rozhraní u klienta v prohlížeči
- Hybridní aplikace, které využívají jak rozhraní serveru, tak klienta. Například většina webového a uživatelského rozhraní se vykreslí na serveru a klientské komponenty jsou přidány podle potřeby. [3]

2.2.1 ASP.NET WebForms

Webforms tzv. webové formuláře vyšly zároveň s ASP.NET v prvním .NET frameworku, jedná se architekturu, kdy kód je spuštěný na serveru a generuje výstup do webové stránky. Jedná se o kombinaci serverových ovládacích prvků, kódu serveru, HTML a klientských scriptů. Tato technologie je ovšem zastaralá a už se moc nepoužívá daleko více se používá architektura MVC, která je v práci popsána o kapitolu níže.

2.2.2 ASP.NET Core MVC

Mnoho dnešních aplikací je napsáno právě pomocí technologie MVC, dříve se hodně používala technologie Silverlight. Microsoft ji ale čím dál tím míň podporoval až ke konci minulého roku její podporou úplně ukončil. Aplikace vytvořené se Silverlightem se však nadále dají používat v internetu exploleru 11 a v Microsoftu Edge za použití integrovaného módu EI. Což ale není uživatelsky přívětivé. Pokud někdo chce i nadále Silverlight využívat existuje projekt OpenSilver, jedná se open-source projekt, který funguje podobně jako Silverlight jen používá moderní nástroje.

Architektura MVC existuje už hodně let, a využívá jí hodně programovacích jazyků. Účelem celé architektury je oddělit logiku od výstupu. Řeší nám problém tzv. „špagetového kódu“ kdy v jedné třídě se nachází data, logické operace i renderování výstupu. Třída teda obsahuje kód, html tagy i databázové dotazy. Microsoft se snaží, aby se vše dalo napsat v jazyce C# a výstup by se renderoval v html. Není to ale tak jednoduché a občas se vše v C# napsat nedá.

Architektura se dělí do třech hlavních komponent Model, view a controller dále budou postupně po jednom rozebrány jednotlivé komponenty. [3]

Model

Model se stará o veškerou logiku programu. Například výpočty, validace, databázové dotazy atd. Model může být i datová struktura bez logiky. [7]

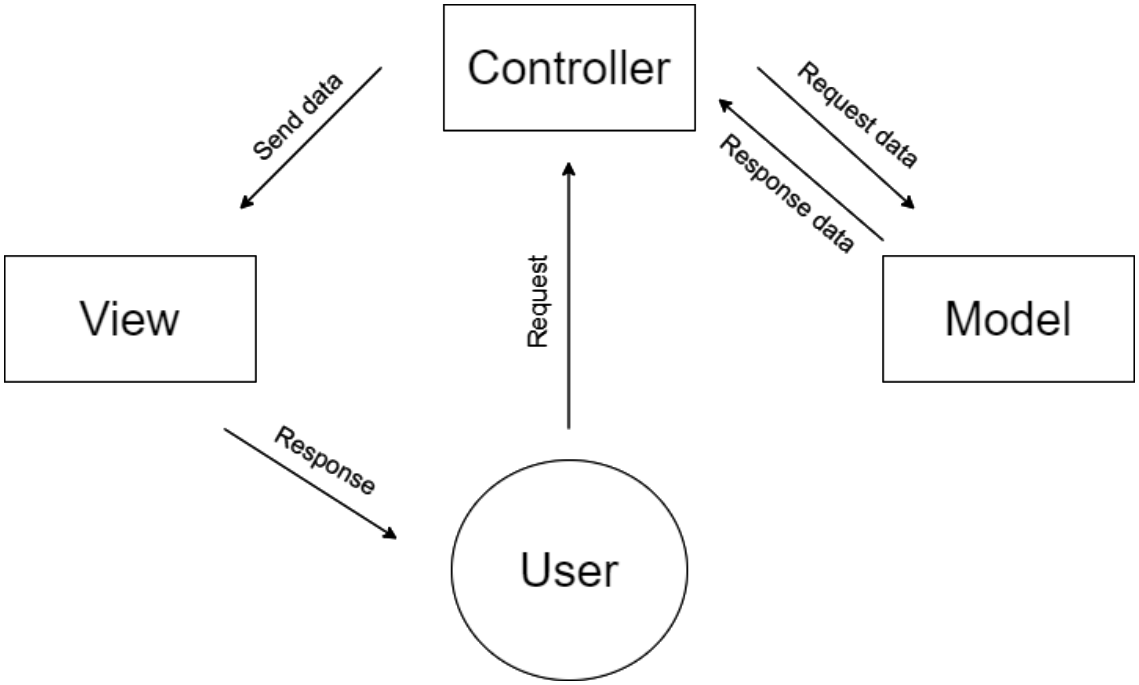
View

View v češtině pohled, se stará vyobrazení dat uživateli jedná se o takový mix html jazyka a C# jménem Razor, má příponu .cshtml. Umožnuje do html kódu přidávat vypisovat data z modelu a používat je v cyklech a v podmínkách. Umožňuje také volat kontrolery a jejich funkce. Pokud bychom chtěli vložit jeden view do druhého i toto nám architektura MVC umožnuje. Dokonce existuje layout.cshtml, který tvoří šablonu celé stránky.

Razor poskytuje syntaxi pro vytváření dynamických webových stránek pomocí HTML a C#. Kód v C# je vyhodnocen na serveru a výsledný obsah HTML je odeslán uživateli.

Controller

Poslední prvek architektury je Controller, jedná se o takového prostředníka mezi modelem a pohledem. Samotný kontrolér si vyžádá data od model a posílá je do pohledu, který je vyobrazuje uživateli. Jak můžeme vidět na obrázku níže. [8]



Obrázek 3 - MVC (Vytvořeno autorem dle [18])

2.2.3 Blazor

Pro serverovou část se často používají programovací jazyky jako je C#, Java, Python a další. Pro klientskou část se používá JavaScript nebo jeho frameworky jako je Angular, React a další. Abychom tedy mohli programovat webové aplikace potřebujeme znát hned minimálně 2 jazyky. V roce 2018 přišla firma Microsoft s řešením. Představila světu Blazor.

Jedná se technologii, jenž umožňuje psát Javascript pomocí jazyka C#. To usnadní život spoustu programátorům, kteří mnohdy válčí s Javascriptem. Jak to vlastně ten Blazor dělá? Odpověď zní WebAssembly (WASM).

Při kompilaci Javascriptu, celý kód nejprve zpracuje Parser, jehož práce je projít kód řádek po řádku. Jestliže je syntaxe v pořádku, Parser vytvoří stromovou datovou strukturu AST. Tu poté překladač převede na IR, což je mezikód, který využívá jako mezikrok při kompilaci do strojového kódu. Poslední krok je samotný převod na kód strojový.

Staticky typované jazyky vyžadují deklaraci typů předem, proto se typy kontrolují se při kompilaci. Poté je vygenerován předkompilovaný modul WASM. Následně je možné tento kód spustit přímo kompilátorem, přeskočit Parser a transformaci na Intermediate Representation. [10]

2.2.4 Single Page Aplication

Jedná se o technologii, která navazuje na MVC. Pokud je potřeba aplikaci mít rychlou a menší, s malou zátěží na serveru, tak je lepší využít SPA. Tato technologie klade obrovský důraz na JavaScript na straně klienta. Se serverem komunikuje pomocí Web API. Celá aplikace, jak už z názvu vyplývá je pouze jednostránková. Ta se jednou načte ze serveru a můžeme ji používat, protože všechny funkce jsou na straně klienta. Stránku jde rozdělit na jednotlivé podstránky. Když ale mezi nimi přecházíme nenačítáme stránku se serveru pouze voláme Javascript na straně klienta.

Na server posílá buď XML nebo JSON a ve stejném formátu i ze serveru odchází. Šablony jsou vytvořené tak aby se rovnou vygenerovala stránka i s podstránky, které JavaScript skrývá. [9][3]

2.2.5 Signal R

Jedná se o open-sourcovou knihovna. Tato knihovna umí jednu zásadní věc, a to odesílat serveru asynchronní data z webového klienta. Znamená to, že díky této knihovně jsme schopni vytvářet aplikace v reálném čase.

Pro komunikaci v reálném čase Signal R používá tyto tři techniky. Techniky automaticky vybírá podle potřeby serveru a klienta.

