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

Marek Kantor

Bakalářská práce
2021

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 the Production Process Monitoring Application for on Semiconductor Czech Republic, s.r.o.**

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

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

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

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

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

Ve Zlíně dne 24. ledna 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 se autor bude věnovat dané aplikaci. Zpracuje studii proveditelnosti, analýzu a popíše řešení. Cílem práce bylo vytvořit funkční prototyp této aplikace.

Klíčová slova: .NET, ASP.NET

ABSTRACT

This bachelor thesis deals with the extension of the application for monitoring the production process. The theoretical part focuses on .NET, ASP.NET technologies, how they work and the way they evolved. In practice part, the author will focus on the application. It will carry out a feasibility study, analysis and describe the prototype of 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

Tohle téma jsem si autor vybral, protože ho studoval již na střední škole Informatiky, elektrotechniky a řemesel v Rožnově pod Radhoštěm, konkrétně se jmenoval Informační technologie, kde se naučil spoustu užitečných základů pro vývoj webových stránek. A když se ukázala možnost psát bakalářskou práci pro firmu Onsemi v Rožnově pod Radhoštěm, autor ani chvíli neváhal a hned příležitost chytl za pačesy.

Tato bakalářská práce je založena na úpravě webové aplikace MVC, konkrétně se bude jednat a aplikaci pro monitorování procesu výroby. Kde si firma přeje možnost ručního přidávání hodnot. Aplikace se jmenuje Logbook a načítá data ze systému Promis, který je bere přímo ze zařízení. Tato práce se věnuje technologiím firmy Microsoft, které byly v práci použity. Konkrétně se jedná o .NET a ASP.NET v práci autor vysvětlí i mnoho dalších pojmů, které s těmito technologiemi bezprostředně souvisí.

V Praktické práci autor poté vytvoří podrobnou studii proveditelnosti a Analýzu. Popíše stávající aplikaci podrobně po jednotlivých třídách navrhne databázi a postup řešení, který detailně rozebere.

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 Stackoverflow survey 2021 se na prvním místě jako nejpoužívanější framework umístil React.js. V průzkumu hlasovali jak profesionální programátoři, tak lidé, kteří nejsou profesionálové. Autor 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. Stará se pouze o 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 píší JavaScript. [1]

Třetí místo si vybojoval google se svým produktem jménem Angular. Jedná se o bezplatný multiplatfromní 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 o němž se ještě autor hodně zmiňuje. Tento framework se umístil těsně na pátem místě před Veu.js. Na místě šestém se ale umístil starší brácha tohoto frameworku a to ASP.NET, kdybych je brali jako jeden framework, protože jsou si hodně podobné tak by se umístili na druhém místě. ASP.NET autor popisuje v kapitole 1.5

Na zbylých místech se umístili jak frameworky javasriptové, tak frameworky servrového jazyka PHP, jedná se konkrétně o Spring, Flask, Django, Angular.js, Laravel, Ruby on Rails ,Gatsby, Symfony, FastAPI, Svelte a Drupal. [2]

2 TECHNOLOGIE .NET

V této kapitole se autor bude snažit přiblížit historii vývoje a funkce většiny technologií .NET.

2.1 Microsoft .NET

.NET je bezplatná, multiplatformní, open source vývojářská platforma pro vytváření mnoha různých typů aplikací. S .NET může používat více jazyků, editorů a knihoven k vytváření webových, mobilních, desktopových aplikací, her a IoT. [3]

2.1.1 Microsoft .Net framework

Koncem 20. století firma Microsoft začala pracovat na projektu nazvaný NGWS (Next Generation Web Services). Cílem bylo sjednotit všechny produkty microsoftu a přidat koncovku .NET. To se však všechno nepovedlo podle plánu a Microsoft se rozhodl z NGWS udělat .NET jak ho dnes známe. Projekt přejmenovali a za pár let vyšel první .NET framework 1.0

První beta verze .NET frameworku byly vydána na konci roku 2000 a 13. února 2002 byla vydána první verze .NET 1.0. Jeho hlavní funkcí bylo CLR(Common Language Runtime) viz níže oddíl 1.5.A objektově orientovaný vývoj webových aplikací ASP.NET a desktopových aplikací WinForms.

.NET 2.0 v roce 2005 nám tato verze přinesla spostu nových funcí pro ASP.NET, ale mimo jiné i generické kolekce, iterátory a nullable typy.

O rok později však vyšla nový verze a to .NET 3.0 která obsahovala WPF(Windows Presentation Foundation), WCF (Windows Communication Foundation) a WWF(Windows Workflow Foundation). WPF je architektura uživatelského rozhraní na vývoj aplikácí pro stolní počítače.S pomocí WCF můžeme odesílat data asynchroně a to na koncové body hostované službou ISS nebo službou v aplikaci.

Další verze .NET byla 3.5 v roce 207, která obsahovala funkce na podporu jazyku AJAX, LINQ nebo také ASP.NET MVC viz 1.5. 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 obashovaly třeba MEF(Managed Extensibility Framework), což je framework pro rozšiřovaní projektu bez nutnosti konfigurace, vývojáři se díky němu lehce vyhnou pevným závislostem. Také zlepšili výkon, ladění a vytovřili Razor view engine. [3][4]

2.1.2 Microsoft .NET Core

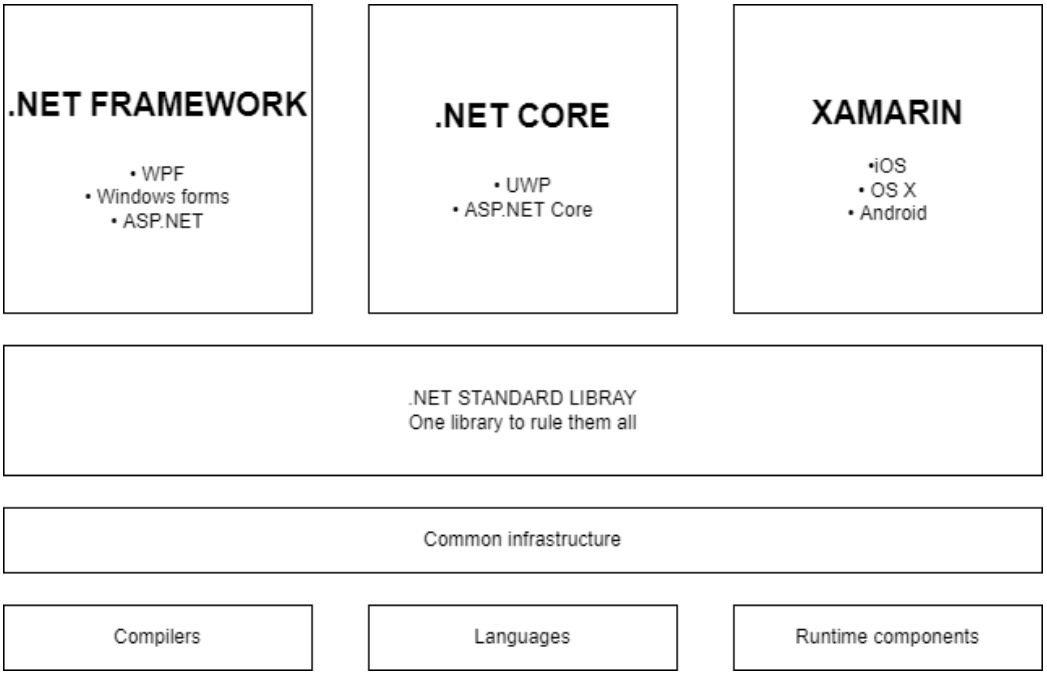
Jelikož .NET Framewok byl čím dál více pomalejší v roce 2014 Microsoft oznámil vývoj systému Microsoft .NET Core. Microsoft předělal vrstvu pod programovacími jazyky a to CLR, nově se jmenovalo CCLR (Core Common Language Runtime) přestali podporovat spoustu starých funkcí, které už nebyly potřebné. Celý systém vyšel v roce 2016 a neskutečným způsobem zrychlil celý systém 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 neznámé, že třeba na Linuxu poběží Windows forms, ty jsou totiž přímo jiné pracují s windowsem. [5][3]

2.1.3 Microsoft .Net

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

2.1.4 Microsoft .NET Standard

Je tokový interface pro knihovny základních tříd jednotlivých platforem .Net. Znamená to že .NetFramework, .Net Core, Xamarin, Silverlight, Unity atd. mohou sdílet jednotlivé knihovny.[http://www.codedigest.com/quick-start/9/what-is-netstandard]



Obrázek 1 .NET Standart

2.1.5 Jazyky

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

Standart se jmenuje CLI (Common Language Infrasturcute). Specifikuje a definuje prostředí, jenž umožňuje používání více vysokoúrovňových programovacích jazyků. Aniž by bylo nutné přepisovat jejich překladače. Obsahuje sadu CTS(Common Type System) jedná se o sadu, která definuje jaké typy jsou deklarovány, používány a spravovány v CLR viz níže 1.5. Systém poskytuje objektově orientovaný model, ten .NET používá v několika jazycích. Další sada co patří do podmnožiny sady CTS se jmenuje CLS(Common Language

Specification) 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.

