VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

# AUTOMATICKÉ ZJIŠŤOVÁNÍ ROZMĚROVÝCH ZMĚN BETONOVÝCH SMĚSÍ PŘI TUHNUTÍ

AUTOMATIC DETERMINATION OF DIMENSIONAL CHANGES OF CONCRETE MIXTURES DURING SOLIDIFICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Martin Hortvík

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Pavel Heriban, Ph.D.

BRNO 2017

# Zadání diplomové práce

| | |
|---|---|
| Ústav: | Ústav automatizace a informatiky |
| Student: | **Bc. Martin Hortvík** |
| Studijní program: | Strojní inženýrství |
| Studijní obor: | Aplikovaná informatika a řízení |
| Vedoucí práce: | **Ing. Pavel Heriban, Ph.D.** |
| Akademický rok: | 2016/17 |

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

## Automatické zjišťování rozměrových změn betonových směsí při tuhnutí

**Stručná charakteristika problematiky úkolu:**

Experimentální zjišťování rozměrových změn betonových směsí při tuhnutí je časově velmi náročné. Vytvářená aplikace bude tato měření provádět automaticky. Desktopová aplikace načte obrazová data z mikroskopů a provede jejich zpracování a vyhodnocení. Získané výsledky budou automaticky ukládány do databáze na server. Serverová aplikace umožní vyhodnocení získaných dat. Umožní také získání dat aktuálního měření z experimentální stanice.

**Cíle diplomové práce:**

1) Seznámení se s moderními technologiemi používanými při tvorbě aplikací pro operační systém Windows.
2) Analýza a návrh desktopové a serverové aplikace pro automatizaci experimentálního zjišťování rozměrových změn betonových směsí při tuhnutí.
3) Implementace obou částí aplikace.
4) Ověření funkčnosti aplikace v reálném nasazení.

**Seznam doporučené literatury:**

SHARP, John. Microsoft Visual C# 2010: krok za krokem. Brno: Computer Press, 2010. Krok za krokem (Computer Press). ISBN 978-80-251-3147-3.

DYKSTRA, Tom. Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application. ASP.NET. [online]. 30.7.2013 [cit. 2016-10-31]. Dostupné z: https://www.asp.net/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application.

AForge.NET Framework. AForge.NET. [online]. [cit. 2016-10-31]. Dostupné z:
http://www.aforgenet.com/framework/

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

## ABSTRAKT

Tato diplomová práce se zabývá návrhem a realizací desktopové aplikace určené ke zjišťování rozměrových změn betonových směsí při tuhnutí. Součástí práce je implementace serverového portálu pro vyhodnocování získaných dat v reálném čase. Úvodní část práce se věnuje analýze úlohy a původního řešení, dále pak vymezením specifikací nové aplikace při nasazení v reálném provozu. V další části práce jsou popsány současné softwarové technologie používané při vývoji desktopových a serverových aplikací. Návrh a samotná implementace obou částí řešení tvoří podstatnou část práce. V poslední části je prezentováno ověření funkčnosti aplikace nasazené v reálném provozu.

## ABSTRACT

This diploma thesis deals with the design and implementation of a desktop application designed to determine the dimensional changes of concrete mixtures during solidification. The work includes implementation of the server portal for real-time data evaluation. The introductory part deals with the analysis of the measurement task and the original solution, as well as the specification of the new application and its deployment in real operation. The next part of the thesis describes current software technologies used in the development of desktop and server applications. The design and implementation of both parts of the solution is a major part of the work. In the last part, the real-time functionality test of the application is presented.



## KLÍČOVÁ SLOVA

Web API, MSSQL, Entity Framework, software, C#, MVC, WPF, XAML, desktopová aplikace, AJAX, http klient, webová služba, digitální mikroskop, zpracování obrazu

## KEYWORDS

Web API, MSSQL, Entity Framework, software, C#, MVC, WPF, XAML, desktop application, AJAX, http client, web service, digital microskope, image processing

# PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem diplomovou práci vypracoval samostatně dle pokynů vedoucího a s použitím uvedené odborné literatury.

V Brně dne 26. 5. 2017 …………………………………..

Martin Hortvík

# BIBLIOGRAFICKÁ CITACE

HORTVÍK, M. *Automatické zjišťování rozměrových změn betonových směsí při tuhnutí*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 68 s. Vedoucí diplomové práce Ing. Pavel Heriban, Ph.D.

## PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval vedoucímu Ing. Pavlu Heribanovi, Ph.D. za jeho čas, ochotu a cenné rady při tvorbě této závěrečné práce.

# OBSAH

# 1 ÚVOD

Neustálá poptávka po kvalitních stavebních materiálech ze strany spotřebitelů vyvolává tlak na výrobce, kteří v zachování vlastní konkurenceschopnosti kladou důraz na inovaci a vývoj nabízených produktů.

S přínosem vyšší technologické vyspělosti rozšířila společnost Profibaustoffe CZ, s.r.o. své vývojové centrum průmyslového výzkumu v Brně o nové laboratorní vybavení a stroje. V rámci vývoje nových produktů byly zavedeny odlišné postupy vyžadující automatizaci při sledování fyzikálních změn stavebních materiálů. V rámci nahrazení původních řešení vznikl požadavek na tvorbu měřící průmyslové aplikace, která ušetří čas a prostředky odborných pracovníků. V této věci byla kontaktována softwarová firma Pinya s.r.o., která se v současné době stará o jejich kompletní obchodní a logistické procesy skrze SharePoint portál.

Cílem předložené diplomové práce je návrh a realizace robustní desktopové aplikace včetně návrhu a implementace serverového portálu pro vzdálené vyhodnocování naměřených dat z experimentální stanice.

Úvodní část diplomové práce se zabývá analýzou úlohy a původního řešení. Následuje popis současných softwarových technologií při vývoji desktopových a serverových aplikací určených pro platformu Windows. Hlavní část práce obsahuje návrh a implementaci obou částí řešení. Na závěr je prezentováno ověření funkčnosti aplikace v reálném nasazení a splnění požadavků zadavatele.

# 2 ANALÝZA ÚLOHY

Následující kapitola pojednává o rozboru úlohy definované zadavatelem.

## 2.1 Zázemí a požadavky

Pro potřeby odborných pracovníků zadavatele je třeba vytvořit spolehlivý měřící software, který umožní sledování rozměrových změn tuhnoucí hmoty v reálném čase.

Pro měření je vyhrazena speciální místnost v podzemních prostorách budovy vývojového centra. Strategické umístění je vhodné zejména díky stálosti teploty a proudění vzduchu, které má vliv na probíhající měření. K dispozici jsou připraveny experimentální počítačové stanice, na kterých bude aplikace spuštěna (viz. obr. 1) [1].

*Obr. 1 Prostor a technické vybavení připravené k realizaci experimentálních měření* [1]

Časová náročnost běžného měření se pohybuje v rozmezí od jednoho do čtyř týdnů. Měření je nutné průběžně sledovat, což je pro pracovníky obsluhy velmi časově náročné. Z důvodu fyzické nedostupnosti místnosti je vhodné, aby šlo k aktuálně probíhajícímu měření přistupovat vzdáleně.

V tomto ohledu byla z důvodu zjednodušení navrhnuta varianta připojování se z kanceláře na experimentální stanice přes vzdálenou plochu. Od návrhu bylo však upuštěno z následujících důvodů [1]:

- Díky složité interní síťové infrastruktuře a zabezpečení musí být stanice v síti VPN, a proto se k výsledkům bez počítače v interní síti není možné dostat
- Sledování průběžných výsledků na mobilních zařízeních je prakticky nemožné
- Data z jednotlivých měření jsou roztroušena po stanicích

Všechna výše uvedená omezení řeší realizace serverového portálu, který bude sloužit jako centrální archiv všech provedených měření. Mezi hlavní výhody tohoto řešení patří zejména:

- Dostupnost všech naměřených dat s jednodušším ověřením totožnosti
- Přístup z přenositelných zařízení jako tablety a mobilní telefony
- Centralizace provedených měření a možnost porovnat výsledky na jednom místě
- Vizuální přehled aktuálně probíhajících měření

Přístup do serverového portálu dostanou odborní pracovníci vývojového centra a dle jejich zařazení budou moci přistupovat k jednotlivým částem systému. Ověřování totožnosti uživatelů portálu bude založeno na zadání přihlašovacího jména a privátního hesla [1].

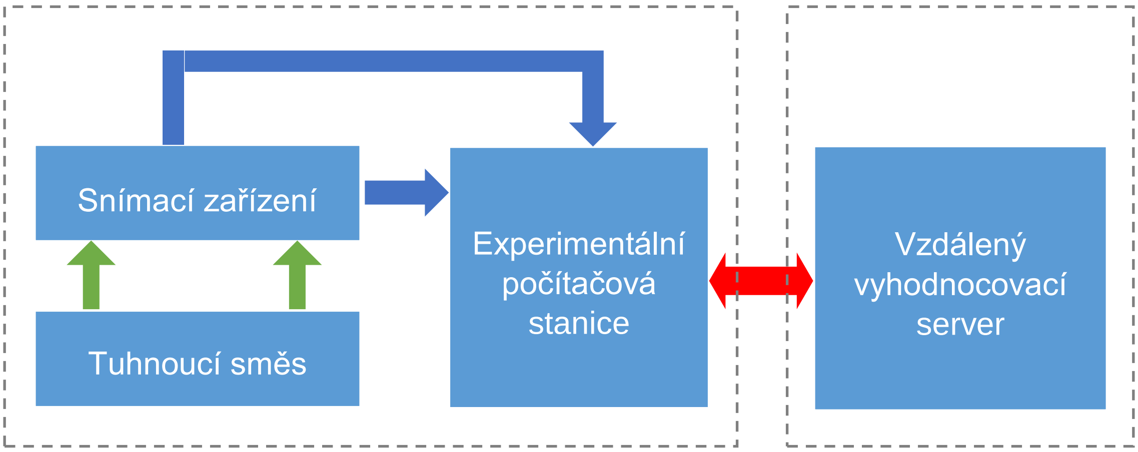
## 2.2 Základní rozvržení

Analýzou výše uvedených požadavků bylo navrhnuto řešení o dvou následujících dílčích částech:

*Experimentální část* – zahrnuje sledovaný objekt, snímací zařízení a desktopovou aplikaci běžící na počítačové stanici. Aplikace má za úkol řídit proces měření dle parametrů zadaných obsluhou a zpracovávat obrazová data. Současně komunikuje se vzdáleným serverem přes síťové rozhraní.

*Vyhodnocovací část* – webová aplikace běžící na vzdáleném serveru, která přijímá veškerá data z experimentální části a ukládá je na disk, zpracovává požadavky uživatele a je schopna v omezené míře řídit proces měření [1].

Datové propojení základních částí řešení je zobrazeno na obr. 2.




*Obr. 2 Blokové schéma základního návrhu řešení úlohy*

## 2.3 Příprava experimentální části

Experimentální část je zahájena zamícháním materiálové směsi odborným pracovníkem vývojového centra. Jakmile je směs připravena, rovnoměrně se rozlije na pružnou plochu ve tvaru obdélníka, který umožňuje pohyb hmoty v podélném směru.

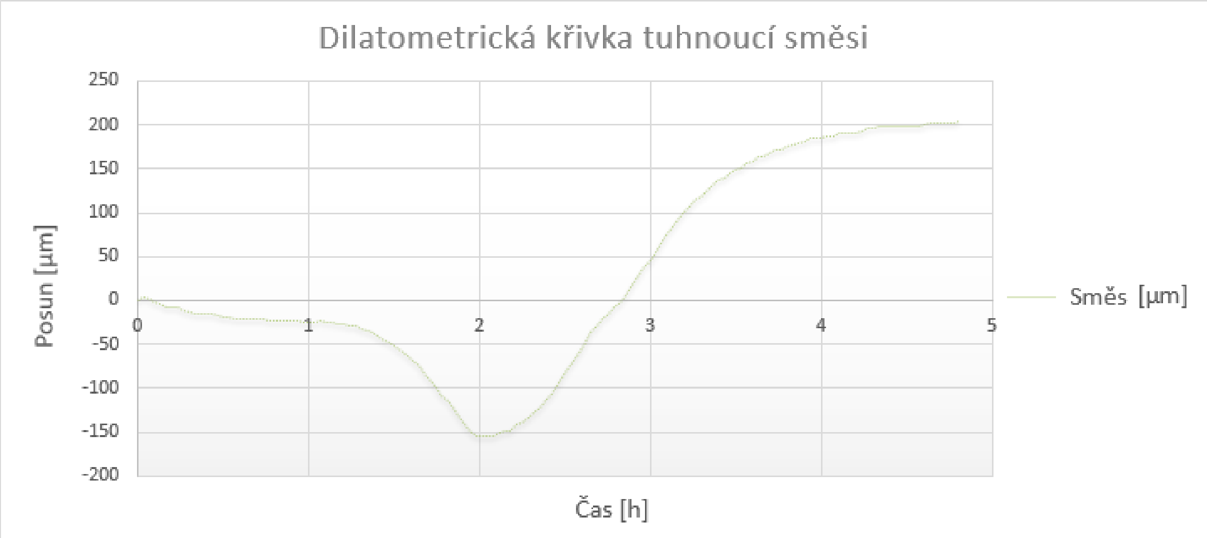
Při tuhnutí materiálu dochází vlivem chemických procesů uvnitř hmoty ke smršťování nebo roztahování. Experimentální metoda pro měření délkových změn se nazývá dilatometrie [1] [2].

### 2.3.1 Dilatometrie

Dilatometrie patří mezi experimentální metody, kdy sledujeme rozměrové změny pevných látek nebo objemovou roztažnost kapalin v důsledku fyzikálních či chemických procesů zkoumané látky. Tyto procesy mohou být sledovány v závislosti na čase nebo teplotě. Přístroje sloužící k takovému měření se nazývají dilatometry.

Výstupem dilatometrické metody bývají zpravidla dilatometrické křivky. Na základě jejich tvarů vyvozujeme závěry o zkoumané látce [2].

Na obr. 3 je uveden příklad dilatometrické křivky délkových změn tuhnoucí směsi. Graf znázorňuje závislost posunu hmoty (v mikrometrech) na čase (v hodinách).




*Obr. 3 Příklad dilatometrické křivky délkových změn sledované tuhnoucí směsi*

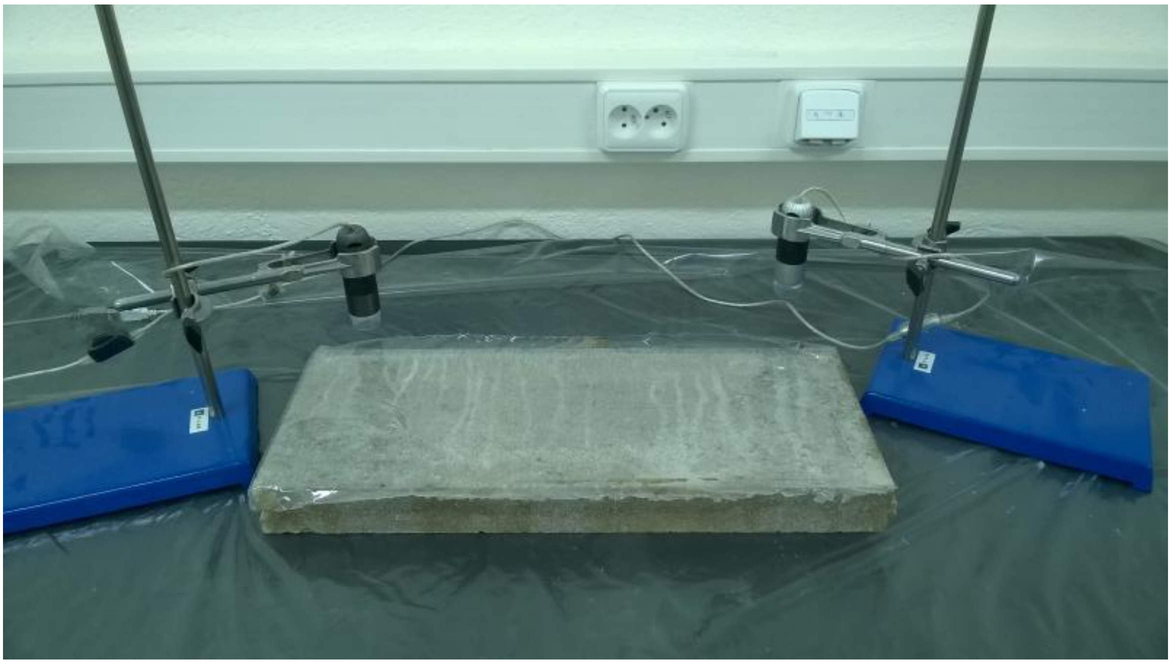
## 2.4 Rozbor úlohy

Úloha spočívá ve sledování rozměrových změn tuhnoucí směsi v horizontálních osách X a Y. Tyto změny probíhají na různých místech s různou intenzitou. Z toho důvodu musíme měřit minimálně ve dvou různých místech. Proto umístíme snímače na každou stranu obdélníka symetricky naproti sobě ve stejné vzdálenosti od středu. S touto vzdáleností poté počítáme při určování absolutního přetvoření hmoty v čase. Výstupem měření jsou dilatometrické křivky pro každý snímač včetně průměrné hodnoty z obou datových složek. Rozlišujeme tedy snímače:

*levý* – při výpočtu relativního přírůstku musíme v axiální ose provést výpočet s opačným znaménkem.