- Websocket
- Server-Sent Events
- Long Polling
- Forever Frame

Komunikaci mezi klientem a serverem zařizuje takzvaný hub. Hub nám umožnuje vytvářet skupiny, díky nimž je možné posílat zprávy jen určité skupině klientů. Hub používá dvojici komunikačních modelů. PersistenConnection a Hub.PersistenConnection, tyto protokoly umožnují ovládání jednotlivých paketů a jejich zpráv.

Huby volají kód na straně klienta odesláním paketů, které obsahují název a parametry dané metody na straně klienta. Tyto objekty jsou deserializovány pomocí funkčního protokolu. Klient pokusí spárovat metodu, pokud existuje, zavolá metodu a předá do ní deserializovaná data parametru.[3][11]

3 TECHNOLOGIE POUŽITÉ V APLIKACI

Následující kapitola popisuje technologie použité pro tvorbu aplikace. V aplikaci byla použita technologie MVC a následující technologie.

3.1 ORACLE databáze

Pro ukládání dat je v práci použita databáze Oracle, což je multiplatformní databázový systém, s možností pokročilého zpracování dat a vysokým výkonem. Systém podporuje standartní relační dotazovací jazyk SQL, ale také rozšíření vytvořené přímo firmou Oracle. Pro práci s databází je využit Oracle SQL Developer. Jedná se o grafické prostředí díky, kterému nemusí autor všechny jednotlivé příkazy psát pomocí jazyka SQL. [12]

3.2 HTML

HyperText Markup Language je základním stavebním kamenem webových stránek. Jedná se o značkovací jazyk, kdy značky nebo tagy tvoří celou strukturu webové stránky, HyperText v názvu představuje hypertextovými odkazy, které vzájemně propojují jednotlivé stránky na webu. HTML představuje jen část webového programování abych byli schopni programovat vetší a funkční weby budeme potřebovat i CSS a Javascript viz. níže.

3.3 CSS

Cascading Style Sheets, kaskádové styly vznikly proto, aby programátorům zjednodušili programování webových stránek. Díky těmto stylům můžeme zapisovat některé značky přímo do značek jiných. Nebo můžeme dokonce vytvořit třídu a zde nastavit jednotlivé parametry a tento styl potom pouze v HTML volat.

3.4 Javascript

V předchozích kapitolách je hodně zmiňován Javascript, co to ale vlastně ten JavaScript je? Jedná se multiplatformní objektově orientovaný jazyk, jehož programy se nazývají scripty. Tyto klientské skripty se zapisují přímo do HMTL kódu. Existují i nějaké jiné skriptovací jazyky, ale ve směs se žádné v hojném množství nevyužívají.

3.5 Open XML SDK

Open office XML je knihovna pracující s Office Word, Excel a Powerpoint. Zvládá všechny typy operací, základní třeba jako zápis nebo čtení z jednotlivých souborů. Nebo třeba

slučování souborů, vyhledávání pomocí regulárních výrazů nebo jiné operace přímo v Microsoft dokumentu. [14]

II. PRAKTICKÁ ČÁST

4 STUDIE PROVEDITELNOSTI

První sběr požadavků na aplikaci proběhl prostřednictvím rozhovorů s autory původního řešení. Aplikace LogBook je primárně určena pro sledování výroby křemíkových desek. Sbírá a zobrazuje data z řídícího systému Promis a kategorizuje je. Data jsou zobrazeny v samostatných tabulkách pro sady nebo tasky v zjednodušeném nebo rozšířeném módu.

4.1 Cíle projektu

Jak bylo již řečeno aplikace LogBook zobrazuje data ze systému Promis. Bohužel ne všechny data se do systému nahrávají. Existují i sady mimo Promis, nebo třeba testovací sady, které se do aplikace nezapisují. Cíl projektu je přidat do stávající aplikace možnost přidání inženýrských záznamů. Tento způsob se má začít používat ve výrobě. Dnes operátoři používají papír a tužku, tudíž se jedná o rozvoj výrobního procesu. Dále si firma onsemi přeje přidat editaci těchto přidaných záznamů, aby byla možnost případě nějaké chyby záznam upravit.

4.2 Sběr požadavků

Na základě požadavků firmy onsemi byly definovány následující požadavky:

Funkční požadavky

- Přidání ručního záznamu
- Editace přidaného záznamu
- Autorizace při přidávání i editaci
- Auditování změn
- Dynamická validace zadaných dat

Nefunkční požadavky

- Vedení auditních záznamů
- Nápověda

Nebude se realizovat

- Uživatelská a programátorská dokumentace

4.3 Harmonogram projektu

Tabulka harmonogramu projektu byla vytvořena ke sledování progresu vývoje.

Tabulka 1. Harmonogram projektu

Úkol	Termín
Studie proveditelnosti	31.1.2022
Analýza	15.1.2022
Design aplikace - Návrh DB - Vytvoření projektu a jeho modulu	30.1.2022
Realizace - Úprava databázové vrstvy - Realizace programového kódu (kódovaní)	10.5.2022
Interní testy	10.6.2022
Dodělání dalších požadavků s nízkou prioritou	30.6.2022

4.4 Mind model aplikace

Jedná se o diagram (obrazec, schéma) představující slova, myšlenky, představy a úkoly uspořádané paprskovitě kolem centrálního slova nebo myšlenky. Myšlenkové mapy se používají ke grafickému záznamu myšlenek a vizualizaci myšlenkového procesu. Mohou být využity při studiu, organizaci, řešení problémů, rozhodování, psaní a řadě dalších činností. Na následujícím obrázku 4 je vytvořena myšlenková mapa pro LogBook. [15]

Obrázek 4 - Myšlenková mapa

5 POPIS EXISTUJÍCÍHO ŘEŠENÍ

V následujících kapitolách bude popsán současný stav aplikace.

5.1 Uživatelské rozhraní aplikace

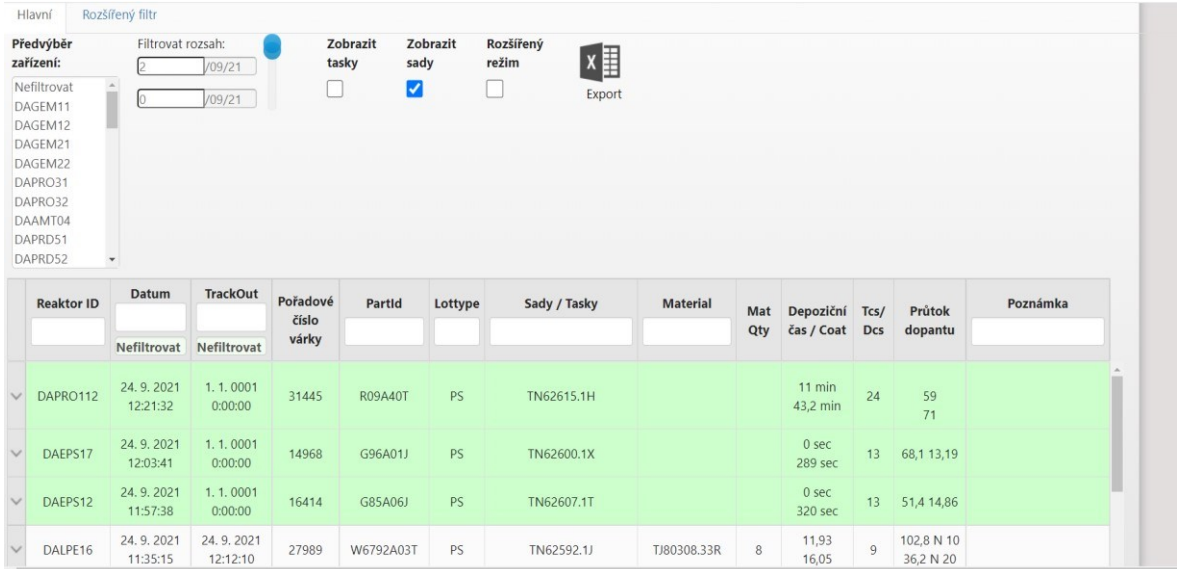
Aplikace je napsaná v jazyku c# a jedná se o webovou aplikaci typu MVC vytvořenou pomocí frameworku ASP.NET MVC. Aplikace umí základní filtraci dat pomocí ReaktorID jak můžeme vidět v levém okně, dále filtruje rozsah, defaultně je nastaven na poslední dva dny. V check boxech můžeme zvolit, zda chceme zobrazit tasky, sady nebo rozšířený režim.