2.1.5.1 C#

Je jednoduchý, moderní, objektově orientovaný a bezpečný programovací jazyk. Autor jazyk v práci používá a pracuje sním.

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

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

2.1.5.2 F#

je programovací jazyk, který usnadňuje psaní stručného, robustního a výkonného kódu.

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

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

2.1.5.3 Visual Basic

Je jazyk s jednoduchou syntaxí pro vytváření bezpečných, objektově orientovaných aplikací.

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

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

[3]

2.1.6 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. NuGet balíčky.

Jedná se o soubor s příponou .nupkg, který obsahuje zkompilovaný kód ddl, 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 mohou být privátní. Autor v projektu používá jak Nugety veřejné, tak Nugety privátní, firemní. Které definují a standardizují třeba přihlašovací systém.[3]

2.1.7 Common Language Runtime

CLR je nedílnou součástí .NET jedná se o systém, jež je společný pro všechny jazyky .Net. Každý jazyk vygeneruje metada data, které mají stejný formát, tyto medata poté zpracovává tento modul. Používá metadata k vyhledání a načtení tříd, rozmístění instancí v paměti, řešení vyvolání metod.

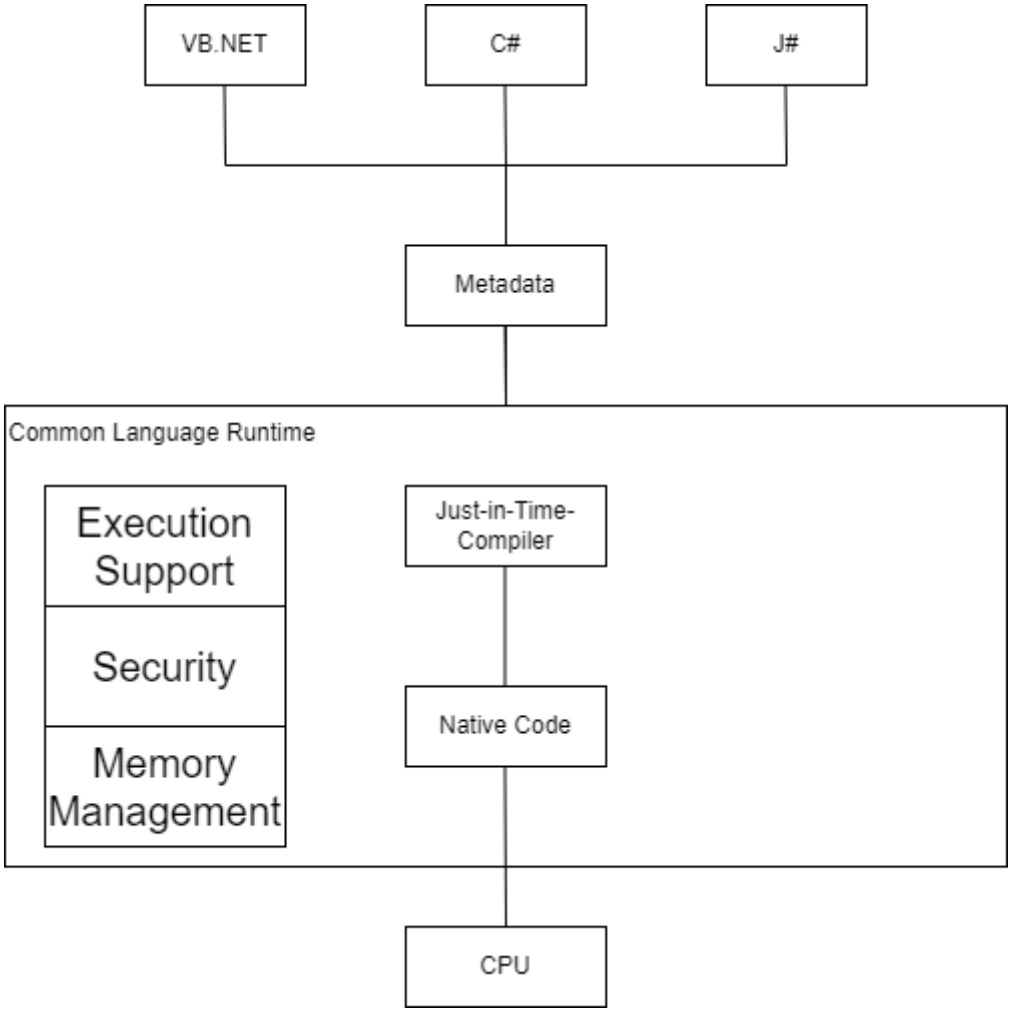
Díky tomuto modulu je možné také třeba definovat třídu a potom použít jiný jazyk abychom metodu zavolali. Můžeme předat instanci třídy metodě třídy napsané v jiném jazyce.

“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 take compiler JIT (Just In Time Compiler) jenž se stará o převod daných metadata do strojového kódu což je posloupnost strojových instrukcí pro procesor počítače. [3][6]

[https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/]



Obrázek 2 - Common Language Runtime

2.2 ASP.NET

ASP.NET je bezplatná webová technologie pro vytváření webů a webových aplikací pomocí HTML, CSS a JavaScriptu. Můžete také vytvářet webová rozhraní API a používat technologie v reálném čase, jako jsou webové sokety. Technologie je multiplatfromní. Poprvé byla vydána v roce 2002 v .NET frameworku. ASP.NET se stal nástupcem technologie ASP(Active Server Pages), která byla taky vyvíjená společností Microsoft nová verze byla daleko rychlejší a ve všech ohledech lepší. Díky velikému množství knihoven a enormímu výběru ovladacích prvků se vývoj aplikací neskutečně zrychlil. Schopnost využití paměti Cache nám uvolní 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 ale zastaralá a už se moc nepoužívá daleko více se používá architektura MVC, kterou autor popisuje o kapitolu níže.