*pravý* – pro výpočet relativního přírůstku nemusíme nic upravovat.

Oba snímače před každým měřením nastavíme do správné pozice vůči sobě. Jsou pevně upnuty do ramen stojanů, aby během měření nedošlo k nechtěnému posunu. Základní rozvržení měřící aparatury úlohy lze vidět na obr. 4 [1].



*Obr. 4 Podložka pro rozlévání směsi se stojany a snímači* [1]

Rozměrové změny hmoty jsou okem téměř nepostřehnutelné, jejich velikost měříme v mikrometrech. Proto je nutné použít kvalitní snímače, ze kterých bude možné rozpoznat velikost změny [1].

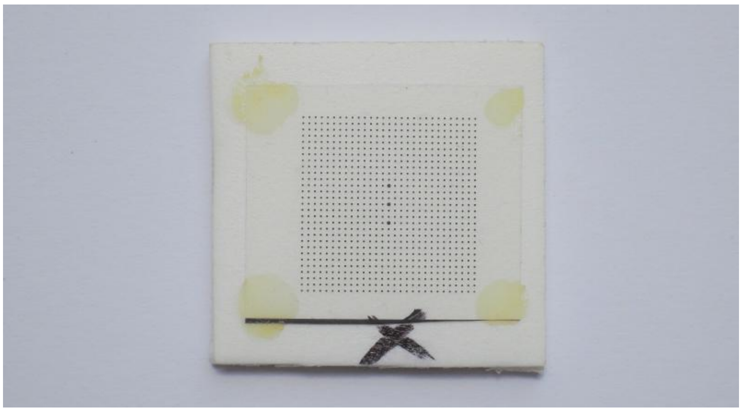
### 2.4.1 Určení polohy

Princip určení polohy spočívá ve snímání dilatometrického terče digitálním mikroskopem. Terčík je připevněn na malém plováčku z lehkého materiálu, který opatrně položíme přímo na měřenou hmotu. Terčík je tvořen značkami, ze kterých lze určit polohu jeho středu.

Absolutní souřadnice středu prvního snímku měření přestavují výchozí údaje pro výpočet relativních posunů terčíků. Každý další snímek bude vyhodnocen tak, že učíme absolutní pozici středu terče a odečteme ji od výchozích souřadnic prvního snímku.

Relativní vzdálenosti každého snímku v obou osách tvoří datový podklad pro tvorbu výsledných grafů.

Dilatometrické terče nechal zadavatel vyrobit profesionální tiskařskou firmou na zakázku (viz obr. 5). Terčík je tvořen čtvercovou mřížkou s 29 tečkami po straně o průměru 0,125 mm a rozteči 0,5 mm. Tři centrální tečky mají průměr 0,25 mm [1].



*Obr. 5 Detail dilatometrického terčíku na polystyrenové podložce* [1]

### 2.4.2 Snímací zařízení

Jako snímač je použit digitální mikroskop, který komunikuje s počítačem přes rozhraní USB. Snímky z mikroskopů jsou v definovaných časových momentech zasílány do počítače ke zpracování.

Z hlediska dobrého poměru cena/kvalita byly pro měření vybrány mikroskopy značky Levenhuk řady DTX 50, které zaručují spolehlivý chod na počítačích s operačním systémem Windows. Mezi nejdůležitější technické údaje patří [1] [3]:

- Rozlišení obrazového snímače: 1,3 MPx
- Zvětšení: 20 – 400x
- Zaostření: ruční 0–150 mm
- Dostupná rozlišení: 1920x1080 (interpolované), 1280x720, 640x480
- Výstupní formáty: *.jpeg, *.bmp, *.avi
- Snímková rychlost: 30 fps
- Osvětlení: systém s 8 LED žárovkami s regulací jasu

Na obr. 6 je zobrazena vzájemná konfigurace snímače a dilatačního terče při praktické úloze (bez směsi) [1] [3].


*Obr. 6 Mikroskop Levenhuk řady DTX 50 s dilatometrickým terčem* [1]

## 2.5 Původní řešení

V počátcích výzkumu a testování materiálových směsí bylo pro měření dilatometrických křivek použito odlišného přístupu. K počítačové stanici připojené snímače pořizovaly snímky v pevně stanovené frekvenci po celou dobu měření. Po dokončení měření byla obrazová data souhrnně zpracována programem, který exportoval výsledky.

Průběh délkových změn tuhnoucí směsi má přibližně logaritmický charakter. Nejvýznamnější změny připadají na prvních 48 hodin od zahájení měření, kdy je třeba pořizovat snímky s minimální periodou 5 minut. Po uplynutí této doby stačí snímat po 20 minutách.

Původní software snímkoval každé 2 minuty, což při průměrné velikosti snímku 1,5 Mb a délce měření 3 týdny dává dohromady 22 680 Mb = 22,7 Gb obrazových dat. Následné zpracování těchto dat bylo časově a technicky velmi náročné (každý snímek je vyhodnocen zvlášť). S technickým vybavením vývojového centra se časová náročnost zpracování naměřených dat pohybovala od 6 po 8 hodin plného vytížení experimentální stanice. Zde jsou shrnuté důvody, proč bylo potřeba nalézt nové řešení [1]:

- Časově náročné zpracování výsledků měření
- Bez složitého manuálního přesouvání dat prakticky nemožné dostat se k průběžným výsledkům měření
- Vyšší nároky na výkon počítačové stanice
- Kapacitní nároky pevného disku stanice z důvodu archivace
- Zbytečné zpracování redundantních obrazových dat bez vypovídající hodnoty po uplynutí prvních 48 hodin od počátku měření. Částečné řešení tohoto bodu spočívá v ručním protříděním duplicitních snímků obsluhou.
- Časově náročné manuální zákroky obsluhy
- Nepohodlná vzdálená kontrola měření
- Pokud došlo v průběhu měření k vychýlení snímačů či jiné chybě způsobené obsluhou či situací ústící k neplatnému měření, bez průběžného zpracování výsledků nikdo neví, že je potřeba aktuálně probíhající měření zrušit a začít znovu od začátku – markantní ztráta času.

Návrh nového řešení odstraňuje všechny výše uvedené nedostatky původního softwaru.

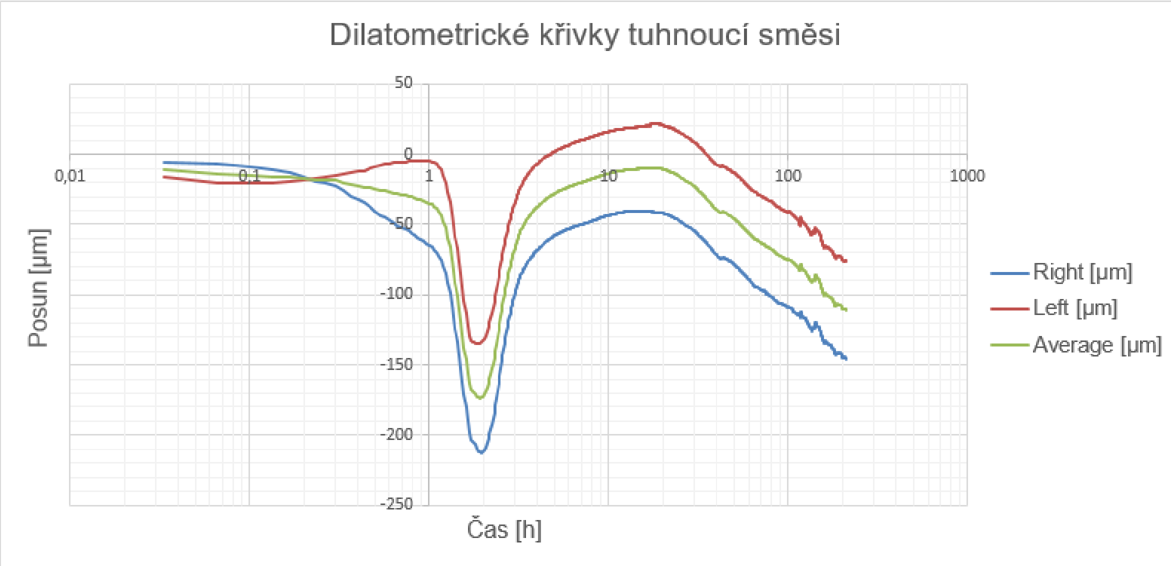
## 2.6 Reprezentace výsledků

Výsledkem měření materiálových směsí při tuhnutí jsou dilatometrické křivky. Pro odborné pracovníky vývojového centra slouží jako podklady k validaci, zda testovaná směs vyhovuje předepsaným parametrům.

Aplikace musí umožňovat přístup k aktuálním výsledkům měření ve formě excelového souboru s daty a sestrojenými grafy.

Na obr. 7 je uveden příklad dilatometrických křivek tuhnoucí směsi sestavených na základě naměřených dat z obou snímačů (levého a pravého). Graf vynáší závislost posunu dilatometrického terče v axiální ose v mikrometrech na čase v hodinách. Vodorovná osa je pro lepší vizuální přehled změn v logaritmickém měřítku.

Díky symetrickému umístění snímačů naproti sobě, můžeme obě řady dat zprůměrovat a vynést do grafu jako průměrnou hodnotu.



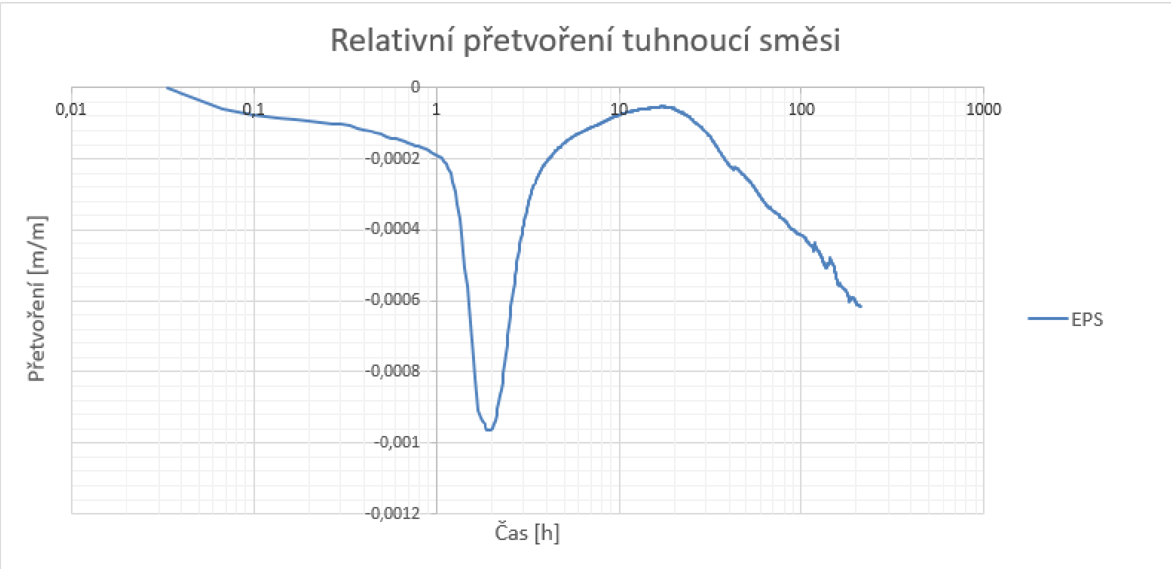

*Obr. 7 Příklad dilatometrických křivek tuhnoucí směsi*

Průměrná hodnota délkových změn obou snímačů v podélné ose bude použita pro sestrojení grafu relativního přetvoření materiálové směsi.

Uvažujeme stejnou vzdálenost snímačů od středu obdélníka. Tato vzdálenost bude zadána jako jeden z parametrů před spuštěním měření. K získání hodnoty výsledného přetvoření v jednotce metr na metr (m/m), musíme zadat délku ramene v mikrometrech. Výpočet relativního přetvoření je založen na [1]:

*Přetvoření [m/m]* = průměrná hodnota relativního posunu terče [µm] / vzdálenost středu terče od středu obdélníka (délka jednoho ramene) [µm]

Na obr. 8 je zobrazen graf relativního přetvoření tuhnoucí směsi při zadané délce ramene 180 000 µm. Vodorovná osa je rovněž v logaritmickém měřítku. Graf udává závislost relativního přetvoření (v metrech na metr) na čase (v hodinách).




*Obr. 8 Příklad grafu relativního přetvoření tuhnoucí směsi*

# 3 PŘEHLED SOFTWAROVÝCH TECHNOLOGIÍ

Při vývoji aplikací dnes může programátor ušetřit drahocenný čas používáním vhodných frameworků a knihoven, u kterých je spolehlivost a funkce prověřená mnohaletým používáním v řadě praktických aplikací.

Následující kapitola obsahuje základní seznámení se softwarovými technologiemi, architekturami, knihovnami a doplňky určenými pro platformu Windows, které budou použity při návrhu a realizaci obou částí řešení zadané úlohy.

## 3.1 Visual Studio

Nejznámějším vývojovým prostředím pro programování nových aplikací pro operační systém Windows je Microsoft Visual Studio. Nejnovější verze VS 2017 byla uvedena v březnu letošního roku a přináší inovace v podpoře dalších platforem jako je Android a iOS.

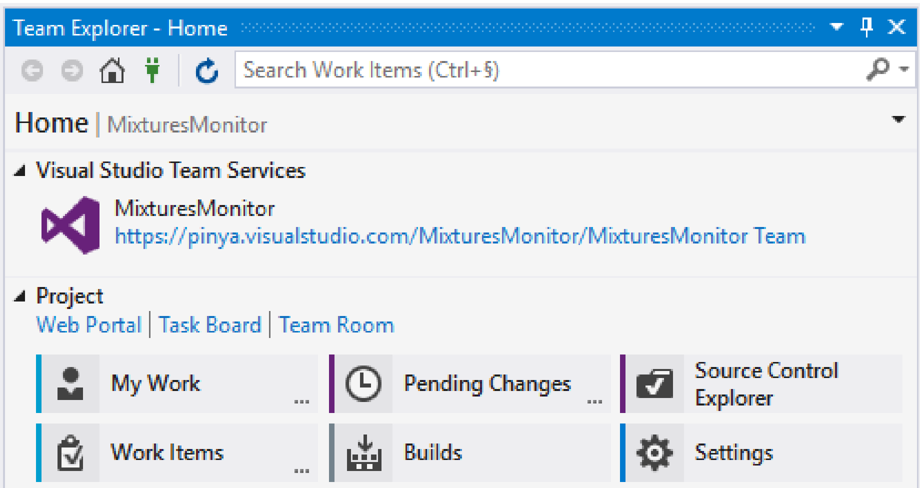
Slouží pro vytváření softwaru, od plánování přes tvorbu uživatelského prostředí, psaní kódu, testování, ladění a analýzu výkonu výsledné aplikace. Společně s následujícími doplňky tvoří silné zázemí pro vývoj profesionálních aplikací [4].