- Tasky: zobrazí tasky
- Sady: zobrazí sady
- Rozšířený režim: zobrazí i defaultně skrytá data

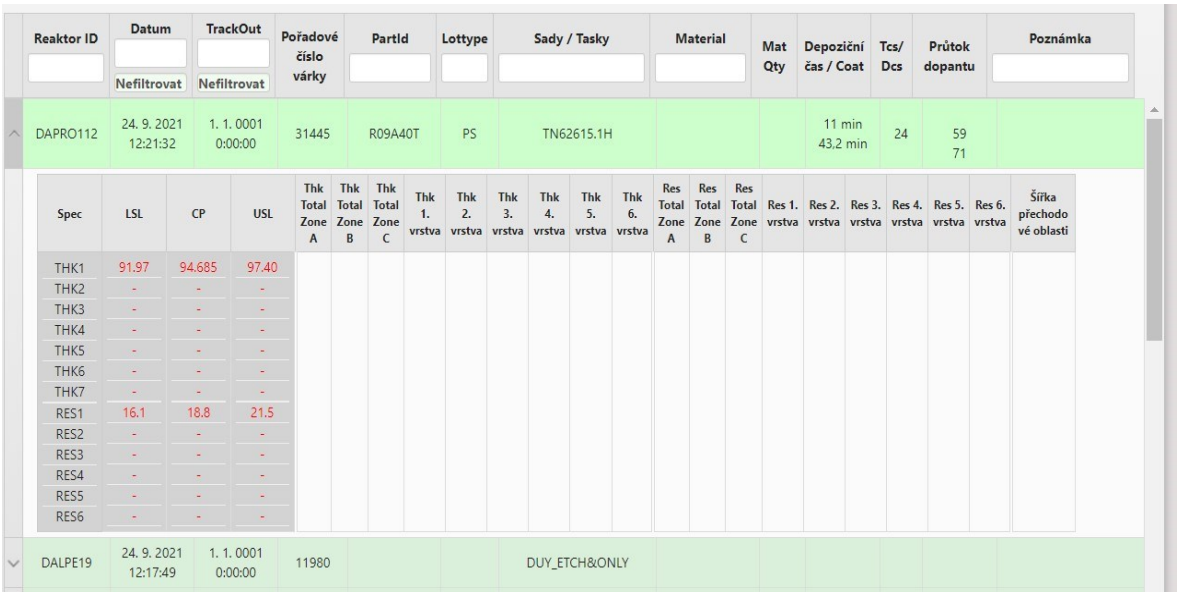
Poslední tlačítko je na export do excelu, který vyexportuje tabulku dat do excelu.

Zde v tomto prostoru, v této navigaci se bude nacházet i nově vytvořené tlačítko pro vytvoření nového záznamu.

V rozšířeném režimu můžeme rovnou vyhledat data pomocí specifických parametrů. Pod navigací se nachází samotná tabulka s daty. Po kliknutí na řádek se zobrazí detail tabulky, ve kterém se nacházejí konkrétnější data. Na obrázku níže vidíme, jak aplikace vypadá po zapnutí LogBooku Epi.



Obrázek 5 - Základní pohled původní aplikace



Reaktor ID	Datum	TrackOut	Pořadové číslo várky	PartId	Lottype	Sady / Tasky	Material	Mat Qty	Depoziční čas / Coat	Tcs/ Dcs	Průtok dopantu	Poznámka
	Nefiltrovat	Nefiltrovat										
DAPRO112	24. 9. 2021 12:21:32	1. 1. 0001 0:00:00	31445	R09A40T	PS	TN62615.1H			11 min 43,2 min	24	59 71	

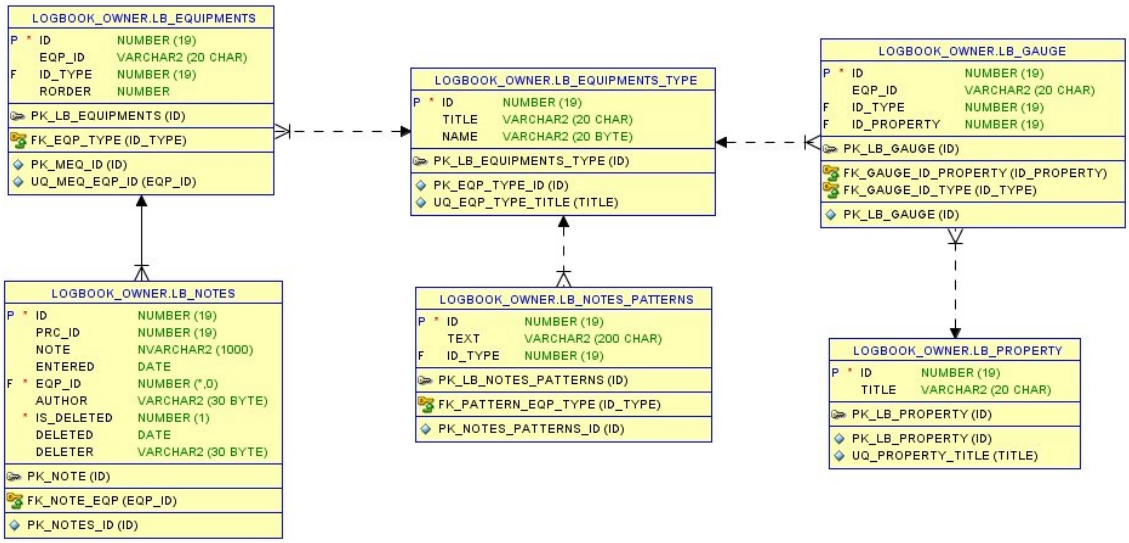
Spec	LSL	CP	USL	Thk Total Zone A	Thk Total Zone B	Thk Total Zone C	Thk 1. vrstva	Thk 2. vrstva	Thk 3. vrstva	Thk 4. vrstva	Thk 5. vrstva	Thk 6. vrstva	Res Total Zone A	Res Total Zone B	Res Total Zone C	Res 1. vrstva	Res 2. vrstva	Res 3. vrstva	Res 4. vrstva	Res 5. vrstva	Res 6. vrstva	Šířka přechodové oblasti
THK1	91.97	94.685	97.40																			
THK2	-	-	-																			
THK3	-	-	-																			
THK4	-	-	-																			
THK5	-	-	-																			
THK6	-	-	-																			
THK7	-	-	-																			
RES1	16.1	18.8	21.5																			
RES2	-	-	-																			
RES3	-	-	-																			
RES4	-	-	-																			
RES5	-	-	-																			
RES6	-	-	-																			

DALPE19	24. 9. 2021 12:17:49	1. 1. 0001 0:00:00	11980			DUY_ETCH&ONLY						

Obrázek 6 - Detail původní aplikace

5.2 Původní datový model

Většina dat se získává ze systému PROMIS, což je databáze Torrent. Tato databáze ale v práci využita nebude a místo toho bude využita databáze CZ4AUT. V této databázi se nachází poznámky a nově zde budou uložena i data pro ruční přidávání inženýrských záznamů.



Obrázek 7 - Původní datový model

CZ4AUT

Databáze, která je nutná z důvodu přidávání poznámek k jednotlivým záznamům.

Torrent

Jedná se o databázovou kopii dat systému PROMIS, tato databáze je rozsáhlá a nachází se v ní miliony záznamů přímo z výroby. Původní autor aplikace pomocí této databáze čte data ze systému PROMIS.

5.3 Popis projektu ELogBook

5.3.1 Projekt OnSemi.Ei.ELogBook.Extensions

Obsahuje několik pomocných funkcí, které se používají v jiných projektech.

5.3.2 Projekt OnSemi.Ei.ELogBook.Resources

Pro každý jednotlivý logbook obsahuje resources, tedy zdroje, řetězce, textové popisky – které je možné využít v logboocích – hlavně tedy v GUI aplikace (v prohlížeči).

5.3.3 Projekt OnSemi.Ei.ELogBook.DataAccessLayer

Společný projekt pro získávání dat, hlavně z Torrentu. Používá funkce z OnSemi.Ei.ELogBook.Extensions. Nejdůležitější složka je DbConnection, obsahuje SQL dotazy pro dotazování do různých databází, hlavně tedy Torrentu a databáze LogBooku.