2.2.2 ASP.NET Core MVC

Mnoho dnešních aplikací je napsáno právě v MVC, dříve se hodně používala technologie Silverlight. Microsoft ji ale čím dál tím míň podporoval až ke konci minulého roku s 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 příjemné. 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 autor postupně po jednom rozebere dané komponenty. [3]

2.2.2.1 Model

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

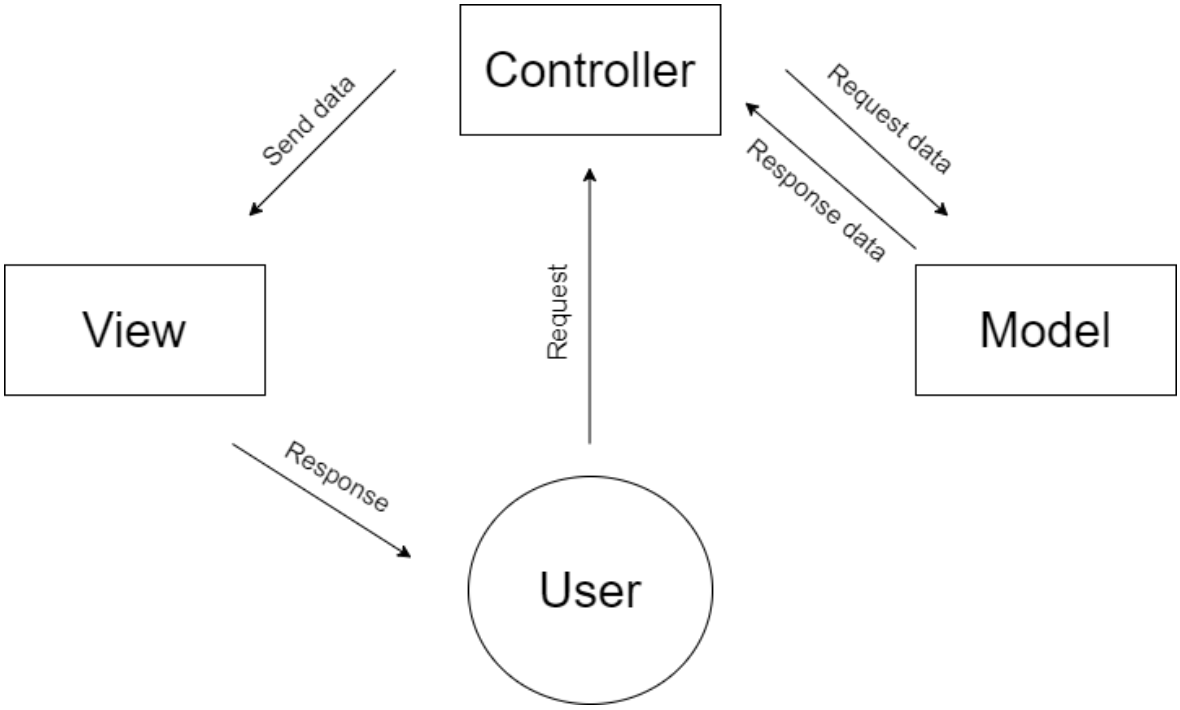
2.2.2.2 View

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.

2.2.2.3 Controller

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



Obrázek 3 - MVC

2.2.3 Blazor

Pro serverovou část se často používají programovací jazyky jako je C#, Java,Python a ostatní, pro klientskou část se používá JavaScript nebo jeho frameworky jako je Angular, Rect 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. [10]

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

2.2.4 Single Page Aplication

Jedná se o technologii, která navazuje na MVC. Klasické webové aplikace jsou určitě žalovatelnější. Ale co když je potřeba aplikaci mít rychlou a menší, s malou zátěží na serveru? Využijeme SPA. Tato technologie funguje tak že 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.

Server je vypadá jako technologie ASP.NET MVC. Web Api pro komunikaci Javascriptu jsou v našem případě kontrolery. Na server se tedy 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

Aplikace je převážně napsaná v ASP.NET MVC, tato technologie však sama o sobě nestačí, proto se používají i ostatní technologie. Těmato technologiema se autor bude zabývat právě v této kapitole.

3.1 ORACLE

Jako systém řízení báze dat autor používá v projektu Oracle, jedná se o multiplafrmní databázový systém, s možnostní pokročilého zpracování dat a vysokým výkonem. Systém podporuje standartní relační dotazovací jazyk SQL, ale také rozšíření vytvořené přímo firmou Oracle. Pro práci s oraclem autor využívá Oracle SQL Developer. Jedná se o grafické prostředí díky, kterému nemusí autor všechny jednotlivé příkazy psát pomocí SQL.[12]

3.2 HTML

HyperText Markup Language je zádkladním stavebním kamenem webovych stránek. Jedná se o značkovací jazyk, značky nebo tagy tvoří celou strukturu webové stránky, HyperText v názvu stojí za tzv. hepertextovými odkazy, které vzájemně propojují jednotlivé stránky na webu. HTML je uplný vrchol ledovce 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 vzniky aby programátorům zjednodušili programování webových stránek. Pokud bychom museli i dnes zapisovat všechno v HTML asi by se nám z toho udělalo nevolno. 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

Autor v předchozích kapitolách hodně zmiňuji 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 pracujicí s Office word, excel a powerpoint. Zvádá všechny typy operací, zakladní třeba jako zápis nebo čtení z jednotlivých souborů. Nebo třeba slučování souborů, vyhledevání pomocí regulárních výrazů nebo jiné operace přímo v microsoft dokumentu.[14]

II. PRAKTICKÁ ČÁST

4 STUDIE PROVEDITELNOSTI

Když autorovi bylo řečeno, že taky ve firmě onsemi kde dělá dlouhodobého stážistu, je možnost napsat si bakalářskou práci, ani chvíli neváhal a šel do toho. Začalo to první informativní schůzkou, kde bylo řečeno o co se jedná. Po sérii schůzek, kde se autor s pomocí kolegů seniorů ve firmě se snažil získat co nejvíce informací ohledně projektu a požadavků. Zjistil o co se jedná, a také porozuměl celé funkcionalitě této aplikace.

Aplikace LogBook je primárně určena pro sledování a plánování výroby křemíkových data. 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 investor přeje přidat editaci těchto přidaných záznamů, aby byla možnost případě nějaké chyby záznam upravit.

4.2 Sběr požadavků

4.2.1 Funkční požadavky

- Přidání ručního záznamu
- Editace přidaného záznamu
- Autorizace při přidávání i editaci
- Auditování změn
- Dynamická validace zadaných dat
- Propojení se systémem Space (nice to have)

4.2.2 Nefunkční požadavky

- Vedení auditních záznamů
- Nápověda

4.2.3 Nebude se realizovat

- Uživatelská a programátorská dokumentace

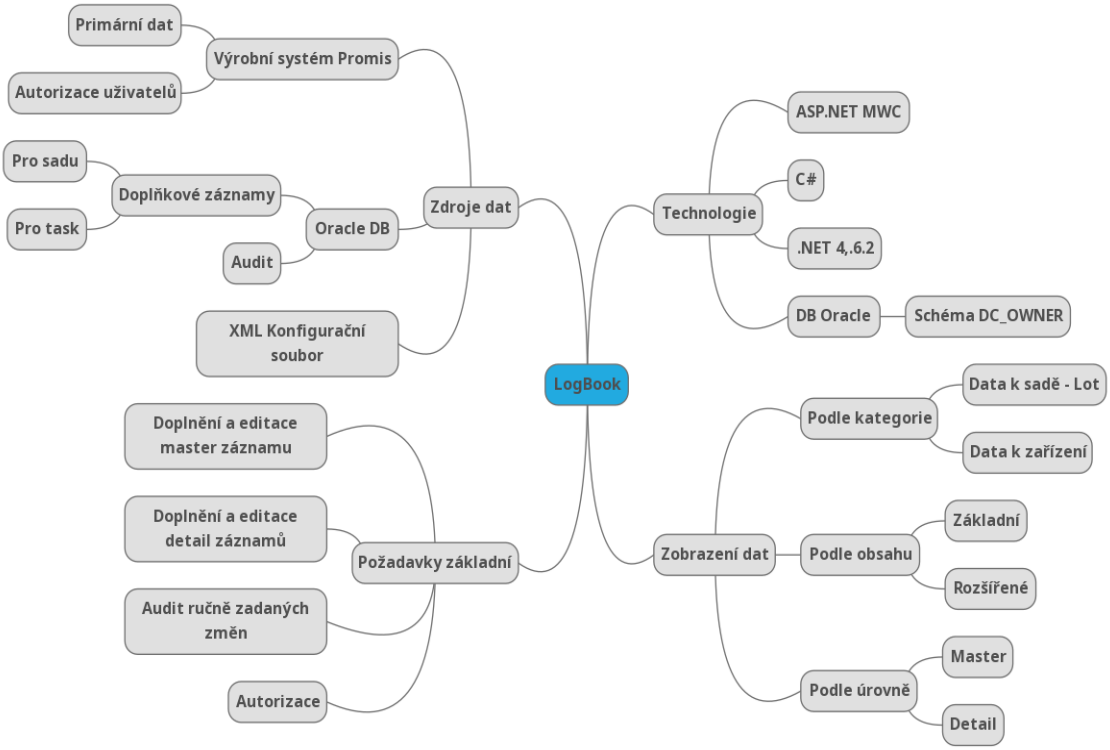
4.3 Harmonogram projektu

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.[15]



Obrázek 4 – Myšlenková mapa

4.5 Závěrečné shrnutí studie

Po důkladné studii a konzultacích s kolegy autor zjistil, že realizace toho projektu je reálná.