### 3.1.1 Visual Studio Team Services

Zkráceně VSTS, je označení pro cloudovou službu zajišťující plně agilní vývoj softwaru. Služba je tvořena serverovým repositářem projektů a sleduje veškeré změny zahrnutých souborů. Lokální změny souborů jsou zasílány na server, ke kterému se mohou připojit ostatní členové týmu a stáhnout si vždy nejaktuálnější změny.

Je ideálním nástrojem pro firemní účely, jelikož centralizuje všechny projekty na jednom místě s nastavitelnou viditelností pro jednotlivé členy týmu. Po založení účtu a konfiguraci připojení lze využívat mimo jiné těchto služeb: kompletní správa projektů, plánovací strategie, verzování a historie souborů, online editace kódu a další. Z hlediska zálohy dat je dnes při vývoji softwaru nepostradatelnou službou [5].

V rámci firemního účtu je celé řešení (MixturesMonitor) zálohované na tomto cloudovém serveru. Na obr. 9 lze vidět záložku Team Explorer otevřeného projektu.



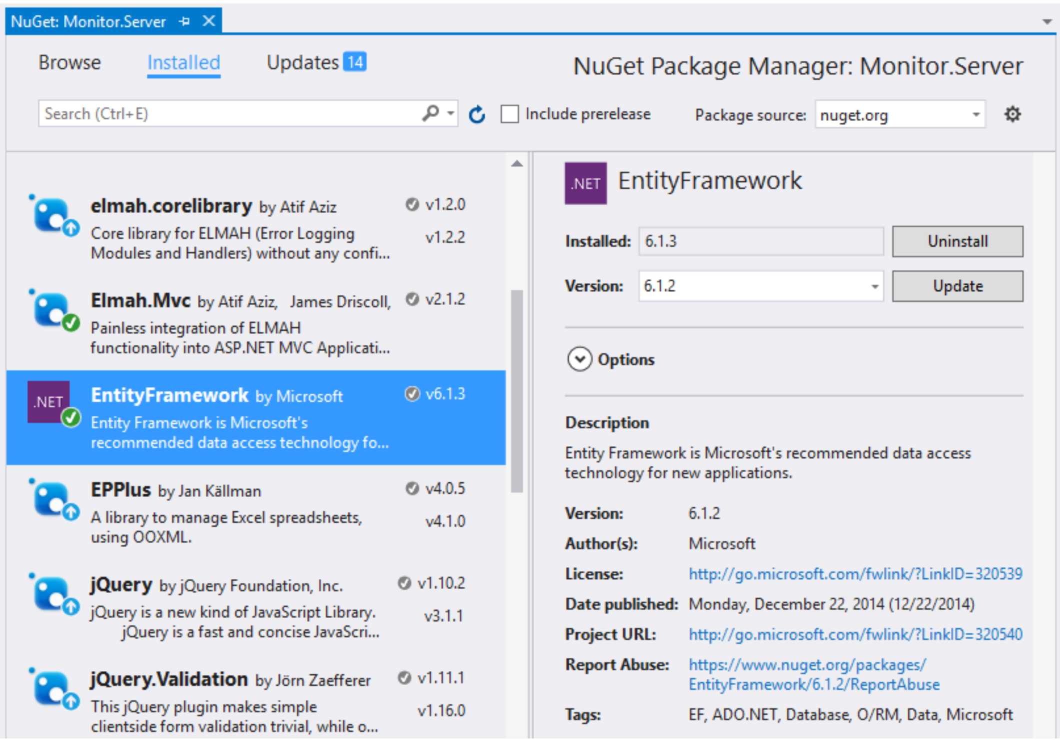

*Obr. 9 Záložka Team Exlorer Visual Studia 2015 s otevřeným projektem MixturesMonitor*

### 3.1.2 NuGet balíčky

NuGet Package Manager je nástroj pro správu doplňkových referencí v projektech. Tvoří jej jednoduché uživatelské prostředí a funkční konzole. Díky tomuto doplňku má programátor možnost aktualizovat použité knihovny (či vybírat z dostupných verzí) z jednoho místa. Od Visual Studia verze 2012 je tento nástroj zahrnut v každé edici.

V poslední verzi došlo k výrazným změnám uživatelského prostředí do aktuální podoby zobrazené na obr. 10. Původně byl doplněk realizován jako dialogové okno, které během stahování knihoven blokovalo celé vývojové prostředí. Časté zasekávání uprostřed aktualizací knihoven také vedlo k porušení dílčích souborů projektu. Všechny uvedené problémy se v rámci aktualizací podařilo opravit.

Zdrojem standartních balíčků je veřejná API nuget.org, kde jednotlivé organizace přidávají poslední verze populárních frameworků a knihoven pro vývojáře po celém světě. Vytvořit balíček může každý přihlášený uživatel Visual Studia a sdílet ho veřejně přes API nuget.org nebo v rámci své privátní organizace [6].




*Obr. 10 Zvolený NuGet balíček, stručný popis a aktuálně nainstalovaná verze*

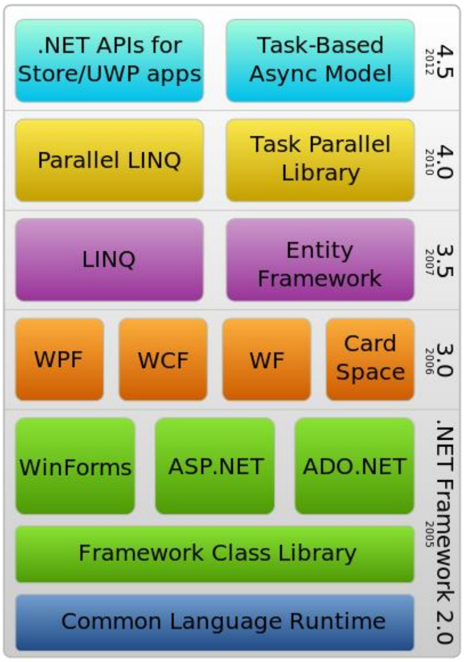
## 3.2 Použité programovací jazyky

V rámci celého řešení bylo použito několika programovacích jazyků. Zde se nachází stručná charakteristika tří nejvýznamnějších z nich.

### 3.2.1 C#

Microsoft Visual C# je výkonný, objektově orientovaný programovací jazyk, navržený pro vývoj široké škály aplikací běžících na .NET frameworku. Jako evoluce jazyků C a C++ je C# jednodušším, modernějším a typově bezpečnějším.

.NET framework je rozsáhlá softwarová platforma, která zahrnuje jazyk, virtuální stroj a knihovny. Na rozdíl od jazyků C a C++ není zdrojový kód přeložen kompilátorem do strojového kódu, nýbrž do mezikódu. Ten je spouštěn na virtuálním stroji, který zajišťuje běh a kompilaci aplikace pro konkrétní počítačovou sestavu. Základní struktura platformy .NET s dostupnými funkcemi v průběhu času je zobrazena na obr. 11 [7] [8].




*Obr. 11 Struktura .NET frameworku* [8]

Mezi hlavní výhody oproti výše zmíněným jazykům patří:

- Jednoduchý a rychlý vývoj – rozsáhlé knihovny a datové struktury
- Bezpečnost – správu paměti řeší běhové prostředí
- Kombinovatelnost – zdrojový kód jazyka Visual Basic přeložitelný do stejného mezikódu jako C#
- Snadnější odhalení chyb ve zdrojovém kódu

Nevýhodami .NET frameworku jsou vyšší paměťové nároky a pomalejší start aplikací. Pokud vyvíjená aplikace nevyžaduje zásadní práci s pamětí, hardwarem nebo grafikou (například ovladače nebo hry) a neobsahuje složité algoritmické výpočty, z hlediska rychlosti vývoje a robustnosti řešení, je výhodnější použít programovací jazyk C# [8].

### 3.2.2 HTML

Základním jazykem pro programování internetových stránek je Hypertext Markup Language, zkráceně HTML. Strukturu dokumentu tvoří hlavička a tělo složené z jednotlivých značek – tagů. Zdrojový kód není kompilován do binární podoby. Místo toho je jako textový soubor překládán internetovým prohlížečem do výsledné podoby.

Informace obsažené v hlavičce se nazývají metadata a určují chování stránky v prohlížeči (například kódování). Dále obsahuje odkazy na soubory s kaskádovými styly (CSS), které definují, jak budou jednotlivé tagy v prohlížeči vykreslovány.

Tělo dokumentu obsahuje tagy, které jsou dle definovaných pravidel do sebe zanořeny. Do nich vkládáme textový obsah, který chceme na stránce zobrazit. Tagy mohou být párové a nepárové. Při použití párových tagů musí programátor dbát na jejich správné uzavření v opačném pořadí jejich otevření [9].

V počátcích obsahovaly internetové stránky pouze text, obrázky a odkazy na další stránky. Dnes jsou webové stránky plně zaměřené na interakci s uživatelem. Obsahují formuláře na zadávání dat, tlačítka, která po kliknutí způsobí určitou změnu zobrazovaných informací, pohyblivé prvky, tabulkové seznamy, výběry, dialogová okna či přehrávače médií. Funkce těchto prvků jsou naprogramovány v jazyce Javascript. Kódové části Javascriptu mohou být umístěny přímo ve stránce, ale častěji jsou obsaženy v oddělených souborech, na které se v dokumentu pouze odkazuje.

Zdrojový kód můžeme psát v jakémkoliv textovém editoru, ale z hlediska rychlosti psaní, a především kontroly správnosti dokumentu, je vhodnější použít editor určený přímo pro jazyk HTML. Visual Studio nabízí propracovaný textový editor s automatickým zvýrazněním syntaxe a aktivní nápovědou při psaní, která značně urychluje postup při tvorbě internetových stránek.

Aktuální verze HTML 5 se od předchozí verze 4 liší zkrácenými zápisy značek. Zavedením nové sémantiky klade důraz na jednoduchost a přehlednost zdrojového kódu. Nové značky upravují strukturu dokumentu, umožňují práci s multimediálním obsahem, přidávají funkce a atributy formulářovým prvkům. V kombinaci s CSS 3 tvoří základ pro budování moderních webových stránek [9].

### 3.2.3 Javascript

Javascript je interpretovaný programovací jazyk se základními objektově orientovanými schopnostmi. Zdrojový kód je překládán za běhu ve webovém prohlížeči. Umožňuje dynamicky měnit obsah stránek, reagovat na podněty uživatele, validovat zadaná data ve formulářích či vykreslovat vektorovou grafiku.

Syntaxe je velmi podobná programovacím jazykům C a Java, tím ale podobnost s těmito jazyky končí. V návrhu nepoužívá třídy, ale funkce, které mohou být předávány jako proměnné. Dědičnost je zde založena na tvorbě objektových prototypů. Není typově bezpečný, což znamená, že všechny typy proměnných jsou odvozeny až při běhu kódu.

V poslední době dochází k rozšiřování funkcionalit na straně klienta zejména dostupností různých frameworků. Mezi nejznámější patří: JQuery, AngularJS a KendoUI. Nabízí funkce a interaktivní prvky – widgety, které používáme v kontextu webových stránek.

S technologií AJAX se stal Javascript datovým prostředníkem mezi serverem a klientem bez nutnosti obnovovat stránku, což vede k zajímavé uživatelské zkušenosti. Na tomto principu jsou založeny tzv. jednostránkové aplikace (Single Page Application SPA), kdy na jednu HTML stránku vykreslujeme různé stránky na základě navigace v adresním řádku. Veškerý obsah stránek je dynamicky získáván přes API běžící na webovém serveru. Příkladem takové aplikace je webový emailový klient Gmail [10].

## 3.3 WPF

Microsoft Windows Presentation Foundation, zkráceně WPF, je soubor knihoven pro tvorbu formulářových aplikací, který je součástí .NET frameworku od verze 3.0. Obsahuje běžné ovládací prvky, jako popisky, textová pole, výběry, posuvníky, tabulky, tlačítka i další složitější komponenty. Představuje alternativu uživatelského prostředí Windows Forms při vývoji formulářové aplikace programovacích jazyků rodiny .NET [7].

WPF přináší v návrhu grafického uživatelského prostředí oproti Windows Forms výrazné změny. Struktura prezentační vrstvy se inspiruje jazykem HTML v kombinaci s atributy XML, čímž vznikl jazyk XAML (Extensible Application Markup Language).

Nahrazuje tak absolutní pozicování prvků formuláře, které není přizpůsobeno změně tvaru a velikosti aplikačního okna (na mobilních zařízeních).

Rozvrstvení uživatelského prostředí také napomáhá při vývoji v týmu, kdy každý člen může pracovat nezávisle na jednotlivé vrstvě. Nakonec jsou tyto vrstvy propojeny pomocí předem definovaných datových modelů.

Další významnou změnou je použití vykreslovacího rozhraní Direct3D nahrazující původní pomalé rozhraní Windows GDI. Uživatelské prostředí je vykreslováno za pomoci grafického procesoru, a proto mohou být WPF aplikace rychlejší a méně zatěžovat hlavní procesor. Bez původního omezení výběru formulářových prvků, lze nyní jednoduše budovat graficky pokročilé aplikace [7] [11].

### 3.3.1 Architektura MVVM

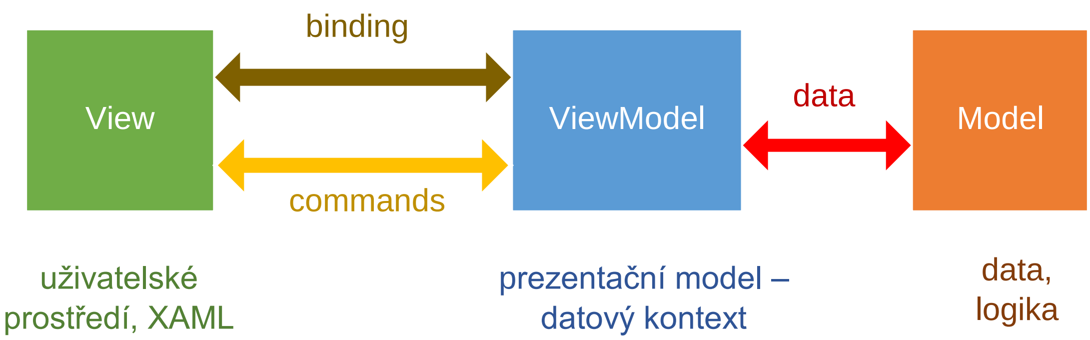
Model View ViewModel, zkráceně MVVM je návrhový vzor oddělující logiku aplikace od uživatelského rozhraní. Používá se v javascriptových frameworcích (například Knockout JS) a WPF desktopových aplikacích. Skládá se ze tří vrstev [12]:

- *Model* – obsahuje data a logiku, se kterou aplikace pracuje.
- *View* (pohled) – uživatelské prostředí aplikace (formuláře, webová stránka, ovládací prvek).
- *ViewModel* – obsahuje prezentační model, který spojuje model s pohledem. Ovládací prvky pohledu jsou oboustranně provázány s prezentačním modelem.

Prostředí WPF bylo přímo navrženo pro použití tohoto vzoru. V pohledové vrstvě lze definovat prezentační model – datový kontext. Jednotlivé prvky formuláře jsou na základě *bindingu* (provázání) oboustranně propojeny s datovým kontextem a při změně jednoho z nich jsou změny ihned propsány na druhé straně.

Obsluha událostí vyvolaných z pohledu funguje přes *command* (příkaz), který je v prezentačním modelu definován metodou. Tento způsob plně nahrazuje uživatelské prostředí řízené událostmi (Windows Forms) [12].

Na obr. 12 je zobrazeno blokové schéma MVVM architektury ve WPF.