Podsložka SQL_Torrent obsahuje SQL příkazy pro dotazování na data do Torrentu. Konkrétně EPI logbook používá dotazy začínající Alias*. Třída, která dotazy provádí a vrací data, je DbConnectionTorrent.cs. Podsložka SQL_ElogBook obsahuje SQL dotazy pro přístup do databáze Logbooku, třída, která dotazy provádí a vrací data je DbConnectionELogBook.cs.

5.3.4 Projekt OnSemi.Ei.ELogBook.ViewModels

Používá hojně OnSemi.Ei.ELogBook.DataAccessLayer a občas také OnSemi.Ei.ELogBook.Extensions. Pro každý jednotlivý logbook obsahuje složku a v ní základní třídy pro předávání do webové části tedy do OnSemi.Ei.ELogBook, a to ProcessRow.cs, SubDetailRow.cs a SubDetailTask.cs. Třída FiltersModel.cs slouží pro pomocné filtrování vyhledaných záznamů podle doplňkových kritérií, jako jsou např. PartId, LotType, Sady, Material, Datum, Poznámka apod. Filtrační textové pole jsou v

prohlížeči hned v hlavičce tabulky nad daty. DataService.cs je třída, která tahá data potřebná v logbooku. Obsahuje metody jako GetProcesses pro získání sad a dat, podle hlavních kritérií, jako jsou počet dní nebo časový interval ze kterého data získat. GetTasks metoda pro získání tasků a dat k nim podle hlavních kritérií. GetProcessDetail se volá pro získání dat k detailu jednoho řádku, sady. GetTaskDetail pro získání dat k detailu jednoho tasku.

Třída ViewModel.cs je „nadřízeným“ všech těchto tříd pro konkrétní logbook, kde dává dohromady data ze třídy DataService.cs, filtruje je pomocí FiltersModel.cs, případně umí pár dalších funkcí. Společné třídy s prefixem Base jsou společným předkem tříd zmíněných výše, např. BaseDataService.cs je předkem tříd DataService.cs pro všechny logbooky a obsahují několik funkcí, které se dají použít v jejich dceřiných třídách.

5.3.5 Projekt OnSemi.Ei.ElogBook

Webová část projektu, používá všechny ostatní výše zmíněné projekty. Je to ASP.NET MVC5 projekt. Složka Controllers obsahuje kontrolér ke každému logbooku. Kontrolery obsahují metody, které je možné volat „zvenku“, tzn. z prohlížeče. Obsahuje další složky ke každému logbooku. Ty obsahují views, tedy pohledy, samotné UI. Složka Scripts obsahuje JavaScripty. Soubor s kódem LogBooku jsou dialog.js, ten řeší otevření dialogu s poznámkami, je tam i použití Promis přihlašovacího dialogu. V souboru Grid.js se nachází logika pro zobrazení a obsluhu eventů v tabulce. Helper.js, pomocná třída s několika obecnými funkcemi. Index.js obsahuje všechny ostatní Javascriptové funkce, automatické obnovení stránky, přepínání mezi hlavní a rozšířeným režimem, parsování dat a ostatní. Složku Content tvoří obrázky a kaskádové styly.

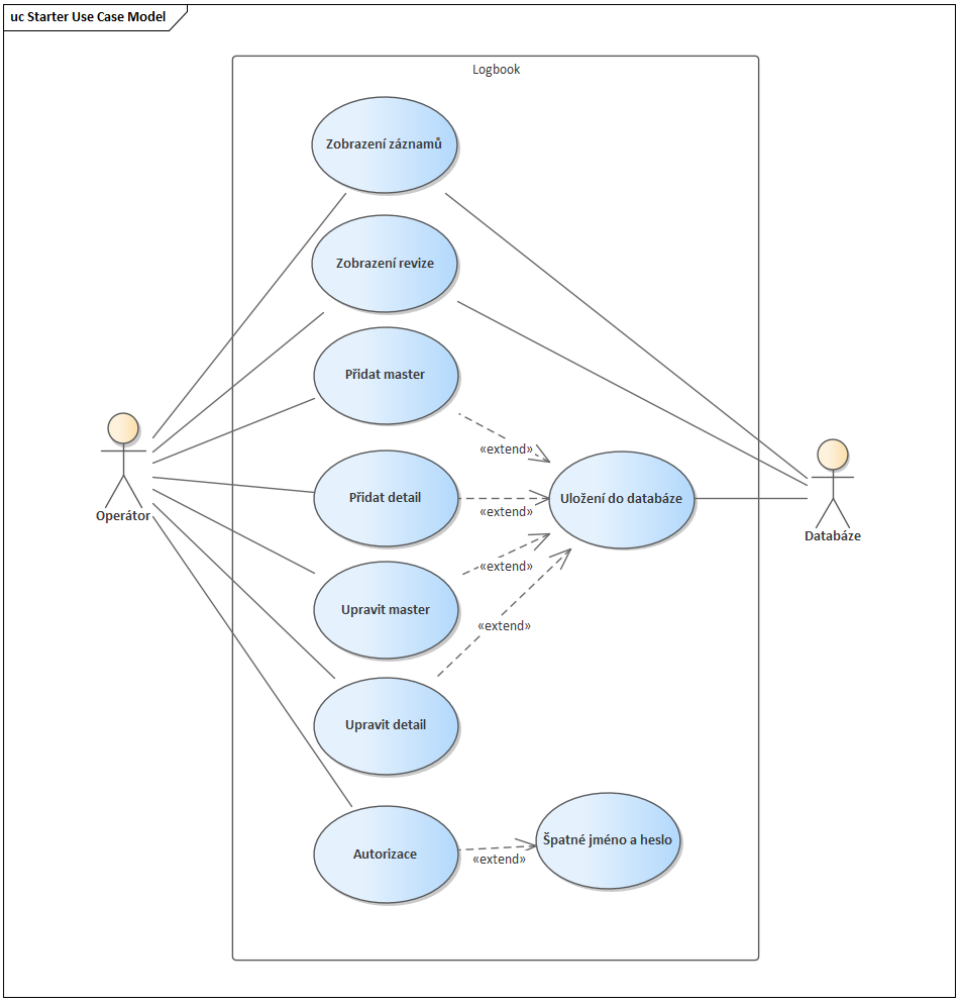
6 NÁVRH APLIKACE

V následujících kapitolách budou popsány případy užití, návrh databáze a návrh modelů uživatelského rozhraní.

6.1 Případy užití

Případy užití nastiňují chování systému, který reaguje na požadavek. Diagramy případů užití jsou jednoduché diagramy, které vizuálně popisují cíle uživatelů s ohledem na systém. Dalo by se to říct, že tyto diagramy z pohledu uživatele nastiňují chování systému, který reaguje na požadavek.

Obrázek 8 představuje diagram případu užití návrhu aplikace. V tomto případě aktér, který používá aplikaci je operátor, ten má přístup k jednotlivým funkcím zobrazení záznamů a revize. Přidání a editaci záznamu Master a záznamu Detail, tyto data se musí uložit do databáze. Poslední funkce je funkce autorizace.



Obrázek 8 - Diagram případu užití

6.2 Návrh databáze

Firma onsemi již roky aktivně používá Oracle databáze. Pro tento projekt je nutné upravit databázi CZ4AUT. Tato databáze je již v aplikaci používána, a to pro přidávání a odebírání poznámek. Databáze má tři připojení a to Owner, User a Read. Owner obsahuje veškeré tabulky a data. User a Read jsou pouhé odkazy na tabulku Owner s jednotlivými pravomocemi.