5 ANALÝZA

5.1 Stávající aplikace

5.1.1 GUI aplikace

Aplikace je napsaná v jazyku c# a jedná se o webovou aplikaci typu MVC. Aplikace umí základní filtraci pomocí ReaktorID jak můžeme vidět v levém okně, dále fitrruje rozsah, deafultně je nastaven na poslední dva dny. V check boxech mužeme zatrhnout zda chceme zobrazit tasky, sady nebo rozšířený režim.

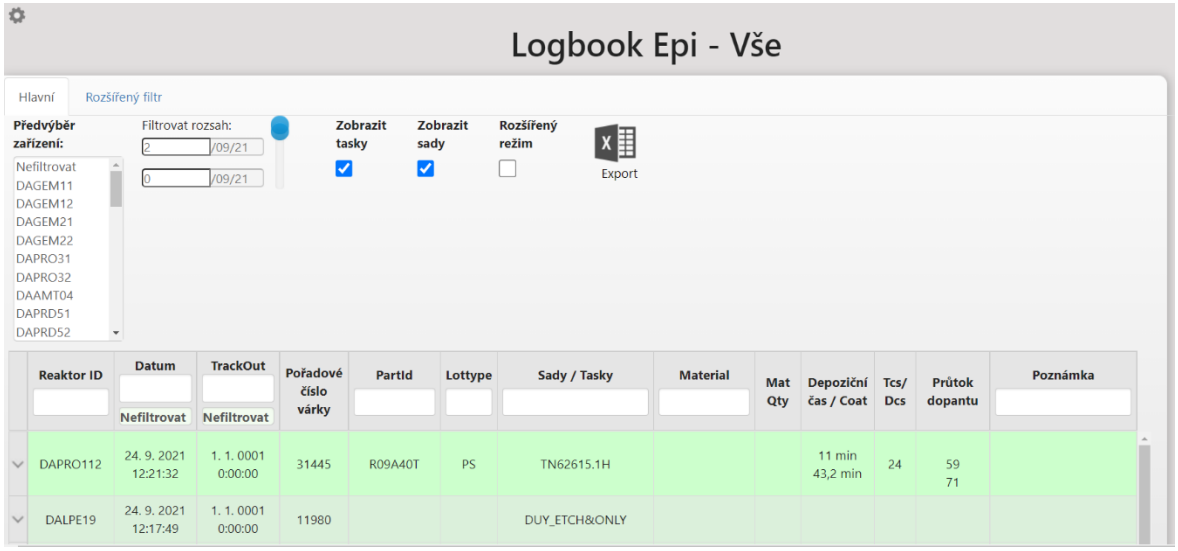
- Tasky : zobrazí tasky
- Sady : zobrazí sady
- Rozšířený režim : zobrazí i defaultně skryté data

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

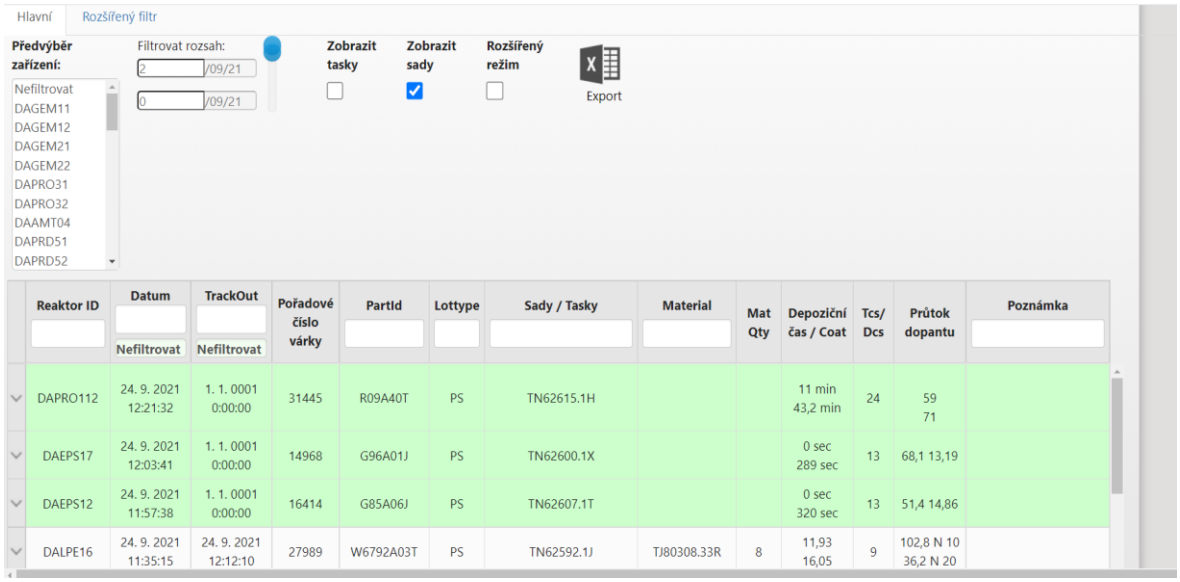
Zde v tomto prostoru, v této navigaci se bude nacházet i autorem nově vytvořené tlačítko pro vytvoření nového záznamu.

V rozšířeném režimu můžeme rovnou vyhledat data pomocí specifických parametrů.

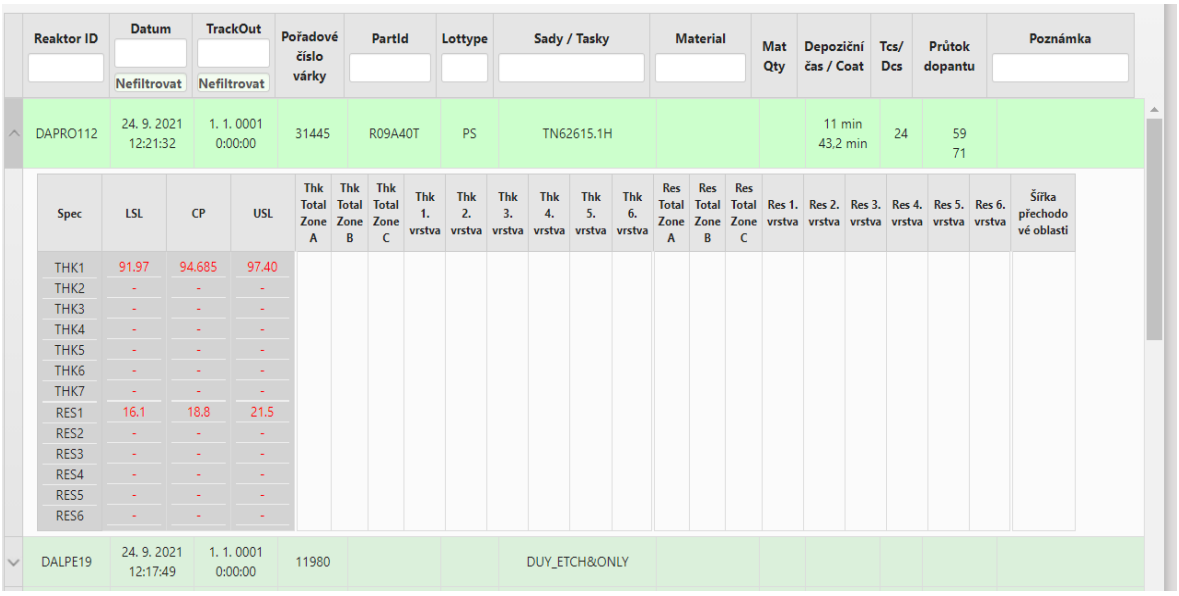
Pod navigací se nachází samotná tabulka s daty. Po kliknutí na řádek, rozjedeme detail tabulky. Kde se nachazejí konkrétnější data.