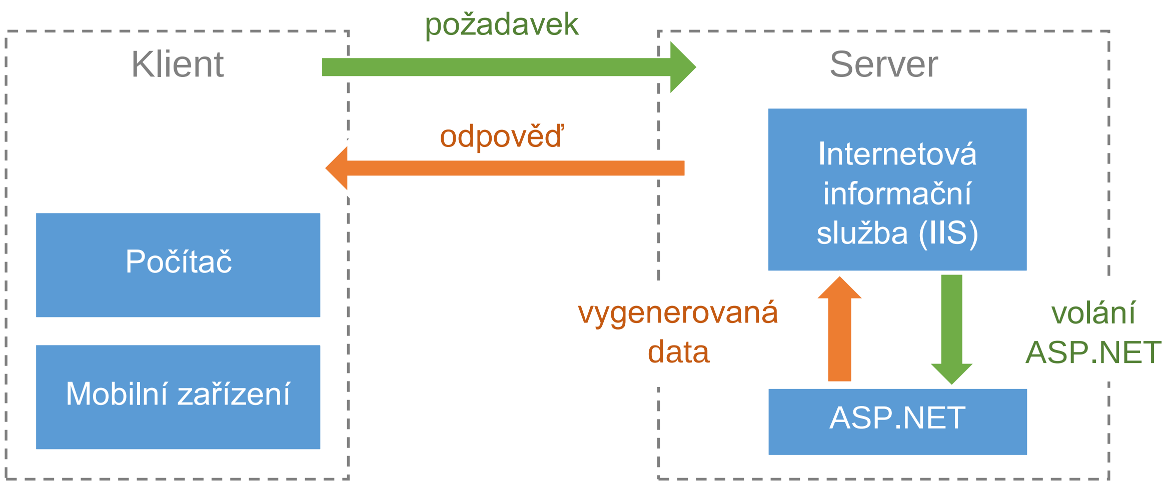

*Obr. 12 Blokové schéma MVVM architektury ve WPF* [12]

## 3.4 ASP.NET

Microsoft Active Server Pages .NET, zkráceně ASP.NET je souhrn knihoven pro vývoj webových aplikací v programovacím jazyce C#. Obsahuje funkce pro zpracování požadavků klienta, autentifikaci uživatelů, práci s databází a správou formulářů.

Dynamické webové stránky fungují na principu vygenerování HTML stránky na základě požadavku klienta. Serverová aplikace je spuštěna pod procesem internetové informační služby (IIS) nainstalované na webovém serveru. Ta má za úkol přeposílat klientské požadavky ASP aplikaci ke zpracování a poté odesílat vygenerované odpovědi zpět na zařízení klienta [13].

Zjednodušené blokové schéma komunikace klient-server při použití technologie ASP.NET je zobrazeno na obr. 13.




*Obr. 13 Blokové schéma základní komunikace klient-server technologie ASP.NET* [13]

Koncept ASP.NET zahrnuje dva základní způsoby, jak vytvářet webové stránky [13].

### 3.4.1 WebForms

WebForms je starší koncept, který simuluje desktopovou aplikaci v prohlížeči.

Z principu funkčnosti HTTP protokolu (je bezstavový) je nutné uchovat tzv. stav. Ten obsahuje informace o všech prvcích ve formuláři. Stavová proměnná (ViewState) je realizovaná skrytými poli se zakódovaným informacemi. Po odeslání formuláře na server se nám vrátí zpět modifikovaná stavová proměnná. Opětovné zasílání stavové proměnné představuje větší nároky na přenášená data, což snižuje rychlost tohoto řešení.

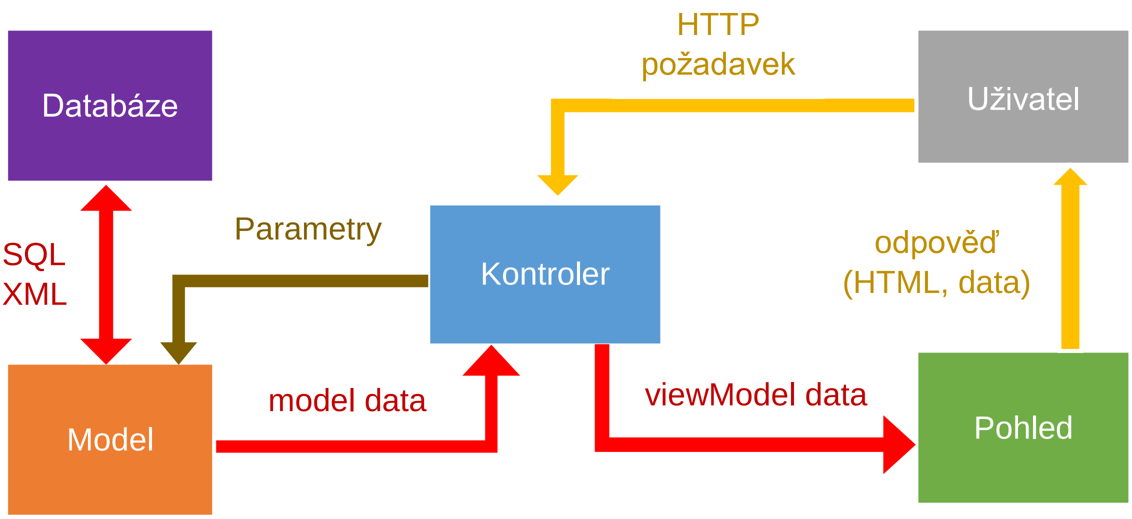
Struktura webových stránek se skládá z HTML šablony a kódu na pozadí. Ten upravuje atributem označené prvky šablony [7] [13].

### 3.4.2 MVC

Architektura MVC je novějším konceptem oddělujícím logiku od výstupu. Mezi vývojáři patří mezi oblíbené, protože může na jednom projektu pracovat více lidí zároveň. Celý vzor se skládá ze tří dílčích komponent [7] [13]:

- *Model* – obsahuje logiku (výběr dat z databáze, výpočtové algoritmy). Výstupem jsou data v surové podobě.
- *View* (pohled) – šablona, do které budou vložena data a vznikne tak výsledná HTML stránka. Obsahuje formuláře pro interakci s uživatelem.
- *Controller* (kontroler) – ústřední komponenta, která komunikuje s oběma dílčími částmi. Vybírá data z formulářů pohledu a předává parametry modelu k provedení logiky s daty. Přebírá data z modelu a vkládá je do pohledu.

Největším benefitem MVC architektury je oddělení pohledu od logiky modelu. Zatímco grafik připravuje styl pohledu, může další programovat vnitřní logiku aplikace. Na obr. 14 je zobrazeno blokové schéma MVC architektury v ASP.NET aplikaci.




*Obr. 14 Blokové schéma MVC architektury v ASP.NET aplikaci* [13]

## 3.5 AForge

Populární open source framework AForge je souhrn knihoven pro softwarové vývojáře .NET, kteří programují aplikace v oblasti počítačového vidění a umělé inteligence. Celý framework je rozdělen do následujících dílčích částí [14]:

- AForge.Imaging – nejobsáhlejší knihovna obsahuje algoritmy ke zpracování obrazu (převod barev, aplikování filtrů, změny rozměrů a rotace, rozpoznávání hran a tvarů, generování textur a šumu).
- AForge.Video – obsahuje funkce, které zpřístupňují obrazové toky z hardwaru (USB a IP kamery), čtení a zápis video souborů, podpora čipu Microsoft Kinect.
- AForge.Vision – počítačové vidění (algoritmy detekce pohybu)
- AForge.Math – matematické algoritmy
- AForge.Neuro – algoritmy neuronových sítí
- AForge.Genetic – genetické a evoluční algoritmy
- AForge.Fuzzy – fuzzy výpočtové metody
- AForge.MachineLearning – obsahuje algoritmy z oblasti strojového učení (Q učení, Boltzmannův stroj a další).
- Aforge.Robotics – obsahuje knihovny pro práci s populárními robotickými kity (Lego Mindstorm).

## 3.6 Autofac

Mezi populární vzory při programování rozsáhlejších .NET aplikací bezesporu patří Depencency Injection (DI), známé taky jako Inversion of Control (IoC). Díky němu je vyvíjená aplikace flexibilnější a snadnější k testování. Oddělováním závislostí tříd jsou pak jednotlivé části znovupoužitelné a eliminují tak nadbytečnost kódu.

Framework Autofac je IoC kontejner na správu instancí v .NET projektech. Lze ho použít při vývoji desktopových i webových aplikací. Do projektu ho můžeme snadno přidat jako NuGet balíček.

Umožňuje nám provádět změny v projektu přepsáním jednoho řádku konfiguračního souboru, což nám ušetří mnoho času při testování nových komponent projektu. Umožnuje nám řídit životní cyklus celé aplikace. Framework vytváří, udržuje a odstraňuje z paměti instance tříd, které používáme v průběhu aplikace.

Nejčastější forma použití spočívá ve vkládání objektů do konstruktoru třídy, ve které chceme přistupovat k metodám a vlastnostem objektu. Autofac se za nás postará o veškerou režii spojenou s vytvořením instance objektu dle konfigurace, která je načtena při spuštění aplikace. Na obr. 15 je zobrazena praktická ukázka použití Depencency Injection [15].

```
private readonly MeasureModel measureModel;
private readonly ILocalDataProvider dataProvider;

0 references | Martin Hortvík, 24 days ago | 1 author, 1 change
public ExportManager(MeasureModel measureModel, ILocalDataProvider dataProvider)
{
    if (measureModel == null) throw new ArgumentNullException("measureModel");
    if (dataProvider == null) throw new ArgumentNullException("dataProvider");

    this.measureModel = measureModel;
    this.dataProvider = dataProvider;
}
```

*Obr. 15 Ukázka vložení objektů do konstruktoru třídy* [15]

## 3.7 Entity framework

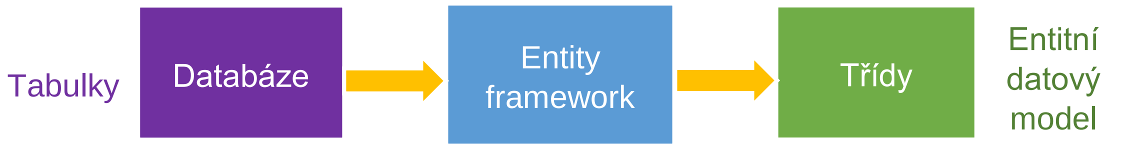
Entity framework je souhrn knihoven tvořící samostatnou vrstvu aplikace, pomocí které lze provádět dotazy nad databázemi a manipulovat s daty v nich obsaženými. V kombinaci s frameworkem ADO.NET tvoří základní nástroj pro práci s daty. Technologie LINQ to Entity poskytuje rozhraní mezi aplikací a databází překladem LINQ syntaxe do jazyka SQL.

Entity framework pracuje s entitním datovým modelem, který reflektuje fyzické tabulky v databázi. Pomocí rozhraní můžeme vytvářet třídy, které mapují prvky logického modelu databáze. Tímto oddělením jsme schopni zajistit nezávislost logiky aplikace na použitém zdroji dat. Pokud například přistupujeme k databázi Oracle a později k MSSQL, nemusíme v aplikaci, kromě mapování, nic upravovat. Jazykem LINQ to Entity provádíme dotazy a manipulujeme s daty prostřednictvím objektového modelu mapovaných entit skrze syntaxi jazyka LINQ.

Entity framework dnes představuje nejrozšířenější prostředek pro práci s databázemi .NET aplikací [7] [16].

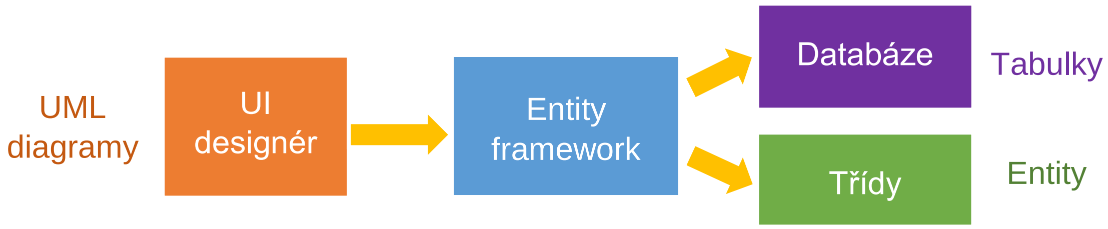
Při vývoji aplikace pracující s databází se dostáváme do tří scénářů [16]:

- *Databáze již existuje* – framework na základě fyzických tabulek databáze vygeneruje entitní datový model v podobě tříd (viz obr. 16.1).
- *Máme připravený entitní datový model* – z vytvořených tříd se vygeneruje logický model databáze (viz obr. 16.2).
- *Nemáme ani jedno* – pomocí uživatelského rozhraní můžeme navrhnout schéma databáze a framework za nás vygeneruje entitní datový model (třídy) i logický model databáze (databázi) (viz obr. 16.3).




*Obr. 16.1 Vygenerování tříd na základě existující databáze* [16]


*Obr. 16.2 Vygenerování logického modelu databáze na základě existujících tříd* [16]




*Obr. 16.3 Vygenerování databáze a entitního modelu na základě návrhu v UI designéru* [16]

## 3.8 Unit Of Work

Unit Of Work je programovací vzor pro manipulaci s daty. Společně s tzv. repozitáři (Repository) tvoří abstraktní vrstvu mezi databázovým kontextem a vnitřní logikou aplikace.

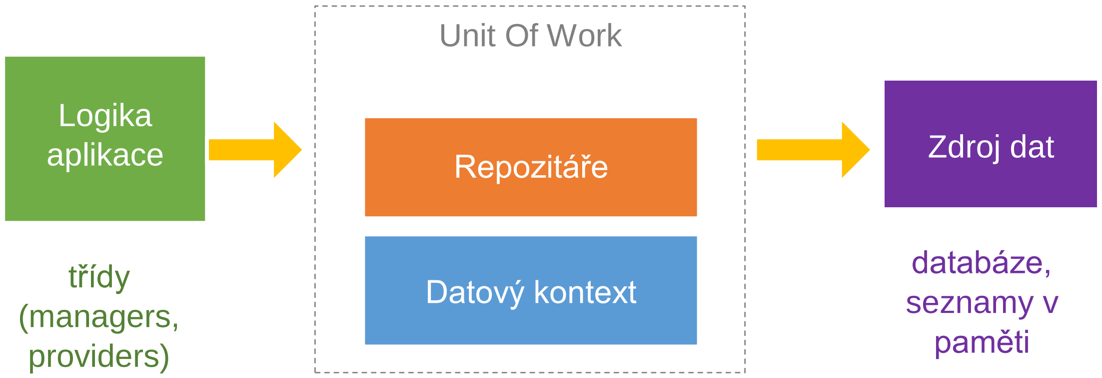
Repozitář je třída, která obsahuje databázový kontext a metody pro výběr a manipulaci s daty – tzv. CRUD operace [17]:

- *C (Create)* – vytvoření záznamu
- *R (Read)* – metody pro výběr dat ze zdroje
- *U (Update)* – mechanismus pro modifikaci dat
- *D (Delete)* – odstraňování záznamů ze zdroje

Rodičem repozitáře je interface IRepository, který definuje základní CRUD metody. Tyto metody jsou potomkem implementovány na základě druhu repozitáře. Například při testování aplikace může repozitář načítat data pouze ze seznamů v paměti (zdrojem je list v paměti). Následně pak přejdeme na databázi SQL a nebudeme muset měnit logiku aplikace – pouze vytvoříme instanci jiného druhu repozitáře (zdrojem je SQL server).

Každý entitní typ datového kontextu aplikace je potomkem třídy Repository. Zdědí tedy metody pro výběr a manipulaci s daty. Abychom mohli pracovat s kompletním datovým kontextem, musíme jednotlivé repozitáře zabalit do jednoho balíku. Nazýváme jej Unit Of Work. Ten obsahuje veškerá data a nástroje pro manipulaci s daty. Pokud v aplikaci používáme Dependency Injection, snadno si Unit Of Work vložíme do konstruktoru třídy, kde budeme chtít s daty pracovat [17].