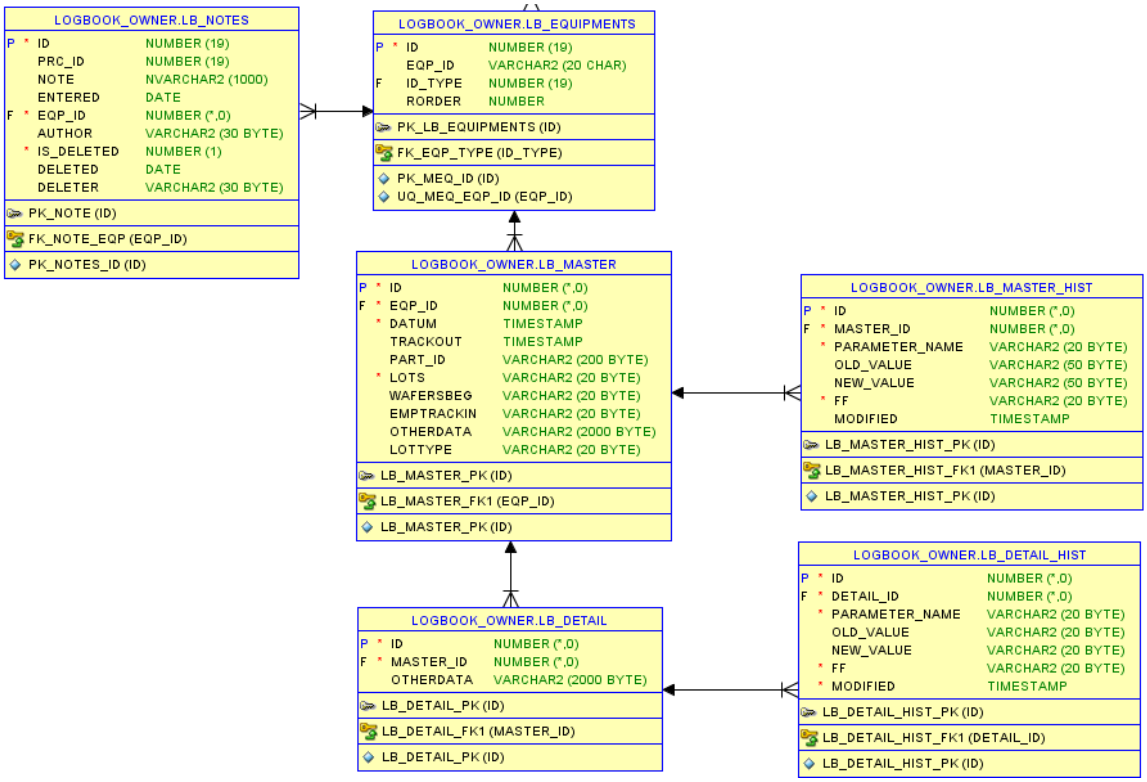
V databázi, budou vytvořeny celkem čtyři nové tabulky, jedná se o tabulku LB_Master, kde se uloží data stejná jako jsou v řádku. Hlavní parametry budou mít své sloupce, ostatní záznamy se zapíší do sloupce s názvem OTHERDATA ve formátu JSON. Tabulka využije již vytvořené tabulky LB_EQUIPMENTS, která obsahuje všechny používané reaktory.

Další tabulka LB_MASTER_HIST bude obsahovat tyto sloupce MASTER_ID, PARAMETR_NAME, OLD_VALUE, NEW_VALUE, FF, MODIFIED.

MASTER_ID bude odkazovat na předchozí tabulku. Díky tomuhle sloupci, který je cizím klíčem, se zajistí, že každý řádek bude mít svou vlastní tabulku s historii.

PARAMETR_NAME bude obsahovat jednotlivý název daného parametru. OLD_VALUE patří k hodnotě, kterou budeme měnit. Hodnotu novou reprezentuje sloupec NEW_VALUE. FF je jméno nebo identifikační číslo uživatele. A sloupec MODIFIED znázorní čas, kdy k dané změně došlo.

Tabulka LB_DETEAIL, je velice jednoduchá obsahuje jen odkaz na LB_MASTER, ke kterému se váže. A sloupec OTHERDATA, v němž jsou opět data zapsány pomocí datové struktury JSON. Poslední tabulka bude úplně stejná jako Tabulka LB_MASTER_HIST jen místo odkazu na tabulku LB_MASTER se bude odkazovat na LB_DETAIL.



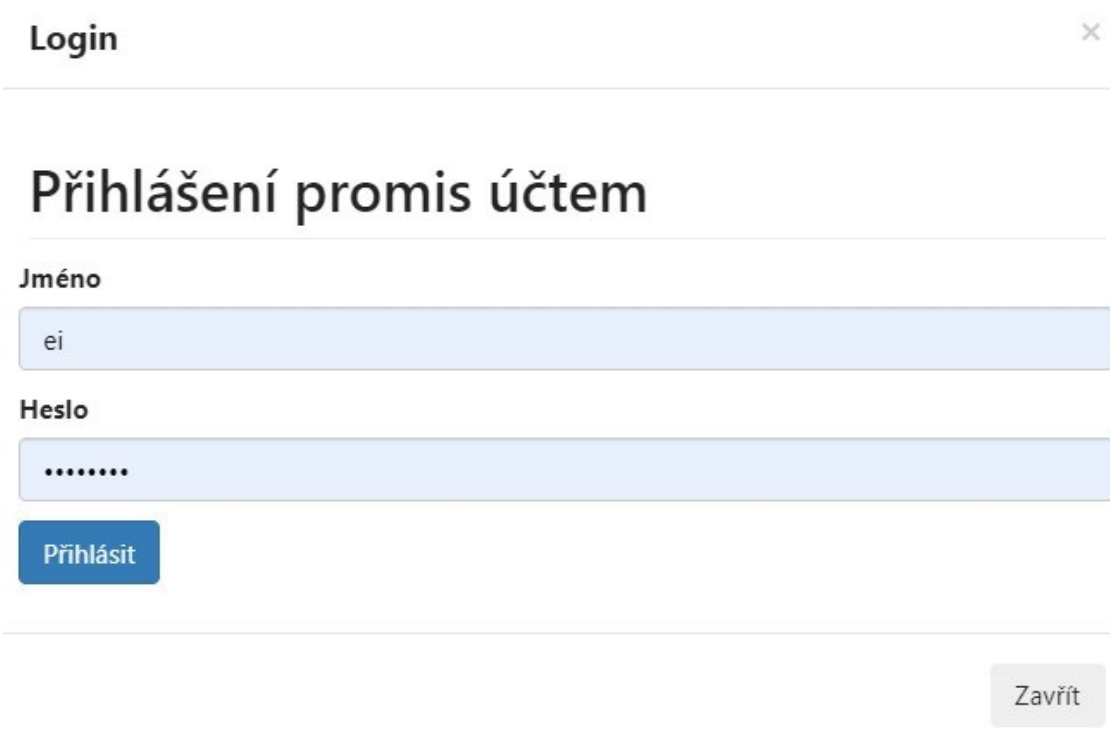
Obrázek 9 - Návrh databáze

6.3 Návrh modelů uživatelského rozhraní

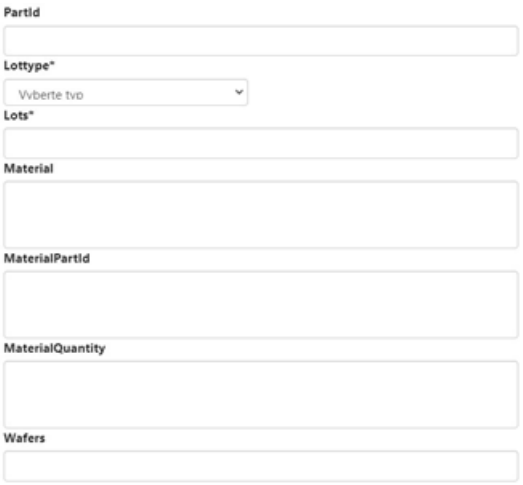
Při navrhování formuláře pro přidávání a editaci nových údajů se autor snažil zachovat původní standardy uživatelského rozhraní zobrazené na příkladu přihlášení do systému viz obrázek 11.

Návrh nových formulářů byl vytvořen pomocí Frameworku boostrap. Konkrétně byl navržen formulář pro přidávání dat typu popup viz obrázek 10. Popup formulář funguje jako vnitřní stránka, která se otevře při zmáčknutí tlačítka. Tato stránka se nikam nepřesměruje, tudíž je to rychlejší a také praktičtější. Formulář obsahuje název jednotlivých sloupců a textové pole pro vyplnění dat.

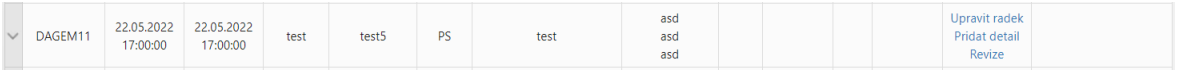
Dále byl navržen návrh změny řádku s funkcemi pro přidání a úpravu řádku a přidání revize pomocí hypertextových odkazů viz obrázek 12. Také bylo navrženo uživatelské rozhraní pro detail s možností zobrazení a úpravy detailu viz obrázek 13. A nakonec byla navržena tabulka revize viz obrázek 14.