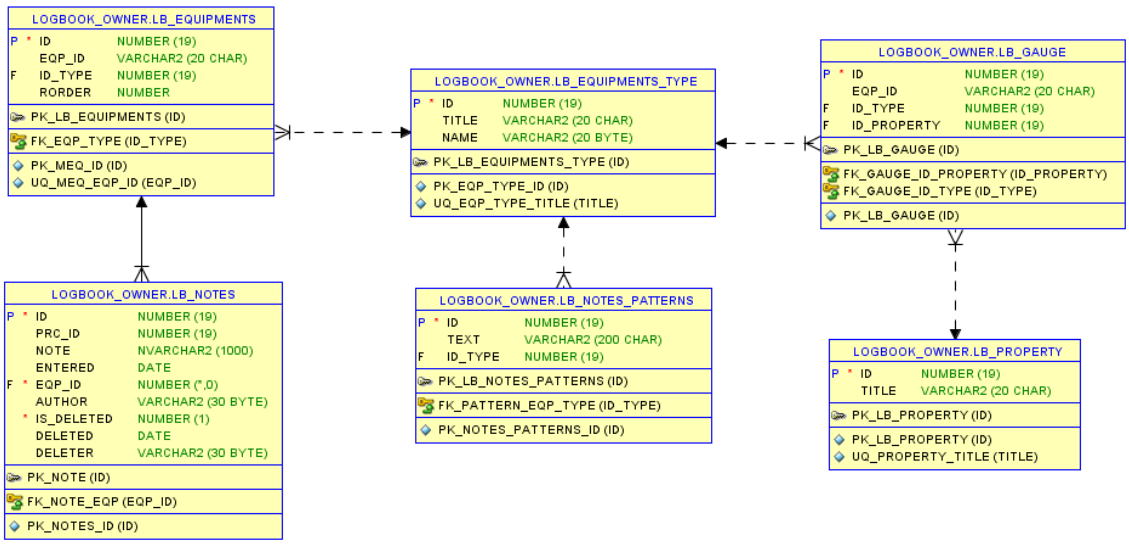
Na obrázku číslo X vidíme jak aplikace vypadá po zapnutí Logbooku Epi.



Obrázek 6 - Základní pohled původní aplikace



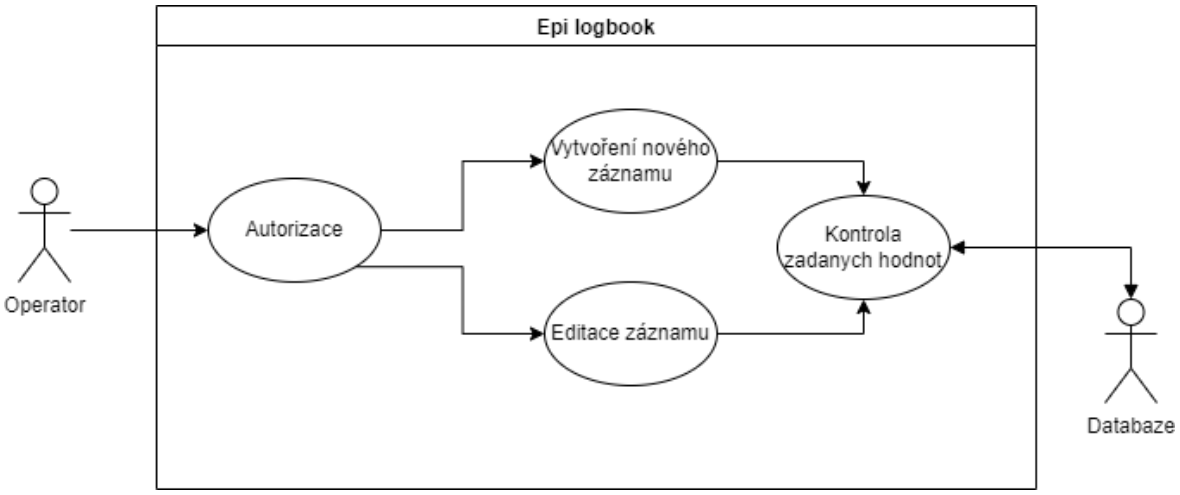
Obrázek 5 - Detail původní aplikace



Obrázek 7 - Původní datový model s hodnotami

Původní diagram databáze, pro realizaci projektu bude nutné přidat více tabulek. Data se nenacházejí pouze v těchto tabulkách, ale berou se i z promisu.

5.1.2 Uses case



Obrázek 8 - Uses case diagram

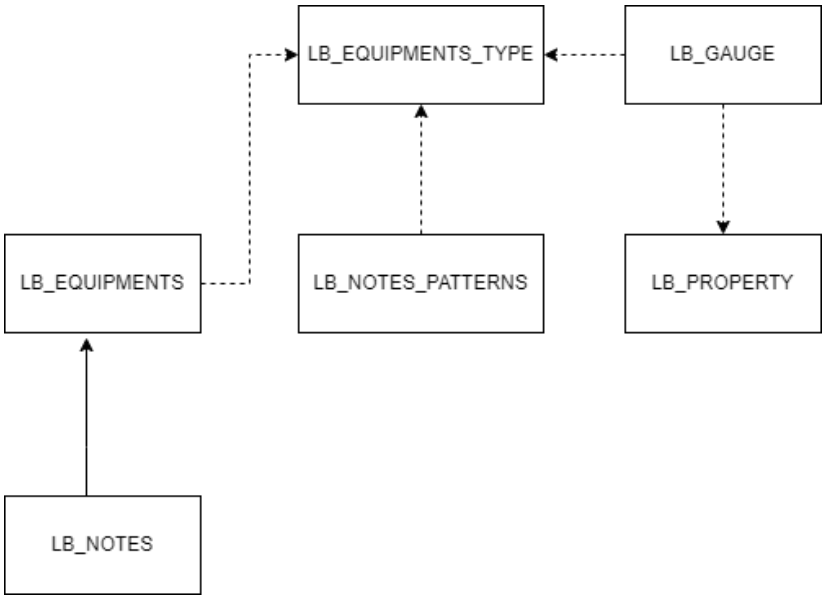
Uses case diagram pro vytvoreni noveho zaznamu a editaci.

5.1.3 Datový model

Většina dat se bere z promisu, což je databáze torrent, s touto databázi ale autor pracovat nebude, autor bude upravovat databázi CZ4AUT, ve které se nachází poznámky, nově zde budou uložena i data pro ruční přidávání inženýrských záznamů.

5.1.3.1 CZ4AUT

Databáze, která je nutná z důvodu přidávání poznámek k jednotlivým sadám nebo taskům.