Na obr. 17 je zobrazeno základní blokové schéma programovacího vzoru Unit Of Work s repozitáři.




*Obr. 17 Základní blokové schéma programovacího vzoru Unit Of Work s repozitáři* [17]

# 4 NÁVRH ŘEŠENÍ

Na základě analýzy úlohy uvedené v kapitole 2 byl vypracován návrh obou částí řešení. Počítačová infrastruktura zadavatele běží na technologiích Microsoft. Stejnou platformu využívá ke svým službám i softwarová společnost zajišťující vzdálený server.

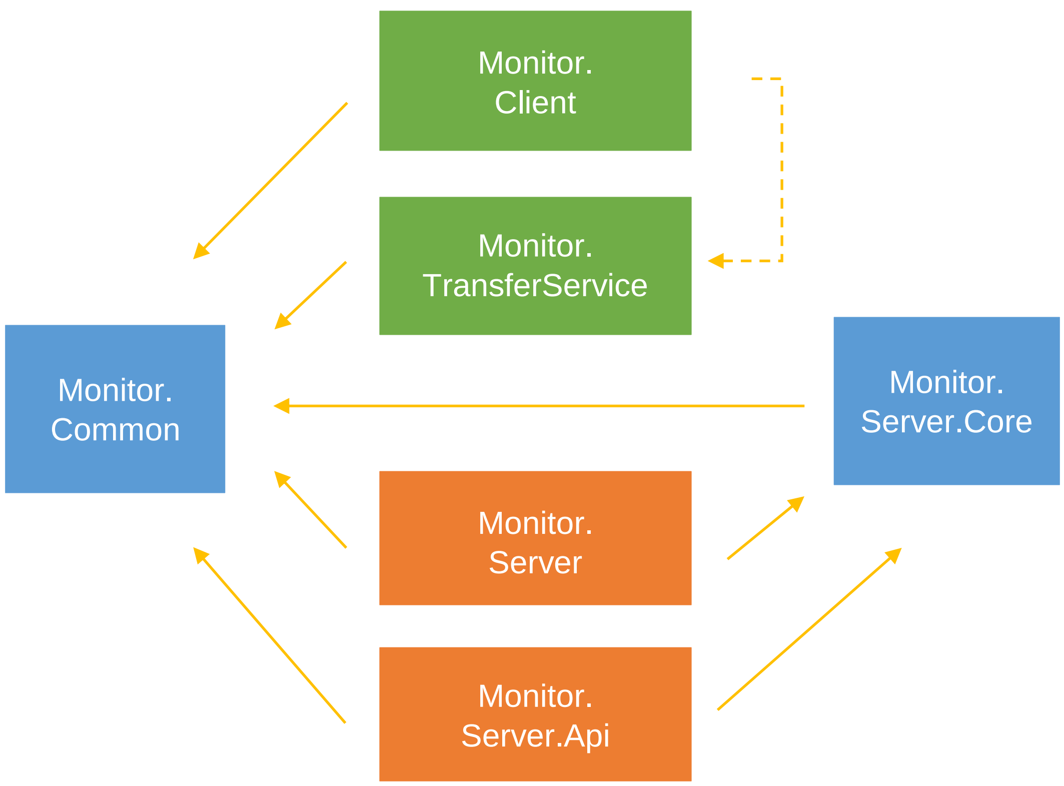
Aplikace budou naprogramovány ve vývojovém prostředí Visual Studio 2015. Celé řešení bude zálohováno přes službu VSTS a doplněno knihovnami nástrojem NuGet Package Manager. Použité technologie při vývoji jsou vysvětleny v kapitole 3.1.

## 4.1 Základní struktura

Dle komplexnosti úlohy je struktura řešení MixturesMonitor rozdělena na 6 dílčích projektů:

- *Monitor.Client* – desktopová WPF aplikace
- *Monitor.Common* – sdílená knihovna mezi všemi projekty obsahující pomocné třídy, rozšíření tříd, nastavení projektů a základní definici modelu.
- *Monitor.Server* – webová aplikace ASP.NET architektury MVC
- *Monitor.Server.Api* – webová aplikace ASP.NET Web API
- *Monitor.Server.Core* – sdílená knihovna mezi serverovými aplikacemi obsahující datovou strukturu, pomocné třídy a konektor databáze.
- *Monitor.TransferService* – webová služba zajišťující přenos obrazových dat

Na obr. 18 jsou v blokovém diagramu zobrazeny vzájemné závislosti jednotlivých projektů.



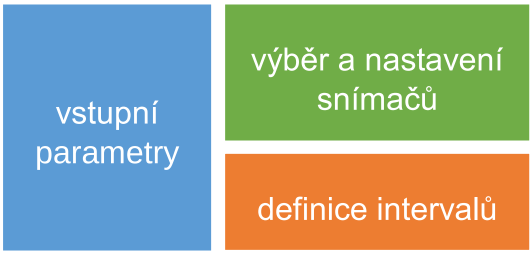

*Obr. 18 Blokové schéma vzájemných závislostí projektů navrhovaného řešení*

## 4.2 Desktopová aplikace

Požadavkům na spolehlivou desktopovou aplikaci s formulářem a grafickými prvky lze vyhovět použitím WFP aplikace na .NET frameworku 4.5 v jazyce C#. Vývojové prostředí umožňuje navrhnout kompletní uživatelské prostředí aplikace (jazyk XAML).

Před začátkem měření je nutné do aplikace zadat vstupní parametry, nadefinovat měřící intervaly, vybrat a nastavit obrazové snímače. Návrh okna vstupní části je zobrazen na obr. 19.1.

Po spuštění měření bude aplikace informovat o aktuálně probíhajícím intervalu, aktuálních výsledcích ve stručné podobě – snímky ze snímačů a souřadnice pro každý snímač zvlášť. Pro vizuální kontrolu obsluhou musí být snímky zobrazeny takto: vlevo umístit první snímek v aktuálním intervalu a vpravo od něj dát poslední snímek. Aplikace musí umožnit dílčí export výsledků a přerušení probíhajícího měření. Po skončení měření obsluha na základě výsledků určí, zda je nutné v měření pokračovat – možnost přidat další interval. Návrh okna probíhajícího měření je zobrazen na obr. 19.2 [1].




*Obr. 19.1 Návrh okna vstupní části*


*Obr. 19.2 Návrh okna při měření*

Obrazové snímače musí být zaostřeny a nastaveny do správné polohy vůči snímanému dilatometrickému terči. Z toho důvodu musí aplikace umožnit náhled z vybraného snímače v dialogovém okně.

Vnitřní logika aplikace bude obsahovat následující funkce [1]:

- Zpracování vstupních dat z formulářů – použití architektury MVVM. Definice prezentačních modelů pro všechna okna aplikace.
- Řízení životního cyklu aplikace, instance tříd – framework Autofac. Registrace tříd v konfiguraci před spuštěním aplikace.
- Správa vláken – časovače, zpracování snímků a export výsledků na vlastních vláknech mimo hlavní UI.
- Ukládání dat na stanici – lokální XML databáze. Tvorba složkové struktury, zápis a čtení XML souborů přes LINQ dotazy.
- Komunikace se vzdáleným serverem – asynchronní HTTP klient.
- Zpracování výsledků – generování excelových souborů s naměřenými daty včetně automatické konstrukce spojnicových grafů s pomocí populární knihovny EPPlus.
- Řízení a výstup snímačů – přístup k obrazovému toku z USB periferií, detekce kruhů ve snímku a výpočet souřadnic těžiště s pomocí knihoven AForge.
- Upload snímků – reference na webovou službu Monitor.TransferService, která se kompletně stará o HTTP přenos snímků na server.

## 4.3 Serverový portál

Serverová část řešení je určena ke vzdálenému vyhodnocování naměřených dat. Dle požadavků zadavatele se bude jednat o webovou aplikaci, která bude obsahovat data ze všech provedených měření a umožňovat řízení desktopové aplikace v průběhu měření.

Z rozboru požadavků zadavatele a dostupného technického zázemí softwarové firmy, vychází jako nejvýhodnější řešení webová aplikace ASP.NET s použitím architektury MVC v .NET frameworku 4.5. Server nabízí MSSQL Express databázi k ukládání dat a dostatek volného místa na disku pro upload snímků. Serverové řešení bude spuštěno na místním IIS (Internet Information Services) [1].

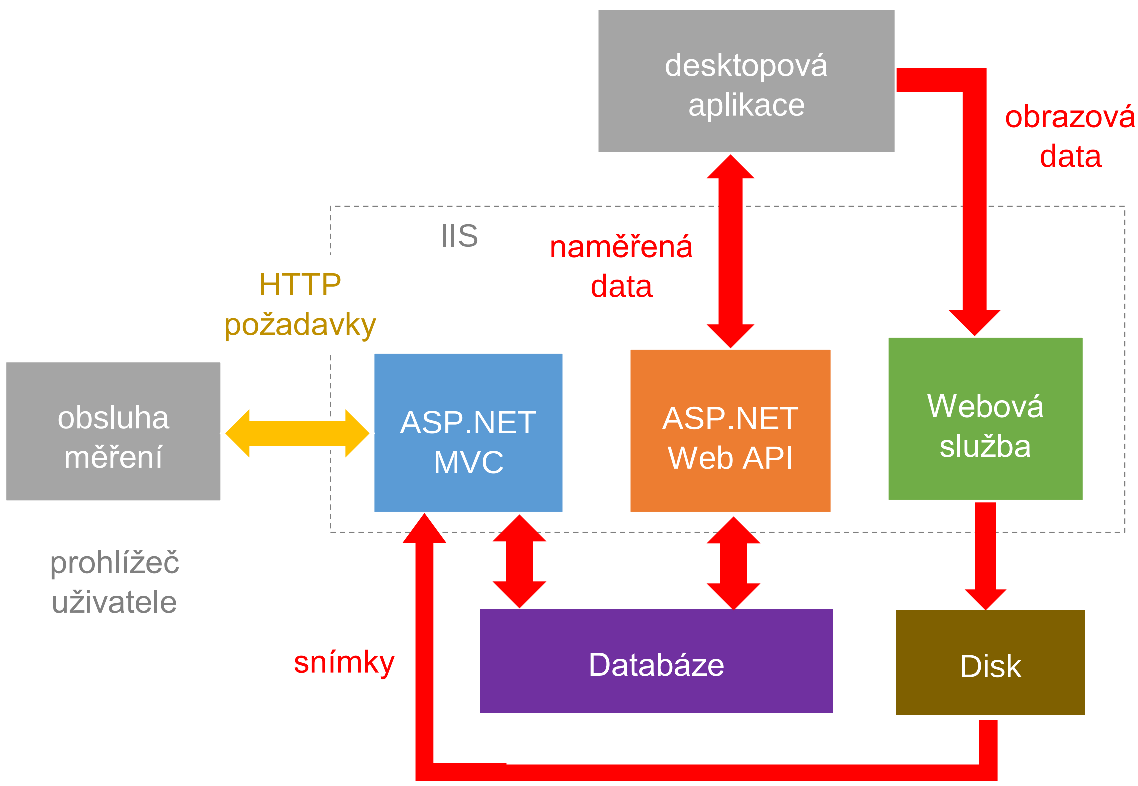
Požadované funkce serverové části nelze realizovat jedinou webovou aplikací. Z toho důvodu se návrh skládá ze tří dílčích projektů:

- *Webové aplikační rozhraní* (Monitor.Server.Api) – projekt, který představuje datové komunikační rozhraní mezi desktopovou aplikací a databází serveru.

- *Webová aplikace* (Monitor.Server) – prezentační webové stránky pro interakci s uživateli. Obsahuje výstupy naměřených dat a možnost řídit probíhající měření.

- *Webová služba* (Monitor.TransferService) – projekt, který přijímá snímky z desktopové aplikace a ukládá je na disk.

Každý projekt bude mít v rámci IIS svůj vlastní port pro veřejnou komunikaci. Na obr. 20 je zobrazeno blokové schéma serverové části řešení v kontextu desktopové aplikace a uživatele.




*Obr. 20 Blokové schéma serverové části řešení v kontextu desktopové aplikace a uživatele*

### 4.3.1 Webové aplikační rozhraní

Ústředním prvkem serverového řešení z hlediska přenosu dat je webové aplikační rozhraní ASP.NET Web API. Bude komunikovat s desktopovou aplikací pomocí HTTP protokolu a ukládat příchozí data do MSSQL databáze. Připojení k databázi včetně datového modelu bude zprostředkováno sdíleným serverovým projektem Monitor.Server.Core.

Webové rozhraní bude obsluhovat následující požadavky z desktopové stanice [1]:

- Příjem hlavičky měření – zadané parametry, vybrané snímače a definované intervaly měření.
- Příjem naměřených dat – stanice dle nastavených intervalů zasílá jednotlivé body.
- Příjem změny stavu desktopové aplikace – ukončení měření obsluhou stanice.
- Příjem nového intervalu po dokončení měření – příprava na pokračování v měření.

Desktopové stanici bude poskytovat tyto data [1]:

- Stav měření – uživatel webu ukončí probíhající měření.
- Nový interval – uživatel webu přidal nový interval, měření na stanici bude pokračovat.
- Seznam produktů – pro výběr produktu jako jeden ze vstupních parametrů měření.

Komunikace pomocí HTTP protokolu není šifrovaná, proto musí být zajištěn mechanismus ověření identity požadavků.

### 4.3.2 Webová aplikace

Prezentační serverovou část tvoří webová aplikace ASP.NET s architekturou MVC. Stejně jako rozhraní Web API bude přistupovat k datům z databáze prostřednictvím sdílené knihovny Monitor.Server.Core, která poskytuje datový model a třídy s vnitřní logikou.

Struktura aplikace vychází z rozboru požadovaných funkcí a skládá se z několika modulů [1]:

- *Modul Produkty* – interaktivní seznam produktů. Slouží jako zdroj dat pro desktopovou aplikaci. Výběr produktu je jedním ze vstupních parametrů měření.