Obrázek 11 - Formulář autorizace

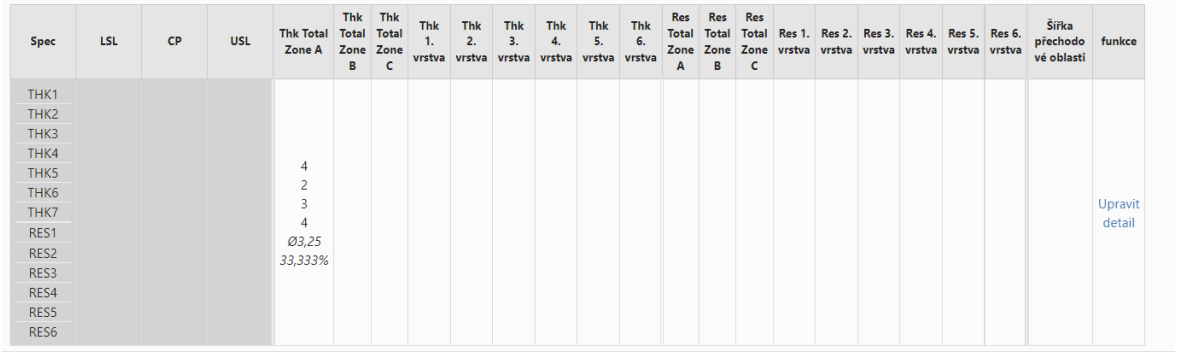


Obrázek 10- Formulář pro přidávání dat



DAGEM11	22.05.2022 17:00:00	22.05.2022 17:00:00	test	test5	PS	test	asd asd asd					Upravit radek Pridat detail Revize	

Obrázek 12 - Návrh řádku



Spec	LSL	CP	USL	Thk Total Zone A	Thk Total Zone B	Thk Total Zone C	Thk 1. vrstva	Thk 2. vrstva	Thk 3. vrstva	Thk 4. vrstva	Thk 5. vrstva	Thk 6. vrstva	Res Total Zone A	Res Total Zone B	Res Total Zone C	Res 1. vrstva	Res 2. vrstva	Res 3. vrstva	Res 4. vrstva	Res 5. vrstva	Res 6. vrstva	Šířka přechodové oblasti	funkce
THK1 THK2 THK3 THK4 THK5 THK6 THK7 RES1 RES2 RES3 RES4 RES5 RES6				4 2 3 4 *Ø3,25* *33,333%*																			Upravit detail

Obrázek 13 - Návrh detailu

#	Parametr Name	Old Value	New Value	FF	Modified
1	ThkZoneA	1;2;3;4	4;2;3;4	EI Account	22.05.2022 20:07:46
2	EqpId	47	45	EI Account	22.05.2022 17:42:57
3	CreateDate	22.05.2022 9:00:00	22.05.2022 17:00:00	EI Account	22.05.2022 17:42:57
4	TrackOut	22.05.2022 9:00:00	22.05.2022 17:00:00	EI Account	22.05.2022 17:42:57

Obrázek 14 - Návrh tabulky revize

7 REALIZACE APLIKACE

Na základě požadavků na aplikaci, návrhu databáze a formulářů byla vytvořena implementace klíčových částí aplikace.

7.1 Implementace tlačítka

V layout.cshtml je vloženo tlačítko vedle checkboxu, dokonce je využit i jejich styl pro div, aby vše zůstalo jednotné.

Jelikož bylo potřeba, aby se tlačítko ukazovalo jen když se jedná o Loogbook s názvem Epi, byla přidána jednoduchá podmínka.

```
@if (MvcApplication.LbEquipmentType == "Epi")
  {
    <div class="checkboxdiv">
      <input type='button'  id="addRow" class="addrow" onclick="saveRefresh()" style="display: inherit"/>
    </div>
  }
```

Projekt MvcApplication.LbEquipmentType načítá data z konfiguračního souboru Web.config, kde je specifikované o jaký logbook se jedná. Jelikož má stránka nastavené automatické obnovení, aby se aktualizovaly data z databáze. Metoda *saveRefresh* již v projektu existovala a zabraňuje tomuto automatickému obnovení.

Po klinutí na tlačítko se zavolá metoda. Tato metoda je event na kliknutí a zavolá se pokud se jedná o dané id nebo třídu. Před id je přidán hashtag a před třídu, se dává tečka.

```
$("#addRow").on("click", function (e) {       login("AddRowDialog",this);
    loggedin = "addRow";
    ID = 0;
  });
```

Metoda volá další metodu, a to metodu pro autorizace. U toho ještě nastaví nějaké globální proměnné.

7.2 Vykreslovaní řádku

V projektu Onsemi.Ei.Elogbook.ViewModels existuje třída jménem *Getprocess*, tato třída posílala dotazy na SQL server a získavala kolekci dat, které se vykreslí v tabulce. Nové metody, které byly vytvořeny, se jmenují *GetProcesses*, *GetproccesEpi*, *GetproccesTorrent*. V metodě *GetProcesses* jsou zavolány zbylé metody, které obě vrací list hodnot. Tyto listy jsou spojeny pomocí metody *Concat*.

```
public List<ProcessRow> GetProcesses()

    {

      var torrentList = GetProcessTorrent();

      var epiList = GetProcessEpi();

      var a = torrentList.Concat(epiList).ToList();

       return a;

    }
```

7.2.1 Vykreslování řádků Epi

Metoda *GetProccesEpi* vykresluje řádky, které budou ručně přidávány. Pro ukládání byla vytvořena nová datová struktura jménem *FullBatch*, která reprezentuje databázi a strukturu *ProccesRow*, jenž využívá aplikace pro vykreslování tabulky. Postup je obdobný jako u metody getProccesTorrent, která obsahuje většinu z původní metody *GetProcess*, kde se většina dat ukládala do datové struktury *Batch*.

```
public List<ProcessRow> GetProcessEpi()
    {
try {
          var batches = ConnectionEpi.GetData(_daysFrom, _daysTo, _advancedFilterValue,
_advancedFilterSpec.ColumnName, _advancedFilterSpec.MonthsFrom, _advancedFilterSpec.MonthsTo);

          var tmpProcesses = batches.Select(b => new ProcessRow(b)).ToList();


          return tmpProcesses;
       }
       catch(Exception e)
       {
          var msg = $"GetProcesses error: {e.Message}";
          Log.Warn(msg);
          throw;
       }
    }
```

Dotazování na server je vytvořeno v metodě *GetData*, použije se již inicializovaný *connection* jménem *ConnectionEpi*. Argumenty metody jsou parametry pro SQL dotaz, například jak staré hodnoty se budou hledat. V Oracle dotazu se přepíše daný filtr podle potřeby viz. *conditionString*. Po provedení dotazu jsou všechna data jsou zapsána do jednotlivých proměnných. Sloupec *OTHERDATA* je zapsán do objektu typu *string*, ale musí být předán přímo do další datové struktury *FullBatchData*. Jenž je použita ve struktuře *FullBatch*. Data jsou poté zapsána do listu a poté vrácena.

7.3 Autorizace

Po kliknutí na tlačítko přidat či později i upravit, se zavolá metoda *Login*. Tato metoda má dva parametry, prvním je název metody v kontroléru, do které chceme být přesměrování po dokončení přihlášení a referenci *this*, která odkazuje například na jednotlivé řádky tabulky, budeme ho potřebovat, když budeme chtít data upravovat.