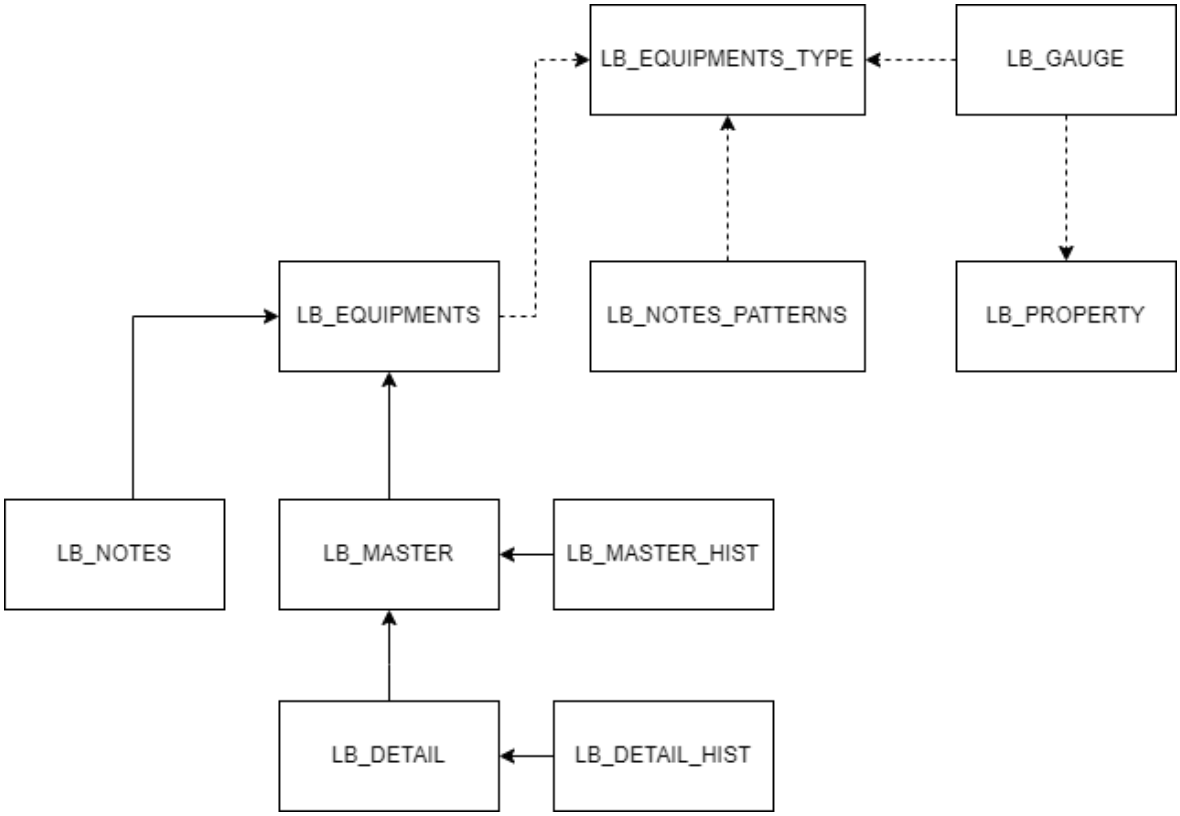
Obrázek 9 - Původní datový model

5.1.3.2 TORRENT

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

5.2 Idealní datový model

Ideální datový model je model, který by byl ideální pro řešení tohoto projektu. V modelu jsou přidané 4 tabulky.



Obrázek 10 - Idealní datový model

5.2.1 LB_MASTER

Tabulka se základními daty. Vazba na LB_EQUIPMENTS.

5.2.2 LB_MASTER_HIST

Tabulka se změnami v tabulce LB_MASTER. Vazba na LB_MASTER.

5.2.3 LB_ DETAIL

Tabulka s detailními daty. Vazba na LB_MASTER.

5.2.4 LB_DETAILHIST

Tabulka se změnami v tabulce LB_DETAIL. Vazba na LB_DETAIL.

5.3 Popis zdrojáků ELogBooku

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, hlavnětedy Torrentu a databáze Logbooku.

Podsložka SQL_Torrent obsahuje SQLka pro dotazování na data do Torrentu. Konkrétně EPI logbookpoužívá dotazy začínající Alias*. Třída která dotazy provádí a vrací data, je DbConnectionTorrent.cs.Podsložka SQL_ElogBook obsahuje SQLka 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 textBoxy 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 k nim podle hlavních kriterií (jako jsou počet dní/interval za který data získat, EquipmentIds pokud jsou zadané), GetTasks pro získání tasků a dat k nim podle hlavních kriterií, 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 dělá pár dalších věcí. 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

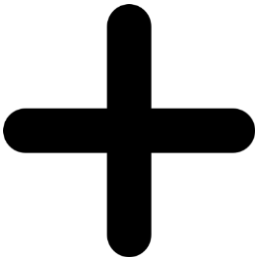
Webová část projektu, používá všechny ostatní výše zmíněné projekty. Je to ASP MVC5 projekt. Složka Controllers obsahuje kontroler ke každému logbooku. Kontrolery obsahují metody, které je možné volat „zvenku“, tzn. z prohlížeče. Obsahuje další složky ke každému logbooku. Ty obsahují Views, tedy pohledy, samotné UI. Složka Scripts obsahuje JavaScripty. Soubory 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, automatický refresh stránky, přepínání mezi taby (Hlavní/Rozšířený filtr), parsování data a ostatní. Složku Content tvoří obrázky a kaskádové styly

6 NÁVRH APLIKACE

Jelikož aplikace je po grafické stránce hotová, není nutné dělat grafický návrh stránky. I když v zadání je, že má autor vytvořit drátové modely rozhraní. Místo toho se rozhodl, že vytvoří návrh tlačítka a jednoduchý grafiky návrh formuláře.

6.1 Návrh tlačítka

Jedná se o obyčejné černé tlačítko znázorňující znak plus. Tlačítko autor umístí podle specifikovaných požadavků investora. Toto tlačítko by nás mělo přesměrovat na přihlašovací formulář. Pokud již nejsme přihlášeni.



Obrázek 11 - Talčítko

6.2 Návrh databáze

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

V databázi, autor vytvoří celkem čtyři nové tabulky, bude se jednat o tabulku LB_Master, kde se budou ukládat data stejné jako jsou v řádku. Hlavní parametry budou mít své sloupce zbytek se bude zapisovat do sloupce s názvem OTHERDATA ve formátu JSON. Tabulka využije již vytvořené tabulky LB_EQUIPMENTS, která obsahuje všechny používané reaktory.

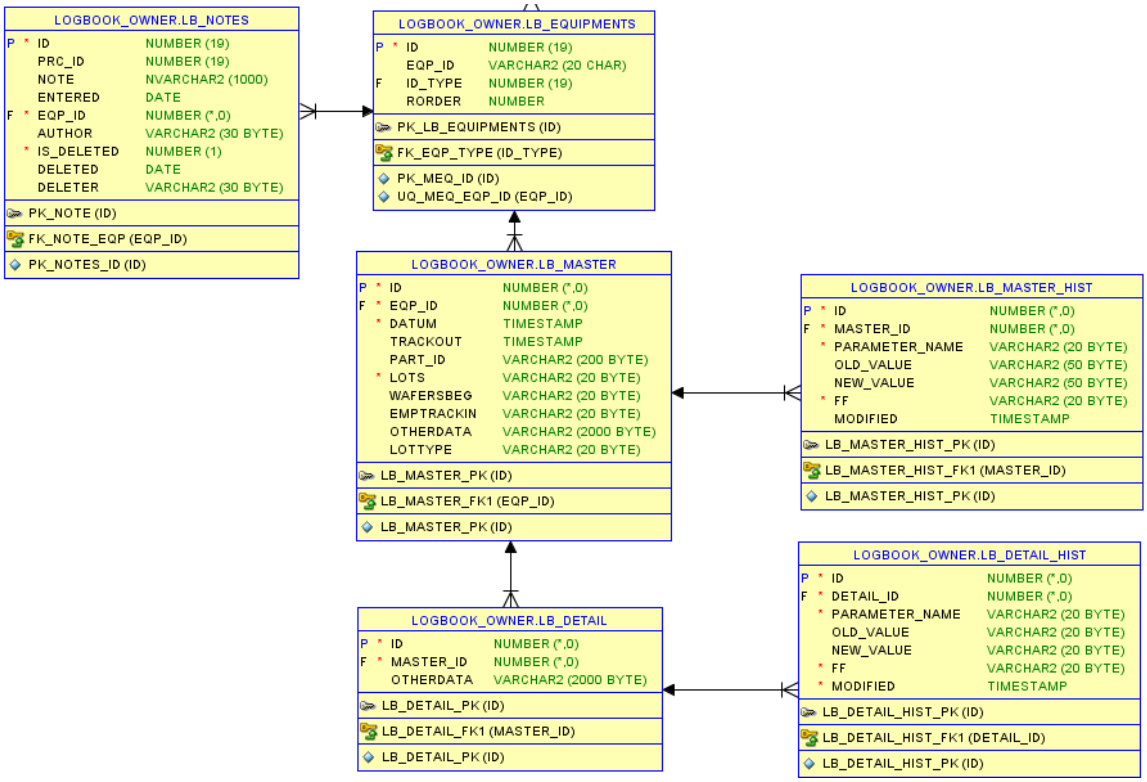
Další tabulka LB_MASTER_HIST bude obsahovat tyto sloupce MASTER_ID, PARAMETR_NAME, OLD_VALUE, NEW_VALUE, FF, MODIFIED.