- *Modul Měření* – je tvořen seznamem všech provedených měření a jejich detaily na samostatných stranách. Karta měření bude rozdělena dle obsahu na:
    - Detaily – informace o aktuálním intervalu, zadané vstupní parametry měření a data z obou snímačů. První a poslední snímky intervalu, aktuální relativní přírůstky polohy těžiště. Editační pole pro poznámky.
    - Intervaly – vizuální seznam intervalů
    - Levé a pravé snímky – oddělené galerie snímků seskupených podle intervalu včetně modálního náhledu.
    - Dokumenty – správce souborů; interaktivní seznam a upload příloh
    - Historie – kompletní seznam změn a provedených akcí

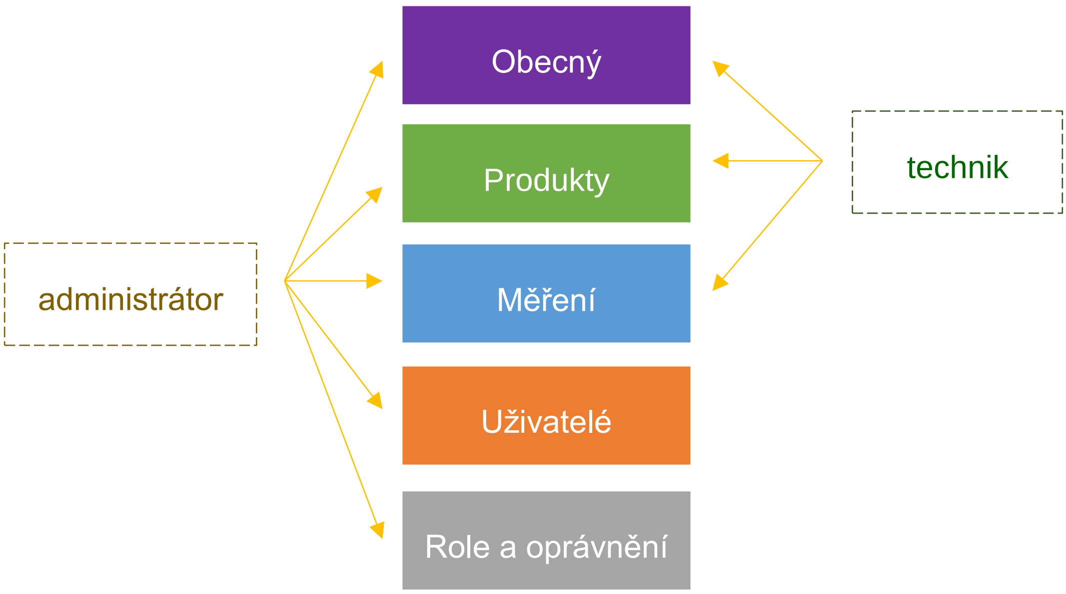
- *Modul Uživatelé* – správa uživatelů webového portálu. Obsahuje formulář na založení, seznam a samostatnou stránku s detaily. Karta uživatele je tematicky rozdělena na:
  - Editační formulář – úprava informací, nastavení přihlašovacího hesla
  - Emaily – seznam odeslaných emailů z portálu
  - Dokumenty – správce souborů
  - Historie – seznam všech změn a akcí, které uživatel provedl

- *Modul Role a oprávnění* – správa rolí a k nim přiřazených oprávnění. Definuje rozsah změn položek modulů, které může uživatel s danou rolí provést. Dělí se na:
  - Viditelné – může prohlížet položky modulu
  - Přidat – může přidat položky modulu
  - Upravit – může upravit položky modulu
  - Smazat – může mazat položky modulu

- *Modul Obecný* – obsahuje obecné stránky portálu:
  - Přihlašovací stránka – validace přihlašovacích údajů
  - Úvodní stránka – vizuální seznam aktuálně probíhajících měření
  - Události – seznam všech změn a akcí v celém portálu
  - Můj profil – základní informace o přihlášeném uživateli
  - Změna hesla – formulář k nastavení hesla

Přístup do webového portálu bude umožněn po ověření totožnosti zadáním přihlašovacího jména a hesla. Základní rozdělení uživatelů je Administrátor a Technik. První z uvedených bude mít na správu všechny moduly portálu. Technik bude mít k dispozici pouze moduly týkající se měření [1].

Na obr. 21 je zobrazeno blokové schéma přístupu modulů portálu na základě typu uživatele.




*Obr. 21 Blokové schéma přístupu uživatelů k modulům portálu*

### 4.3.3 Webová služba

Třetí část serverového řešení tvoří webová WCF služba Monitor.TransferService. která má za úkol zprostředkování přenosu snímků z desktopové stanice na server přes HTTP protokol.

Stejně jako všechny serverové části bude spuštěna v IIS. Abychom mohli přistupovat ke snímkům z prezentačního webu, bude nutné vytvořit virtuální složku, do které budou snímky ukládány.

# 5 REALIZACE DESKTOPOVÉ APLIKACE

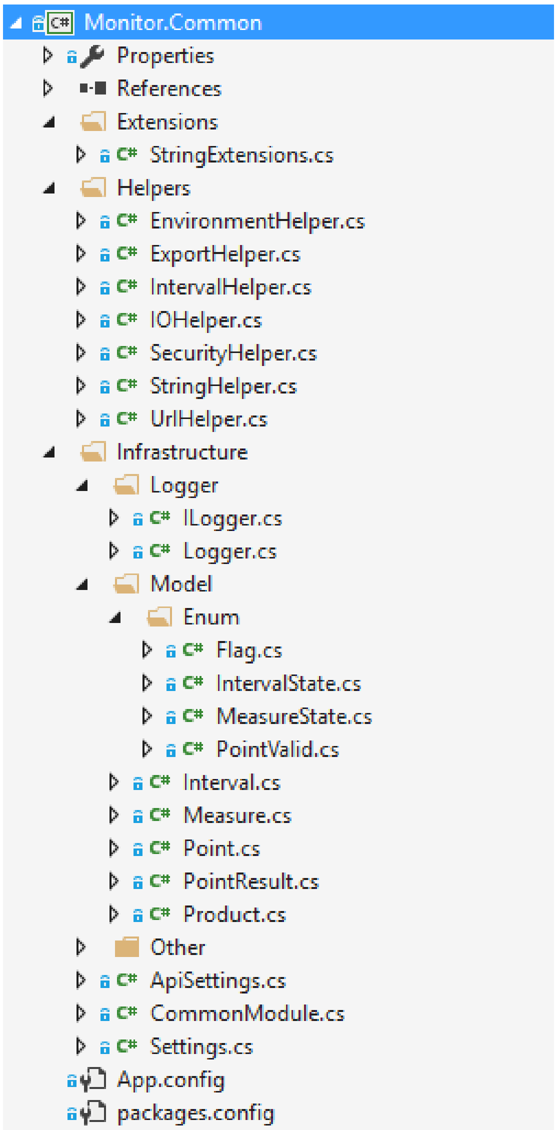
V následující kapitole je popsán postup při vývoji desktopové aplikace, jejíž návrh byl proveden v kapitole 4.2.

## 5.1 Základní struktura

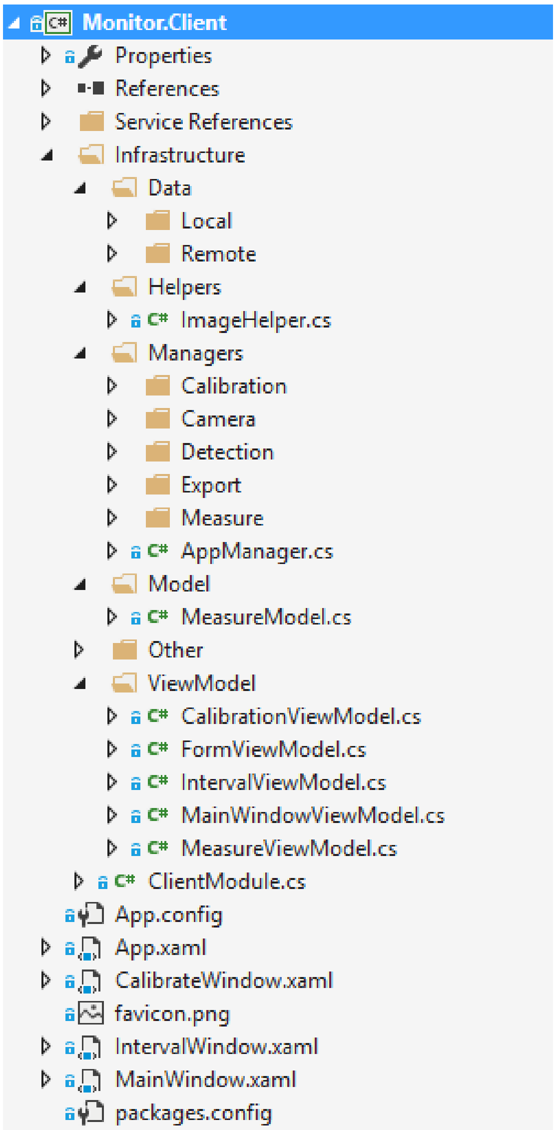
Při tvorbě moderních aplikací je kladen důraz na rychlost vývoje a přehlednost řešení při rozšiřování a provádění změn. K tomu napomáhá rozdělení na jednotlivé projekty a vhodný návrh vzájemných závislostí projektů.

V řešení byly založeny dva základní projekty:

- *Monitor.Common* – sdílená knihovna obsahující pomocné třídy, globální parametry projektů a definici základního modelu. Struktura projektu je zobrazena na obr. 22.1.
- *Monitor.Client* – desktopová WPF aplikace se závislostí na sdílené knihovně. Struktura projektu je zobrazena na obr. 22.2.




Obr. 22.1 Struktura projektu Monitor.Common




Obr. 22.2 Struktura projektu Monitor.Client

## 5.2 Sdílená knihovna

Abychom zbytečně neduplikovali programované funkce, přidáme do řešení sdílenou knihovnu Monitor.Common, která bude referencí v každém dalším projektu.

Struktura knihovny je navržena ke snadné dosažitelnosti nejpoužívanějších funkcí. Mezi ně patří složka Helpers (viz obr. 22.1) obsahující následující statické pomocné třídy:

- *EnvironmentalHelper* – obsahuje funkce zajišťující definici, vytvoření a kontrolu složkové struktury, zápis souborů na disk. Kořenovým adresářem složkové struktury jsou sdílené dokumenty.

- *ExportHelper* – obsahuje funkce pro tvorbu excelových souborů s naměřenými daty a konstrukci grafů.

- *IntervalHelper* – tvoří jej funkce pro převádění a manipulaci se seznamy intervalů.

- *IOHepler* – obsahuje funkce pro manipulaci se složkami a soubory.

- *SecurityHelper* – obsahuje funkci k šifrování řetězců na SHA256 hash.

- *StringHelper* – definuje základní řetězcové formáty časů a datumů. Dále vrací naformátované body a souřadnice.

- *UrlHelper* – obsahuje funkci, která modifikuje vstupní řetězec tak, aby byl validní pro použití v adresním řádku prohlížeče.

Ostatní funkční třídy jsou umístěny ve složce Infrastructure a dále se rozdělují na:

- *Logger* – tvoří jej třída implementující rozhraní ILogger. Slouží k výpisu informací z aplikace a případných vnitřních výjimek, které mohou nastat během chodu aplikace. Zapisování probíhá přidáváním textu na konec souboru umístěného v kořenovém adresáři vygenerované složkové struktury.

- *ApiSettings* – statická třída definuje URL adresy použité při HTTP komunikaci se serverovou API. Umožňuje zvolit si server – při vývoji lokální, následně pak vzdálený.

- *Settings* – statická třída obsahuje globální nastavení, které je definováno v konfiguračních souborech konkrétních projektů. Dále obsahuje jména složek pro nahrávání příloh modulům webové aplikace a definuje název virtuální složky.

- Složka *Other* – obsahuje třídy pro mapování položek seznamů.

Zbývající složka s názvem *Model* definuje datový model aplikace. Ten je rozveden v následujícím odstavci.

## 5.3 Datový model

Tvoří jej třídy s veřejnými vlastnostmi, které slouží jako datový základ desktopové aplikace. Třídy datového modelu jsou také použity při HTTP přenosu (zakódované do formátu JSON) a na straně serveru zpět dekódovány do původní podoby. Následují definice jednotlivých tříd a výčtů datového modelu:

**Třída Measure**

| Název | Datový typ | Komentář |
|---|---|---|
| Number | string | Unikátní identifikátor |
| Product | string | Název produktu |
| Name | string | Název |
| Description | string | Popis |
| Note | string | Poznámka |
| DateTime | DateTime | Datum a čas vytvoření |
| State | Enum | Stav měření |
| LeftCam | int | Index levého snímače |
| RightCam | int | Index pravého snímače |
| LeftCamName | string | Název levého snímače |
| RightCamName | string | Název pravého snímače |
| LeftRes | int | Index rozlišení levého snímače |
| RightRes | int | Index rozlišení pravého snímače |
| LeftResName | string | Název rozlišení levého snímače |
| RightResName | string | Název rozlišení pravého snímače |
| LeftPx | int | Dolní mez detekce levého snímače |
| RightPx | int | Dolní mez detekce pravého snímače |
| SaveAll | bool | Ukládat všechny snímky |
| TargetDist | decimal | Délka ramene |
| Intervals | List<Interval> | Kolekce intervalů |

Unikátní identifikátor je definován jako datum a čas spuštění měření ve formátu ssmmHHddMMyyyy. Enum MeasureState obsahuje následující výčet stavů:

- Running = 0 – probíhající měření
- Completed = 1 – měření dokončeno
- Aborted = 2 – měření přerušeno

**Třída Interval**

| Název | Datový typ | Komentář |
|---|---|---|
| Duration | Decimal | Trvání intervalu v hodinách |
| Ticks | Decimal | Perioda snímání v minutách |
| Count | Int | Počet snímků |
| Order | int | Pořadí |

**Třída Bod**

| Název | Datový typ | Komentář |
|---|---|---|
| GlobalTicks | int | Pořadí bodu od začátku měření |
| Timestamp | string | Datum a čas vytvoření |
| Intereval | string | Pořadí aktuálního intervalu |
| LeftAbs | string | Absolutní souřadnice levého těžiště |
| RightAbs | string | Absolutní souřadnice pravého těžiště |
| LeftRel | string | Relativní přírůstek levého těžiště |
| RightRel | string | Relativní přírůstek pravého těžiště |
| DistPx | string | Levá a pravá velikost pixelu |
| Valid | Enum | Validita bodu |

Enum PointValid definuje stavy vyhodnocení levého a pravého snímače. Pokud jeden nebo oba výsledky snímačů nevyhovují, nemůže být bod použit při konstrukci dilatometrické křivky. Výsledek určuje, zda bylo nalezeno těžiště dilatometrického terče. Enum obsahuje následující výčet stavů:

- Both = 0 – výsledky obou snímačů jsou validní
- Left = 1 – validní je pouze výsledek levého snímače
- Right = 2 – validní je pouze výsledek pravého snímače
- None = 3 – výsledky obou snímačů nejsou validní

**Třída Produkt**

| Název | Datový typ | Komentář |
|---|---|---|
| Number | string | Unikátní identifikátor |
| Name | string | Název |

Poslední důležitou položkou modelu je Enum Flag, který rozděluje jednotlivé snímače:

- Left = 0 – levý snímač
- Right = 1 – pravý snímač

## 5.4 WPF aplikace

Při spuštění aplikace dochází k registraci dílčích modulů Autofac kontejneru. Jeho funkčnost je vysvětlena v kapitole 3.6. Moduly jsou definovány v souboru ClientModule.

Na obr. 23 je zobrazen příklad registrace typu Singleton třídy MeasureModel, která obsahuje veřejné vlastnosti probíhajícího měření. Dále obsahuje ukázku registrace funkční třídy implementující rozhraní – zde definujeme třídu, jejíž instance bude vytvořena. Následuje ukázka registrace prezentačního modelu. Jako poslední je uvedena registrace základní třídy aplikace: MainWindow a datový kontext MainWindowViewModel. Jejich instance bude framework udržovat po celou dobu spuštění aplikace.

```
0 references | Martin Hortvík, 20 days ago | 1 author, 4 changes
protected override void Load(ContainerBuilder builder)
{
    // singleton instance

    builder.RegisterType<MeasureModel>().SingleInstance();

    // data managers

    builder.RegisterType<XmlProvider>()
        .As<XmlProvider, ILocalDataProvider>();

   // form viewmodels

   builder.RegisterType<FormViewModel>().AsSelf();

    // application context instance

    builder.RegisterType<MainWindowViewModel>().AsSelf()
        .InstancePerLifetimeScope();

    builder.RegisterType<MainWindow>().AsSelf()
        .InstancePerLifetimeScope();
}
```

*Obr. 23 Ukázka registrace tříd do kontejneru Autofac*

Datový kontext se skládá z prezentačních modelů. Pro každé okno nebo pohled aplikace z důvodu přehlednosti zakládáme nový. Funkce prezentačních modelů je vysvětlena v kapitole 3.3.6.

Prezentační model je tvořen třídou implementující rozhraní INotifyPropertyChanged obsahující delegáta události, která je vyvolána na základě změny člena modelu. Pokud změna pochází z uživatelského prostředí, generická metoda na základě reflexe automaticky vyvolá událost u toho členu modelu, kde proběhla změna. Pokud dojde ke změně z aplikační logiky, událost vyvoláme zavoláním generické metody s ukazatelem na konkrétního člena modelu. Poté dojde k modifikaci člena na straně uživatelského rozhraní.