Pokud bylo přihlášení úspěšné, zavolá se metoda *loginFinishEventHandler*, metoda má jeden parametr, a to jsou data obsahující informaci, jestli přihlášení proběhlo v pořádku anebo ne. Pokud přihlášení neproběhlo v pořádku, tak se zobrazí chybová hláška, pokud má *data.status* hodnotu *ok*, voláme jednotlivé metody. Podle toho, které tlačítko bylo zmáčkuto, vybere se další Javascriptová metoda a opět si přeneseme potřebná data, a to je jméno uživatele, metody a id. Tato metoda pomocí technologie AJAX přenese data do kontroléru.

```
function loginFinishEventHandler(data) {
if (data.status.toLowerCase() === 'ok') {
      loginSuccess = true;
logged = data.user;
      // success login
      $('#myModal .modal-body').html("<div class='loadermini'></div>");

      if (loggedin === "notes") {
         callGetNotes(GlobalSettings.Controller + '/NotesDialog',              dialogLastRowId,
dialogLastRowEqpId,              dialogLastThisObj);
      }
      else if (loggedin === "addRow") {
         callGetRow(GlobalSettings.Controller + '/AddRowDialog', ID, data.user);

      }
      else if (loggedin === "hist") {
callGetHist(GlobalSettings.Controller + '/HistDialog', ID);
      }
      else if (loggedin === "detail") {
         callGetDetail(GlobalSettings.Controller + '/AddDetailDialog', ID, MasterID, data.user);
      }
   }
```

```
    else
    {
        $('#myModal .modal-body').html('špatné přihlašovací údaje');
    }
}
```

7.3.1 Vykreslení dat do formuláře

Metoda *AddRowDialog* v třídě EpiController ověřuje, zda přenášíme Id. Pokud ano, jedná se o úpravu, pokud ne, tak se jde o vytvoření nového řádku.

```
public ActionResult AddRowDialog(ProcessRow rm)
    {

        var processRow = new ProcessRow();
        if (rm.Id == 0) return PartialView("_AddRowDialog", processRow);

        var dbConnectionNotes = new DbConnectionELogBook();
        dbConnectionNotes.Init(MvcApplication.ELogBookConnectionString);
        var batch = dbConnectionNotes.GetSingleData(rm.Id);
        processRow = new ProcessRow(batch);

        return PartialView("_AddRowDialog", processRow);
    }
```

V partial view se nachází celý formulář pro přidání nového záznamu. Pro ukázku je přiložený formulář v kapitole 6.3. Formulář se skládá se dropdownů, textových polí a elementů *Textarea*. Element *Textarea* je použit abychom mohli do jednoho řádků vložit více hodnot, jednotlivé hodnoty oddělujeme entrem, tak aby to bylo uživatelsky přívětivé. Názvy označené hvězdičkou jsou povinné.

7.4 Ukládání řádku

Data pro ukládání musí být nejprve převedena a pro všechny data z *ProccesRow* vložíme do *Batche*. Dále kontroler validuje, jestli znovu nepřenáší id. Jestliže ne, zavolá se metoda *InsertRowData*, ještě před ní, ale musí být upraveny jednotlivá data. Jedná se o sloupce, které mohou obsahovat vícero hodnot, ty jsou poslány do metod, jenž odstraní řádkování a jsou zapsaný jako hodnoty oddělené čárkou či středníkem.

```
public ActionResult SaveRow(ProcessRow rm)
    {
        var connectionELogBook = new DbConnectionELogBook();
        connectionELogBook.Init(MvcApplication.ELogBookConnectionString);
        var data = new FullBatchData
        {
            RunId = rm.RunId,
            Material = rm.Material,
            MaterialPartId = rm.MaterialPartId,
            MaterialQuantity = rm.MaterialQuantity,
```

```
            DepositionTime = rm.DepositionTime,
            Dcs = rm.Dcs,
            Dopants = rm.Dopants,
            EmpTrackOut = rm.EmpTrackOut,
            GrowthSpeed = rm.GrowthSpeed,
            SusceptorId = rm.SusceptorId,
            SusceptorRun = rm.SusceptorRun,
            StepCmt = rm.StepCmt,
            Order = rm.Order,
            TlMean = rm.TlMean,
            Etch = rm.Etch,
        };

        var fullBatch = new FullBatch
        {

            EqpId = EqpToInt(rm.EqpId),
            CreateDate = rm.CreateDate,
            TrackOut = rm.TrackOut,
            PartId = rm.PartId,
            Lottype = rm.Lottype,
            Lots = rm.Lots,
            Wafers = rm.Wafers,
            EmpTrackIn = rm.EmpTrackIn,
            Data = data
        };

        if (rm.Id == 0)
        {

            fullBatch.Data.Material = fullBatch.Data.Material.BreakLineToComma();
            fullBatch.Data.MaterialQuantity = fullBatch.Data.MaterialQuantity.BreakLineToComma();
            fullBatch.Data.MaterialPartId = fullBatch.Data.MaterialPartId.BreakLineToComma();
            fullBatch.Data.DepositionTime = fullBatch.Data.DepositionTime.BreakLineToSemicolon();
            connectionELogBook.InsertRowData(fullBatch);
        }
        else
        {
            fullBatch.Id = rm.Id;
            var oldbatch = connectionELogBook.GetSingleData(rm.Id);
            var masterComparedList = new List<Variance>();
            masterComparedList = oldbatch.DetailedCompare(fullBatch, masterComparedList,rm.EmpTrackIn);

            fullBatch.Data.Material = fullBatch.Data.Material.BreakLineToComma();
            fullBatch.Data.MaterialQuantity = fullBatch.Data.MaterialQuantity.BreakLineToComma();
            fullBatch.Data.MaterialPartId = fullBatch.Data.MaterialPartId.BreakLineToComma();
            fullBatch.Data.DepositionTime = fullBatch.Data.DepositionTime.BreakLineToSemicolon();
            connectionELogBook.UpdateRowData(fullBatch, masterComparedList);

        }

        return null;
    }
```

Metoda *InsertRowData*, která přenáší data z *batche* využívá vytvořeného sql dotazu, který binduje jednotlivá data.

```
INSERT INTO LB_MASTER
```

```
(EQP_ID, DATUM, TRACKOUT, PART_ID,LOTS,LOTTYPE, WAFERSBEG, EMPTRACKIN, OTHERDATA)
VALUES (:EQP_ID,:DATUM,:TRACKOUT,:PART_ID,:LOTS,:LOTTYPE,:WAFERSBEG,:EMPTRACKIN,:OTHERDATA)
```

Data jsou nabindovány ve správných formátech a celý SQL dotaz je zavolán. Pokud by se objevila nějaká chyba, tak se vypíše do konzole.

```
public void InsertRowData(FullBatch batches)
    {
      try
      {
        if (!Open())
        {
          return;
        }
        _cmdInsertRow.Parameters.Add("EQP_ID", Convert.ToInt64(batches.EqpId));
        _cmdInsertRow.Parameters.Add("DATUM", batches.CreateDate);
        _cmdInsertRow.Parameters.Add("TRACKOUT", batches.TrackOut);
        _cmdInsertRow.Parameters.Add("PART_ID", batches.PartId);
        _cmdInsertRow.Parameters.Add("LOTS", batches.Lots);
        _cmdInsertRow.Parameters.Add("LOTTYPE", batches.Lottype);
        _cmdInsertRow.Parameters.Add("WAFERSBEG", batches.Wafers);
        _cmdInsertRow.Parameters.Add("EMPTRACKIN", batches.EmpTrackIn);
        _cmdInsertRow.Parameters.Add("OTHERDATA", batches.Data.SerializeObjectToJson());
        _cmdInsertRow.ExecuteNonQuery();
      }
      catch (Exception e)
      {
        Console.WriteLine(e);
        throw;
      }
      finally
      {
        Close();
      }
```

Pokud dojde k úspěšnému přidání, řádek bude vidět v tabulce na hlavní stránce. Uživatel může vyplnit vícero dat na jedno přihlášení, přihlášení samo o sobě nějakou chvíli zůstává v paměti. A při otevřeném dialogovém okně se stránka neobnoví.

7.5 Ostatní funkce

Abychom mohli zavolat jednotlivé funkce jsou v řádku vytvořeny dva nové sloupce, jeden se jmenuje *IsAdded* a jedná se o skrytý sloupec. Pokud má hodnotu *true*, tak se do dalšího sloupce jménem *Funkce* vloží tři hypertextové odkazy, jedná se odkazy na dané metody viz. Obrázek 12.

7.5.1 Úprava řádků

Pro úpravu řádku se zavolá stejná funkce jako pro přidávání řádků. Tentokrát se do předá jako argument řádek s příslušným id. Formulář se vykreslí už předvyplněný a uživatel je

schopný provádět změny. V metodě se ověří, zda id existuje, pokud ano jedná se o úpravu řádku. Data se zpracují do správného formátu a porovnají se s původními daty pomocí metody *DetailedCompare*.