Sloupec MASTER_ID bude odkazovat na předchozí tabulku. Díky tomuhle parametru, který je cizím klíčem autor zajistí že každý řádek bude mít svou vlastní tabulku historii.

PARAMETR_NAME bude obsahovat jednotlivý název daného parametru. OLD_VALUE a NEW_VALUE bude patřit hodnotě, kterou budeme měnit a hodnotu novou. FF je jméno nebo identifikační číslo uživatele. A sloupec MODIFIED by nám měl znázornit čas kdy k dané změně došlo.

Tabulka LB_DETEAIL, bude 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 12 - Realný databázový model

6.3 Návrh formulářů

Při navrhování formuláře se autor snažil zachovat původní vzhled stránky a tím pádem se držel standardů vytvořených v autorizaci. Formulář bude vytvořen pomocí Frameworku

boostrap. Formulář je typu popup. Popup formulář funguje jako vnitřní stránka, která se otevře při zmáčknutí tlačítka. Tato stránka nás nikam nepřesměruje, tudíž je to rychlejší a také praktičtější. Formulář obsahuje název jednotlivých sloupců a určité textové pole pro vyplnění dat.

7 REALIZACE APLIKACE

Autor zná všechny požadavky. S aplikací je seznámený, navrhl si jak databázi tak formuláře, může se vrhnout do programování. Jako první se rozhodl implementovat dané tlačítko.

7.1 Implementace tlačítka

V layout.cshtml se autor lehce zorientoval a vložil provizorní tlačítko vedle checkboxu, dokonce využil i jejich styl pro div, aby vše zůstalo jednotné.

Jelikož autor potřeboval aby se tlačítko ukazovalo jen když se jedná o Loogbook s názvem Epi, přidal pro to jednoduchou podmínku.

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

MvcApplication.LbEquipmentType načítá data s konfiguračního souboru Web.config, kde je specifikované o jaký logbook se jedná. Jelikož má stránka nastavený automatický refresh, aby se aktualizovaly data z databáze. Medota saveRefresh() již v projektu existovala a zabraňuje tomuto automatickému refreshi.

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

```
$("#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

Autor nastavil tlačítko. Jenže nemá smysl se věnovat tlačítku, pokud by nevykreslovalo to co by mělo. A proto je třeba naprogramovat, jak se vlastně budou data do jednotlivých řádků vyobrazovat.

V projektu Onsemi.Ei.Elogbook.ViewModels existuje třída jménem Getprocess(), tato posílala dotazy na SQL server a nazpět dostala list dat, které se vykreslí v tabulce. Autor vytvořil nové metody, pojmenoval je 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

GetProccesEpi bude vykreslovat řádky, které budou ručně přidávány. Jelikož databáze již byla navrhnuta je možné to naprogramovat, dříve než samotný formulář s přidáváním. Autor pro ukládání vytvořil novou datovou strukturu a to FullBatch, která presentuje databázi a strukturu ProccesRow, jenž využívá aplikace pro vykreslování tabulky. Inspiroval se u metody getProccesTorrent, která obsahuje většinu z původní metody GetProcess(), kde se většina dat ukládala do datové struktury Batch.

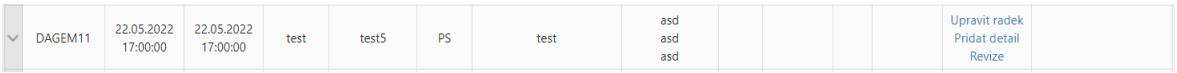
```
public List<ProcessRow> GetProcessEpi()
    {
      try
      {
        var batches = ConnectionEpi.GetData(_daysFrom, _daysTo, _advancedFilterValue,
          _advancedFilterSpec.ColumnName, _advancedFilterSpec.MonthsFrom, _advancedFilter-
Spec.MonthsTo);

        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 podmínky pro se SQL dotaz, jak staré hodnoty se budou hledat, atd. V Oracle dotazu se přepíše daný filtr podle potřeby viz.

conditionString. A dotaz je zavolán. Všechny jednotlivé data jsou zapsány do jednotlivých proměných. Sloupec OTHERDATA je zavolán do stringu, ale musí být zrealizován přímo do další datové struktury FullBatchData. Jenž je volána ve FullBatch. Data jsou zapsána do listu a vrácena.

```
public List<FullBatch> GetData(int daysFrom, int daysTo = 0, string advancedFilterValue = "",
    string advancedFilterColumnName = "", int advFilterMonthsFrom = 0, int advFilterMonthsTo = 0)
    {
       var result = new List<FullBatch>();
        if (!Open())
          {
             return result;
          }

          var cmd = _cmdSelectData;

          string conditionString;
          if (string.IsNullOrEmpty(advancedFilterValue))
          {
             conditionString = $"TRACKOUT > SYSDATE - {daysFrom} and TRACKOUT < SYSDATE - {daysTo}";
          }
          else
          {
             conditionString = String.Empty;
          }

          cmd.CommandText = cmd.CommandText.Replace("&condition", conditionString);
          OracleDataReader reader = cmd.ExecuteReader();

          long lotid = Int64.MinValue;
          long eqpids = Int64.MinValue;
          DateTime trackin = DateTime.MinValue;
          DateTime trackout = DateTime.MinValue;
          string partid = string.Empty;
          string lottype = string.Empty;
          string lots = string.Empty;
          string wafers = string.Empty;
          string emptrackin = string.Empty;
          string data = string.Empty;

          while (reader.Read())
          {
             if (!reader.IsDBNull(0))
             {
                lotid = reader.GetInt64(0);
             }
             if (!reader.IsDBNull(1))
             {
                eqpids = reader.GetInt64(1);
             }
             if (!reader.IsDBNull(2))
             {
                trackin = reader.GetDateTime(2);
             }
             if (!reader.IsDBNull(3))
             {
                trackout = reader.GetDateTime(3);
```

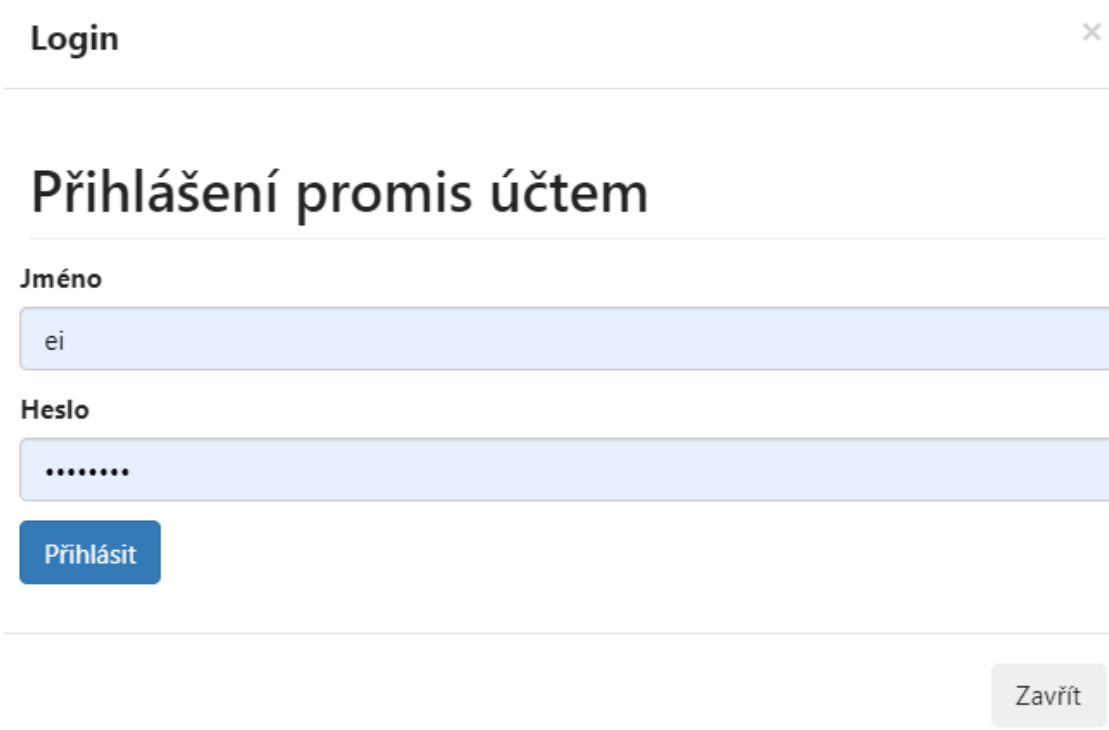
```
            }
            if (!reader.IsDBNull(4))
            {
                partid = reader.GetString(4);
            }
            if (!reader.IsDBNull(5))
            {
                lots = reader.GetString(5);
            }
            if (!reader.IsDBNull(6))
            {
                wafers = reader.GetString(6);
            }
            if (!reader.IsDBNull(7))
            {
                emptrackin = reader.GetString(7);
            }
            if (!reader.IsDBNull(8))
            {
                data = reader.GetString(8);
            }
            if (!reader.IsDBNull(9))
            {
                lottype = reader.GetString(9);
            }

            var desData = JsonConvert.DeserializeObject<FullBatchData>(data);

           result.Add(new FullBatch
            {
                Id = lotid,
                EqpId = IntToEqp(Convert.ToString(eqpids)),
                CreateDate = trackin,
                TrackOut = trackout,
                PartId = partid,
                Lottype = lottype,
                Lots = lots,
                Wafers = wafers,
                EmpTrackIn = emptrackin,
                Data = desData
            });
        }
        reader.Close();
```