Na obr. 24 je zobrazena definice člena prezentačního modelu s ukázkou volání generické metody OnPropertyChanged, která automaticky upraví hodnotu v uživatelském prostředí.

```
3 references | Martin Hortvík, 38 days ago | 1 author, 2 changes
private string name { get; set; }

3 references | Martin Hortvík, 41 days ago | 1 author, 1 change
public string Name
{
    get { return name; }
    set
    {
        name = value;
        measureModel.Measure.Name = name;

        OnPropertyChanged(() => Name);
    }
}
```

*Obr. 24 Definice člena prezentačního modelu vstupního formuláře*

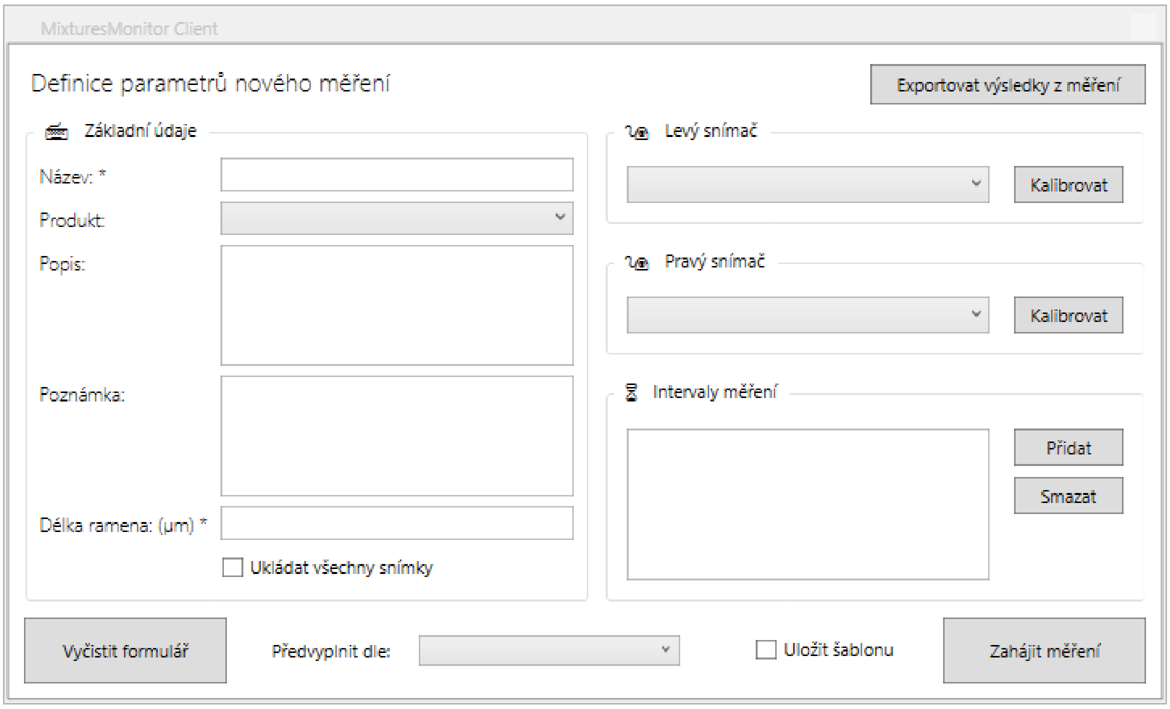
## 5.5 Vstupní formuláře

Při realizaci vstupního formuláře vycházíme z návrhu provedeném v kapitole 4.2. Datový kontext představuje prezentační model FormViewModel obsahující následující členy:

- vstupní parametry měření
  - textová pole
  - zaškrtávací políčka
  - výběr produktu – seznam produktů zprostředkován přes IServerProvider
- výběr snímačů – seznam zařízení zprostředkován přes ICameraManager
- ovládací tlačítka při definici intervalů
- obecné funkce
  - reset nastavených hodnot
  - předvyplnění formuláře – načtení uložených šablon přes ILocalDataProvider, možnost uložit šablonu.
  - tlačítko zahájení měření

K realizaci byla přidána funkce exportu výsledků již provedených měření. Slouží pro pracovníky obsluhy, kteří chtějí zpracovat výsledky z minulých měření přímo na experimentální stanici. Po kliknutí na tlačítko se otevře dialogové okno s výchozí složkou pro výběr datového XML souboru s naměřenými daty. Po výběru dojde ke zpracování výsledků přes ExportManager, jehož výstup tvoří excelový soubor s naměřenými daty a sestrojenými grafy.

Na obr. 25 je zobrazeno realizované okno s formulářovými prvky.

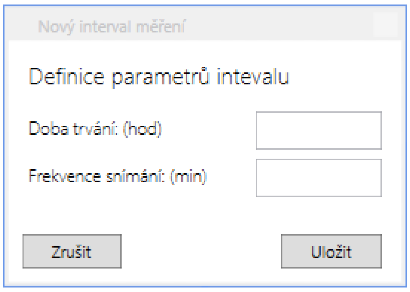



*Obr. 25 Realizovaný vstupní formulář k definici požadovaných parametrů měření*

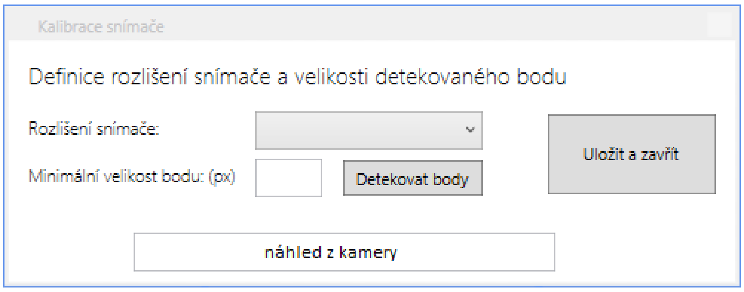
Definice intervalů probíhá přes dialogové okno s textovými poli pro zadání délky trvání intervalu a periody mezi snímky. Prezentačním modelem je IntervalViewModel. Na obr. 26.1 je zobrazeno realizované dialogové okno.

Vybrané snímače musí být před měřením zaostřeny a uvedeny do správné pozice vůči snímaným dilatometrickým terčům. Kliknutím na tlačítko Kalibrovat dojde k otevření dialogového okna s živým náhledem obrazového výstupu vybraného snímače. Při výběru z dostupných rozlišení dojde automatické změně náhledu. Zadáme minimální velikost detekovaného kruhu v pixelech (z důvodu různých rozlišení snímačů) a ověříme detekci kruhů v živém náhledu snímače. Funkce kalibračního okna jsou zprostředkovány přes CalibrationManager. Prezentačním modelem kalibračního okna je CalibrationViewModel.

Na obr. 26.2 je zobrazeno realizované dialogové okno. Náhled z kamery byl z důvodu ukázky zmenšen na malý proužek. Skutečný náhled z kamery bude zobrazen později.




*Obr. 26.1 Dialogové okno definice intervalu*

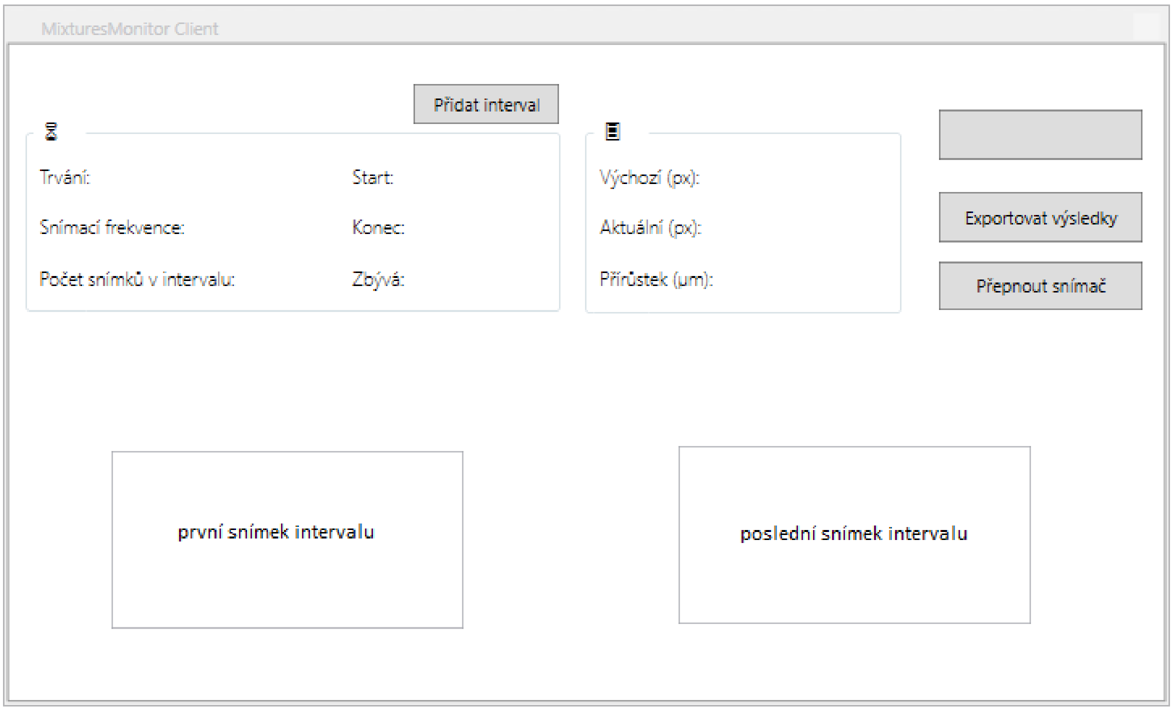



*Obr. 26.2 Dialogové okno kalibrace snímače*

## 5.6 Okno probíhajícího měření

Po spuštění měření se uživatelské prostředí přepne do okna, které zobrazuje detaily probíhajícího měření (viz obr. 27). Vzhled okna vychází z návrhu uvedeného v kapitole 4.6. Prezentačním modelem je MeasureViewModel, které obsahuje následující členy:

- Aktuální interval – zahrnuje všechny vlastnosti definující interval
- Výsledky v podobě souřadnic a relativních přírůstků polohy těžiště
- Náhledy – vlevo první a vpravo poslední snímek intervalu




*Obr. 27 Realizované okno s detaily probíhajícího měření*

## 5.7 Funkce

V následující části práce je popsána realizace funkcí zajišťující měření a zpracování výsledků desktopové aplikace.

### 5.7.1 Měření

Výchozí funkční třídou je MeasureManager, který obsahuje funkce k řízení procesu měření. Veřejné metody slouží k inicializaci, spouštění i přerušení měření.

Jádro tvoří privátní časovač běžící v samostatném vlákně. Obslužná metoda časovače obsahuje základní algoritmus, při kterém jsou pořizovány a zpracovávány snímky, aktualizována data v singleton instanci obsahující globální vlastnosti měření jako jsou aktuální interval a počet iterací. Součástí algoritmu je rozhodnutí, zda má časovač pokračovat a s jakým zpožděním má spustit další iteraci. Vychází přitom z definovaných intervalů měření. Pokud nezbývá žádný další interval, měření je dokončeno.

Sekundární složkou třídy je další časovač, který také běží v samostatném vlákně. S minutovou periodou zjišťuje, zda má v měření pokračovat, nebo jej ukončit na základě stavu měření serverového portálu. V případě, že bylo měření dokončeno, se dotazuje serveru, zda některý z uživatelů přidal nový interval měření. Pokud byl interval přidán, stanice jej uloží do paměti a bude pokračovat v měření.

V průběhu i po dokončení měření lze v programu přidat další interval, který bude zařazen na konec seznamu intervalů.

V okamžiku spuštění měření se nejprve provede ověření zadaných parametrů. V případě, že jsou v pořádku, program pokračuje inicializací vybraných snímačů. Pokud jsou připraveny, založíme adresářovou strukturu s názvem měření a uložíme hlavičku měření do lokální databáze. Poté bude zaslána na server. Až po těchto úkonech dojde ke startu časovače.

### 5.7.2 Kamera

Funkční třída osahující metody pro manipulaci s obrazovým výstupem snímacích zařízení je CameraManager. Použitá knihovna AForge.Video, která byla představena v kapitole 3.5, nabízí metody k plné kontrole video zařízení na sběrnici USB.

K výběru snímačů v uživatelském prostředí slouží seznam všech dostupných video zařízení připojených k počítačové stanici. Metoda Get vrací seznam těchto zařízení.

Před spuštěním měření musí dojít k inicializaci obou vybraných snímačů. K tomu slouží metoda Initialize, která vrací hodnotu True při úspěšné a False při neúspěšné inicializaci kamer. Příprava video zařízení spočívá ve vytvoření instance třídy VideoCaptureDevice, které nastavíme veřejné vlastnosti a přihlásíme se k odběru události při příjmu obrazových dat delegátem NewFrameEventHandler. V obslužné metodě zpracujeme příchozí data.

Algoritmus měření volá metodu Snap, která zajistí pořízení snímků a vrací instanci třídy SnapResult, která obsahuje snímky a detekované kruhy z obou snímačů volám metody Detect třídy DetectionManager. Při pořizování snímku jej rovnou ukládáme na disk do připravené složkové struktury. Kořenový adresář měření obsahuje složky Left a Right, do kterých jsou jednotlivé snímky ukládány. Názvy souborů závisí na zvoleném způsobu ukládání, které může být dvojího typu:

- Ukládat všechny snímky – název je složen z pořadového čísla intervalu v hranatých závorkách a globálního čísla snímku oddělených podtržítkem.

- Ukládat pouze první a poslední snímek intervalu – první snímek měření je pojmenován číslem 1 v hranatých závorkách a slovem first odděleným podtržítkem. Další snímky jsou přepisovány stejným názvem složeným z pořadového čísla intervalu v hranatých závorkách a slova last odděleným podtržítkem. Poslední snímek intervalu totiž představuje první snímek dalšího intervalu.

V průběhu testování činila průměrná velikost jednoho snímku 1,5 Mb. Při ukládání pouze na disk by to nepředstavovalo problém, ale při zasílání snímků na server by došlo ke značnému zatížení síťové komunikace přenosem objemných dat. Z toho důvodu použijeme obrazový enkodér, který nám sníží velikost souboru při relativním zachování kvality. Po snížení kvality je průměrná velikost snímku cca 150 Kb. Tato hodnota je přibližná, protože vždy záleží na výběru snímače a dostupného rozlišení.

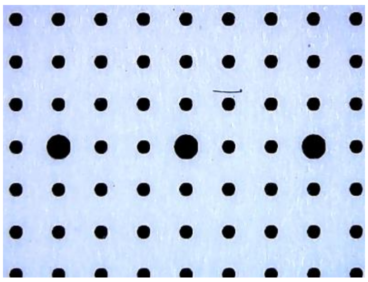
Po ukončení měření nebo uzavření aplikace zavoláme metodu Close, která bezpečně uvolní objekty video zařízení.

### 5.7.3 Detekce

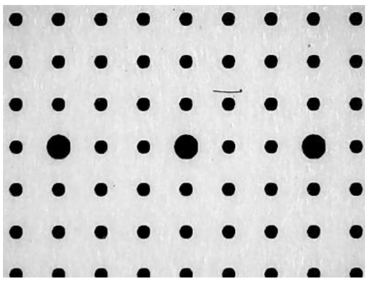
Detekci kruhů na snímku zajišťuje třída DetectionManager, která obsahuje metodu Detect, jenž přijímá bitmapu (data snímku) a prahovou hodnotu detekce kruhu v pixelech a vrací seznam detekovaných kruhů. Metody pro manipulaci s bitmapou jsou zajištěny knihovnou AForge.Imaging. O následnou detekci tvarů ve vhodně upravené bitmapě se postarají metody knihovny AForge.Math.Geometry.

Výchozím objektem je snímek dilatometrického terče, který obsahuje tři větší tečky uprostřed pole malých teček. Centrální tečka určuje těžiště terče. Její souřadnice chceme získat.

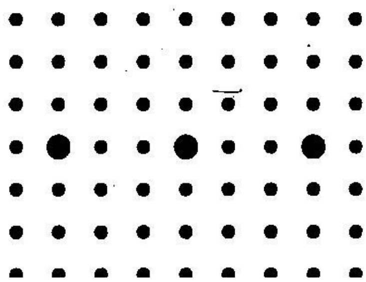
Prvním krokem je úprava originální bitmapy (viz obr. 28.1) redukcí barevných složek (převodem do stupňů šedi) funkcí Grayscale. Na bitmapu ve stupních šedi (viz obr. 28.2) dále aplikujeme filtr BradleyLocalThresholding, který bitmapu binarizuje do složek černé a bílé (viz obr. 28.3). Odstraníme tím barevný šum a zvýrazníme přechod mezi černou a bílou.