7.5.2 Přidaní, vykreslení a úprava detailu

Pro přidání detailu existuje podobný formulář jako pro Master, jen z jinými hodnotami. Vykreslení detail je složitější, nenačítá se totiž přímo do *ProcessRow*, ale načítá se až když je kliknuto na zobrazení detailu. Metoda přečte id masteru a podle něj se detaily vyhledají v databázi. Při vykreslovaní je taky počítána variabilita a průměr. Úprava detailu funguje totožně jako úprava masteru. Kontroluje se, zda se přenáší id, pokud ano upraví se data v databázi a zapíšou se změny do revize.

7.5.3 Revize

V databázi existují dvě tabulky pro zapisování změn, ale pro výpis jsou sloučeny, Metoda získá všechna data vázající se k danému řádku a vypíše je. Zobrazeny jsou tabulka viz obrázek 14.

8 ZHODNOCENÍ VÝSLEDKŮ A MOŽNOSTI DALŠÍHO VÝVOJE

8.1 Zhodnocení výsledků

Aplikace logbook konktrétně logbook Epi, byla upravena do funkční podoby. Vytvořena byla funkce pro přidávání dat s autorizací. Data se přidávají pomocí dvou formulářů, jeden přidává hodnoty do tabulky Master, jedná se o základní data, která jsou validována pomocí regulárních výrazů, data nutná pro vytvoření nového záznamu jsou povinná. Po vytvoření zmíněného řádku, je nutné přidat detail. V logbooku Epi se váže k jednomu řádku pouze jeden detail, v jiných logboocích jich může být i více. Pro vytvoření detailu je použit druhý formulář, do kterého se vyplňují číselné hodnoty. Z těchto hodnot je následně vypočítán průměr a výtěžnost. Úprava těchto hodnot je řešena pomocí stejného formuláře, jenž se otevře s již předvyplněnými údaji podle konkrétních hodnot. Všechny upravené hodnoty se zapisují do tabulky MasterHist a DetailHist. Zapisují se původní a upravené hodnoty, čas změny a autorizovaný autor dané změny. Tabulka změn je volně k nahlédnutí v aplikaci.

Po prvotním nasazení a vyzkoušení aplikace přišla od firmy onsemi první zpětná vazba. Výtěžnost se nepočítala správně, automatické obnovení, které bylo zkráceno na jednu minutu fungovalo i když byl formulář otevřený, což způsobilo zavření formuláře. Po opravení zmíněných připomínek přišlo onsemi ještě s dalšími změnami, a to odebrání některých hodnot z formulářů, automatické před vyplňování datumu. Po těchto provedených změnách byla firma onsemi spokojená a aplikaci nasadila. Žádná další zpětná vazba nepřišla, a proto je možné předpokládat že je aplikace funkční.

8.2 Možnosti dalšího vývoje

Celá aplikace je vytvořena pro konkrétní logbook, ale kód byl navržen co nejvíce obecně tak, aby se funkce mohly použít i v jiných logboocích. V současné době takový požadavek není, ale v budoucnu se uvažuje o možnosti přidání těchto funkcí do všech logbooků. Tato práce by však byla hodně náročná, jelikož každý je logbook je specifický.

Další možnost vývoje tohoto konkrétního logbooku je přidání odkazů na grafy ze systému Space. S firmou bylo dohodnuto že tento bod zadání nebude prioritou pro tuto bakalářskou práci, protože propojení jednotlivých řádků s jednotlivými grafy či skupinou grafů nebylo v době zadání práce systémem podporováno. Jelikož bylo domluveno pokračování spolupráce po dokončení bakalářské práce, tak tento bod by měl být jedním prvním z úkolů.

ZÁVĚR

Cílem práce bylo vytvoření funkčního prototypu aplikace pro monitorování procesu výroby firmy ON Semiconductor Czech Republic, s.r.o. V průběhu práce došlo k častým změnám v návrhu aplikace z důvodu složitosti současného řešení. V teoretické části jsou popsané především technologie .NET, ASP.NET. V praktické části jsou definovány funkční a nefunkční požadavky na aplikaci, případy užití a je navržený databázový model. Na základě návrhu je potom vytvořen prototyp rozšíření aplikace a jsou popsány jeho klíčové časti.

V současné době probíhá testování aplikace v praxi u zadavatele a první ohlasy jsou pozitivní a firma projevila zájem o pokračování tohoto projektu.

SEZNAM POUŽITÉ LITERATURY

[1] *Stackoverflow Developer Survey 2021* [online]. Stack Exchange [cit. 2022-05-22]. Dostupné z: https://insights.stackoverflow.com/survey/2021

[2] *Jquery* [online]. [cit. 2022-05-22]. Dostupné z: https://jquery.com/

[3] *Microsoft* dokumentace [online]. Microsoft [cit. 2022-05-22]. Dostupné z: dotnet.microsoft.com

[4] *Softeco* [online]. [cit. 2022-05-22]. Dostupné z: www.softteco.com

[5] .NET Framework vs .NET Core vs .NET vs .NET Standard vs

C#. www.youtube.com [online]. IAmTimCorey [cit. 2022-05-22]. Dostupné z: https://www.youtube.com/watch?v=4olO9UjRiww

[6] *C-Sharp* [online]. Corner [cit. 2022-05-22]. Dostupné z: https://www.c-sharpcorner.com

[7] *Zdrojak* [online]. [cit. 2022-05-22]. Dostupné z: https://zdrojak.cz/clanky/uvod-dohttps://zdrojak.cz/clanky/uvod-do-architektury-mvc/architektury-mvc/

[8] *Geeksforgeeks* [online]. [cit. 2022-05-22]. Dostupné z: https://www.geeksforgeeks.org/benefit-of-using-mvc/

[9] *Itnetwork* [online]. [cit. 2022-05-22]. Dostupné z: https://www.itnetwork.cz/csharp/asp-net-mvc/single-page-application/tutorial-uvod-do-asp-nethttps://www.itnetwork.cz/csharp/asp-net-mvc/single-page-application/tutorial-uvod-do-asp-net-single-page-applicationsingle-page-application

[10] *Pragimtech* [online]. [cit. 2022-05-22]. Dostupné z: https://www.pragimtech.com/blog/blazor/what-is-blazor

[11] *Skeleton* [online]. [cit. 2022-05-22]. Dostupné z: https://www.skeleton.cz/signalr

[12] *Oracle* [online]. [cit. 2022-05-22]. Dostupné z: https://www.oracle.com/cz/index.html

[13] *Nuget* [online]. [cit. 2022-05-22]. Dostupné z: https://www.nuget.org/packages/DocumentFormat.OpenXml

[14] Co je to XML, k čemu se užívá a jaké jsou jeho hlavní výhody? [online]. [cit. 08.08.2022]. Dostupné z: https://www.fastcentrik.cz/blog/co-je-to-xml,-k-cemu-se-uziva-a-jake-jsou-jeho-vyh

[15] Co je to mind mapping. [online] [cit. 08.08.2022]. Dostupné z http://www.ucimse.cz/profesni-vzdelavani/co-je-mind-mapping.html

[16] .Net Standard – CodeDigest.Com [online]. Copyright © 2017 CodeDigest.Com [cit. 10.08.2022]. Dostupné z: http://www.codedigest.com/quick-start/9/what-is-netstandard

[17] Common Language Runtime | Core C# and .NET. Best books online library [online]. Copyright © 2008 [cit. 10.08.2022]. Dostupné z: https://flylib.com/books/en/4.253.1.13/1/

[18] MVC – Acervo Lima. [online]. Copyright © 2022 Acervo Lima, Certains droits réservés. [cit. 10.08.2022]. Dostupné z: https://fr.acervolima.com/avantage-d-utiliser-mvc/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface.
HTML	Hypertext Markup Language.
PHP	Hypertext Preprocessor.
NGWS	Next Generation Web Services.
CLR	Common Language Runtime.
WPF	Windows Presentation Foundation.
WCF	Windows Communication Foundation.
WWF	Windows Workflow Foundation.
MEF	Managed Extensibility Framework.
CCLR	Core Common Language Runtime.
CLI	Common Language Infrastructure.
CTS	Common Type System.
CLS	Common Language Specification.
JIT	Just In Time Compiler.
ASP	Active Server Pages.
AST	Abstract syntax tree.
SPA	Single Page Application.
IR	Intermediate Representation.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

CD se zdrojovými kódy