Obrázek 13- Řádek s funkcemi

7.3 Autorizace

Po kliknutí se nám zavolá metoda Login. Tato metoda má dva parametry, prvním je název metody v kontroléru, do které chceme být přesměrování po dokončení přihlášení a elemet this, který je schopný přenášet třeba jednotlivé řádky tabulky, budeme ho potřebovat, když budeme chtít data upravovat.



Obrázek 14 - Přihlašovací formulář

Pokud bylo přihlášení uspěšné, zavolá se Event loginFinishEventHandler(data), metoda má jeden argument a to jsou data, v těchto datech vidí jestli přihlášení proběhlo v pořádku a nebo ne, pokud ne vyskočí chybová hláška, pokud se data.status rovná ok, voláme jednotlivé medoty. Podle toho, které tlačítko jsme zmáčkli, podle toho vybíráme další pořád javascriptovou metodu, opět si přeneseme potřebná data, a to je jméno uživatele, metody a id.

```
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');
  }
}
```

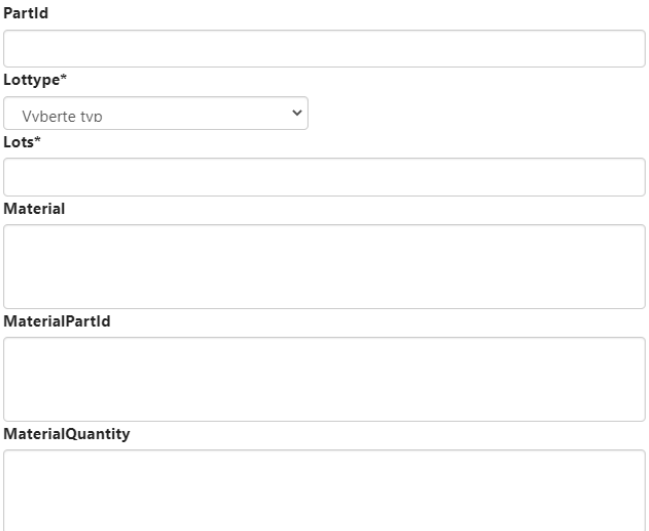
Metoda pomocí Ajaxu přenese data do kontroléru.

7.3.1 Vykreslení dat do formuláře

Metoda AddRowDialog v Epi Controlleru s data přenesenými pomocí ajaxu. Jenom kontroluje zda přenášíme Id, kdyby ano, jednalo by se o úpravu, né 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 autor přikládá formulář. Formulář se skládá se dropdownů, textových polí a textový areí. Textové areye jsou použity abychom mohli do jednoho řádků vložit více hodnot, jednotlivé hodnoty



Obrázek 15- Formulář pro přidávání dat

oddělujeme entrem aby to bylo uživatelsky přívětivé. Názvy označené hvězdičkou jsou povinné.

7.4 Ukládání řádku

Nejprve tyto data musí být převedena takže všechny data z ProccesRow vložíme do Batche. Dále kontrolér kontroluje jestli znovu nepřenáší id. Jestliže ne, zavolá se metoda InsertRowData(), ještě před ní, ale musí být upraveny jednotlivé data. Jedná se o sloupce, které mohou obsahovat vícero hodnot, ty jsou poslány do metod, aby 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(Fullbatch), využívá vytvořeného sql dotazu, který binduje jednotlivé 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 řádků 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 nerefreshuje.

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 se rovná true, tak se do dalšího sloupce jménem Funkce vloží tři hypertextové odkazy, jedná se odkazy na dané metody viz. Obrázek 15.

7.5.1 Úprava řádek

Zavolá se stejná funkce jako pro přidávání řádků. Tentokrát se do ní zavolá řádek s id. Formulář se vykreslí už předvyplněný a uživatel je schopný provádět změny. V metodě se kontroluje zda id existuje, pokud ano jedná se o úpravu řádku. Data se zpracují do správného formátu a zavolají se data původní. Tyto data se porovnají. Pomocí metody DetailedCompare().

7.5.2 Přidání, vykreslení a úprava detail

Pro přidání detailu existuje podobný formulář jako pro Master, jen z jinými hodnotami. Vykreslení detail je trochu 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 zapíší se změny do revize.

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				4																			Upravit detail
THK2				2																			
THK3				3																			
THK4				4																			
THK5				Ø3,25																			
THK6				33,333%																			
THK7																							
RES1																							
RES2																							
RES3																							
RES4																							
RES5																							
RES6																							

7.5.3 Revize

Existují dvě tabulky pro zapisování změn, ale jen tyto data jsou vypisovány jen jedním způsobem a to společně, Metoda vytáhne všechny data vazající se k danému řádku a vypíše je. Zapány jsou do obyčejné tabulky vytvořené v boostrapu.

#	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
5	EqpId	47	45	EI Account	22.05.2022 17:42:57
6	CreateDate	22.05.2022 9:00:00	22.05.2022 17:00:00	EI Account	22.05.2022 17:42:57
7	TrackOut	22.05.2022 9:00:00	22.05.2022 17:00:00	EI Account	22.05.2022 17:42:57

Obrázek 16 - Revize

ZÁVĚR

Autorovi se podařilo vytvořit funkční protop aplikace, a dokázal splnit všechny požadavky, které mu byly zadány. Aplikace sama o sobě byla řešena poněkud nešťastně, mnohdy měl autor nějaký návrh řešení, který musel později měnit kvůli funkčnosti ostatních logbooků.

Práce s daty byla příjemná, dokonce i zábavná, to se ale nedá tvrdit o Javascriptu a AJAXU, který mnohokrát dělal autorovi zbytečné problémy. Možná až zbytečně moc autor upřednostňoval programování před textovou prací. V textové práci potom nestíhal a k obhajobě by mu stačila aplikace nedodělaná. Aplikace, ale byla dokončena a momentálně ji testují zadavatelé práce přímo ve firmě. A vypadá to, že budou s prací spokojeni. S firmou je autor domluvený na možné další spolupráci na projektu Logbook, ale i na jinačích projektech.

SEZNAM POUŽITÉ LITERATURY

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

[2] *Stackoverflow* [online]. Stack Exchange [cit. 2022-05-22]. Dostupné z: https://stackoverflow.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-do-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-net-single-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] https://www.fastcentrik.cz/blog/co-je-to-xml,-k-cemu-se-uziva-a-jake-jsou-jehovyh

[15] http://www.ucimse.cz/profesni-vzdelavani/co-je-mind-mapping.html

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA I: CD

Cd obsahuje zdrojové kódy programu.