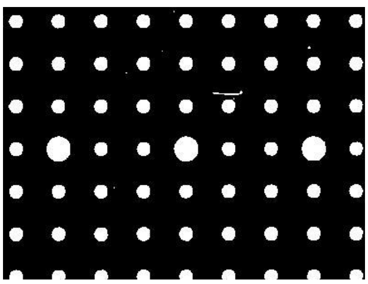
*Obr. 28.1 Originál*



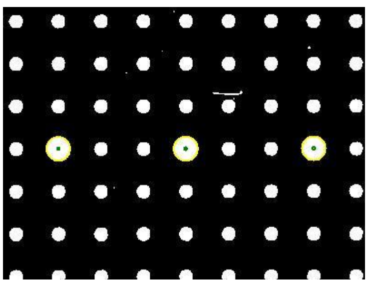
*Obr. 28.2 Stupně šedi*



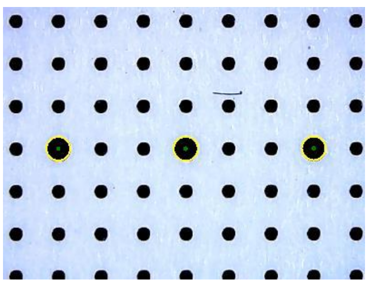
*Obr. 28.3 Binarizování*



*Obr. 28.4 Invertování*



*Obr. 28.5 Detekování*



*Obr. 28.6 Výsledek*

Dalším krokem je invertování barev (viz obr. 28.4). Takto upravená bitmapa je připravena pro detekci tvarů. Pro každý snímač definujeme prahovou velikost detekovaného kruhu v pixelech. Při změně rozlišení proto musíme tuto hodnotu adekvátně upravit. Ukázková detekce na obr. 28.x používá snímek o rozlišení 400x300 pixelů a prahová hodnota pro detekci centrálních teček je 25 pixelů.

K detekci tvarů je použita třída BlobCounter, která vrací pole objektů. Z nich jsou vybrány pouze ty, které splňují kruhovitost o definované velikosti. Vybrané objekty jsou z metody vráceny v podobě seznamu. K výpočtu souřadnic a vykreslení bodů se tato třída nepoužívá. Na obr. 28.5 a 28.6 jsou vykresleny detekované body pomocnou třídou ImageHelper.

### 5.7.4 Aplikace

Funkční třída AppManager obsahuje metody zajišťující širokou škálu funkcí. Používá se k ovládání uživatelského prostředí, jako je aktualizace intervalu, souřadnic, snímků a popisků. Dále provádí validaci zadaných parametrů měření. Do lokální a vzdálené databáze ukládá hlavičku měření, intervaly a body.

Kromě výše uvedených funkcí se také stará o zpracování detekovaných snímků. Výpočet je prováděn pouze u těch snímků, které obsahují přesně 3 detekované body. Z těchto bodů vypočítáme absolutní souřadnice středové tečky v pixelech. Jelikož známe vzdálenosti teček v µm, můžeme vzdálenosti v pixelech přepočítat do µm. Princip výpočtu je popsaný v kapitole 2.6. Po vyhodnocení obou snímků je vytvořena instance bodu, do které jsou uloženy všechny vypočítané hodnoty. Bod je následně uložen do lokální a serverové databáze.

### 5.7.5 Kalibrace

Metody k ovládání dialogového kalibračního okna jsou součástí třídy CalibrationManager. Vybraný snímač je při otevření okna inicializován a nabízí v seznamu dostupná rozlišení. Při změně rozlišení se automaticky změní náhled dle vybrané položky. Na obr. 29 je zobrazeno dialogové okno s živým náhledem a detekcí teček dilatometrického terče.


*Obr. 29 Živý náhled vybraného snímače s detekcí teček dilatometrického terče*

## 5.8 Data

Aby bylo možné k výsledkům měření přistupovat i po ukončení aplikace, je nutné naměřená data uložit na disk. Všechna data budou v průběhu měření zasílána na server, kde budou uložena v databázi a na disku.

### 5.8.1 Lokální databáze

Již výše zmíněná adresářová struktura se skládá z kořenového adresáře MixturesMonitor ve sdílených dokumentech. Obsahuje podsložky s názvy provedených měření, datový soubor se šablonami a textový soubor určený k výpisu případných výjimek aplikace.

K uchování informací je z hlediska rychlosti vývoje i přístupu je nejvhodnější použití ukládání do souborů XML. Metody k zápisu a výběru dat jsou obsaženy v třídě XmlProvider. Pro práci s XML dokumenty je použita knihovna XDocument.

Struktura XML dokumentu obsahuje kořenový element MeasureData, do kterého jsou vloženy následující elementy:

- *Measure* – hlavička měření (parametry, vybrané snímače)
- *Interval* – definice intervalu
- *Point* – bod s vypočtenými výsledky

Jednotlivé elementy obsahují informace v podobě atributů, které odpovídají datovému modelu navrhnutém v kapitole 5.3. Datový soubor je umístěn ve složce s název měření.

### 5.8.2 Vzdálená databáze

Komunikaci desktopové aplikace a serverové části zajišťuje třída ServerProvider. Obsahuje asynchronní metody pro posílání HTTP požadavků a přijímání dat ze serverového aplikačního rozhraní API.

Základem všech požadavků je instance třídy HttpClient, které předáme URL adresu. Přenášená data budou v JSON formátu. Do hlavičky požadavků proto vložíme typ dat application/json.

Z důvodů zajištění bezpečnosti přenosu dat na server, přidáme do hlavičky požadavku SecurityHash, který se musí shodovat s hashem na serveru. V případě, že nebude požadavek tento hash obsahovat, nebude možné se dostat do obslužných metod ani databáze serveru. Hashem je datum ve formátu rok – měsíc, slovo MixturesMonitor a předem náhodně vygenerovaný řetězec R39hGarm. Tento text převedeme na hash metodou SHA256 v pomocné třídě SecurityHelper.

Přenos snímků zabezpečuje Reference na webovou službu Monitor.TransferService. Ta vyžaduje definici modelu, který bude zasílán. Obsahuje bitmapu, název souboru, ID měření a složku, do které bude snímek uložen. Obslužná metoda serverové části pouze ukládá bitmapu na disk dle doručených informací.

## 5.9 Export

Metody pro exportování naměřených dat jsou definovány ve třídě ExportManager. Export dat lze provést jak z aktuálně probíhajícího měření, tak z již ukončeného (výběr datového souboru přes dialog). Výsledkem je excelový soubor s naměřenými daty a zkonstruovanými grafy.

Jednotlivé části, ze kterých je složen výsledný soubor, jsou definovány v pomocné třídě ExportHelper. Export na straně serveru pracuje stejně, pouze s tím rozdílem, že data bere z databáze MSSQL. Výsledný soubor je rozdělen na tři listy:

- Data měření – obsahuje pouze validní body (levé i pravé), ze kterých jsou zkonstruovány spojnicové grafy.
- Left data – obsahuje výsledky pouze z levého snímače
- Right data – obsahuje výsledky pouze z pravého snímače

V záhlaví listů je uveden název měření a datum vytvoření exportu. První list obsahuje sloupce dle požadavků zadavatele uvedené v kapitole 2.6:

- Pořadí intervalu
- Datum a čas
- Čas v hodinách (pro osu X)
- Data osy X
    - Levé relativní přírůstky v µm
    - Pravé relativní přírůstky v µm
    - Průměr přírůstků v µm
    - Relativní přetvoření v m/m
- Velikost pixelu v µm
- Data osy Y
    - Levé relativní přírůstky v µm
    - Pravé relativní přírůstky v µm
    - Průměr přírůstků v µm

Zbylé listy obsahují všechna data (i nevalidní) z obou snímačů v osách X a Y.

Z relativních přírůstků a přetvoření osy X jsou sestrojeny jednotlivé grafy. Práci s excelovými soubory zajišťuje knihovna EPPplus dle návrhu v kapitole 4.2. Obsahuje metody k manipulaci s hodnotami uvnitř buněk, práci s rovnicemi a vynášení dat do grafů.

# 6 REALIZACE SERVEROVÉ APLIKACE

Při realizaci serverové části řešení vycházíme z návrhu provedeném v kapitole 4.3.

## 6.1 Základní struktura

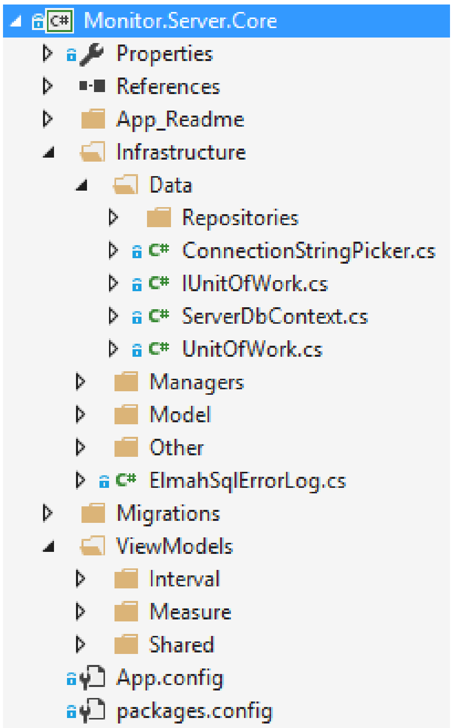
Serverová část je dle návrhu složena ze tří dílčích částí. Dvě z nich sdílí připojení do databáze a datový model. Z toho důvodu oddělíme společné části do samostatného projektu. Vzájemné závislosti vychází z návrhu základní struktury v kapitole 4.1. V řešení byly založeny následující projekty:

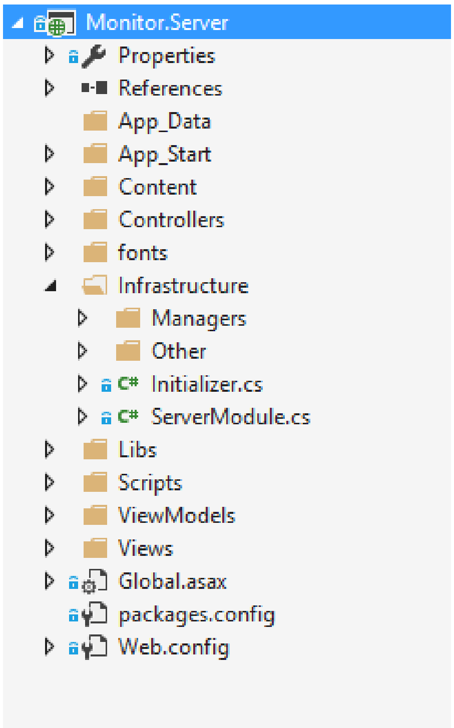
- *Monitor.Server.Core* – sdílená knihovna, která obsahuje datový model a zabezpečuje připojení do databáze. Dále obsahuje funkční třídy s metodami pro manipulaci s datovým modelem. Obsahuje závislost na sdílené knihovně *Monitor.Common*. Struktura projektu je zobrazena na obr. 30.1.

- *Monitor*.Server.Api – webové ASP.NET Web API aplikační rozhraní, které tvoří komunikační vrstvu mezi desktopovou aplikací a serverovou databází. Obsahuje závislosti na sdílených knihovnách *Monitor.Common* a *Monitor.Server.Core*. Struktura projektu je ukázána na obr. 30.3.

- *Monitor*.Server – webová ASP.NET aplikace s architekturou MVC, která slouží jako prezentační webové stránky pro interakci s uživateli. Je závislá na sdílených knihovnách *Monitor.Common* a *Monitor.Server.Core*. Struktura viz obr. 30.2.

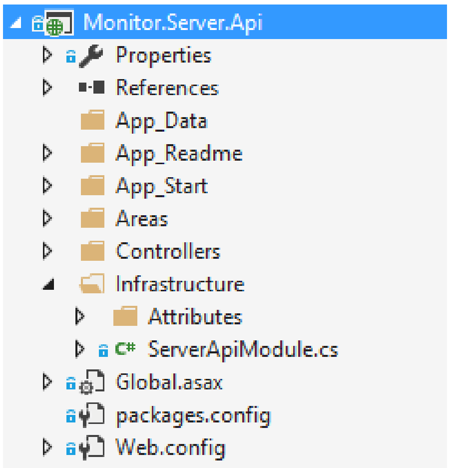
- *Monitor.TransferService* – webová WCF služba hostovaná na serveru, která zajišťuje upload snímků přes HTTP protokol. Obsahuje závislost na sdílené knihovně *Monitor.Common*. Struktura projektu je zobrazena na obr. 30.4.



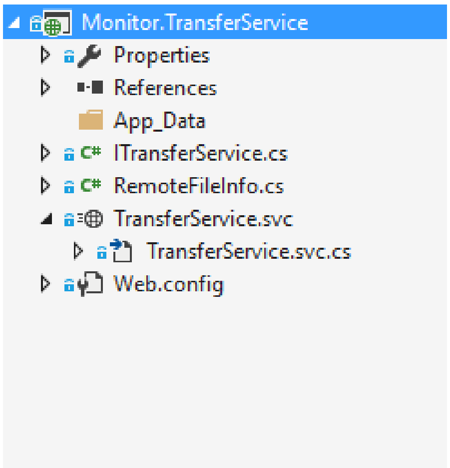




*Obr. 30.1 Struktura projektu Monitor.Server.Core*

*Obr. 30.2 Struktura projektu Monitor.Server*




*Obr. 30.3 Struktura projektu Monitor.Server.Api*




*Obr. 30.4 Struktura projektu TransferService*

## 6.2 Sdílená serverová knihovna

Velké množství funkcionalit je sdíleno mezi oběma webovými ASP.NET projekty. Jedná se hlavně o definici datového kontextu, který by musel být definován pro každý projekt zvlášť. Sktruktura knihovny je skládá ze složek:

- *Infrastructure* – obsahuje definici datového modelu, repozitáře a funkční třídy
- *Migrations* – slouží pro ukládání migračních tříd definující změny databáze
- *ViewModels* – obsahuje sdílené prezentační modely

### 6.2.1 Datový model

Datový model serverových projektů je tvořen třídami, které budou sloužit jako datové entity pro mapování tabulek databáze. Využíváme přitom souhrnu knihoven Entity framework, který byl vysvětlen v kapitole 3.7. Na základě datového modelu vygenerujeme logické schéma databáze (scénář zobrazen na obr. 16.2).

Definice datového modelu obsahuje následující třídy:

**Třída ActivityLog**

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Comment | String | Obsah sdělení (výpis změn) |
| Operation | Enum | Typ operace |
| Type | Enum | Typ modulu |
| TypeId | int | Primární klíč modulu |
| UserId | int | Primární klíč uživatele |
| CreationDateTime | DateTime | Datum a čas vytvoření |

Výsledná tabulka obsahuje záznamy všech provedených akcí a změn na serverovém portálu. Primárně slouží k uchování historie a dohledatelnosti příčin různých problémů. Typ operace je výčet následujících akcí:

- Create = 0 – záznam vytvořen; Read = 1 – záznam přečten
- Update = 2 – záznam upraven; Destroy = 3 – záznam odstraněn

**Třída Interval**

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Order | int | Pořadí |
| Count | int | Počet iterací |
| Counted | int | Průběh iterací (napočítáno) |
| Duration | decimal | Trvání |
| Ticks | decimal | Perioda |
| State | Enum | Stav |
| MeasureId | int | Cizí klíč – Measure |
| CreationDateTime | DateTime | Datum a čas vytvoření |

Stavy intervalu jsou definovány výčtem IntervalState, který obsahuje následující položky:

- Pending = 0 – čekající; Current = 1 – aktivní
- Finished = 2 – dokončen; Extra = 3 – extra; Aborted = 4 - přerušen

**Třída Measure**

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Number | string | Unikátní číslo měření |
| Name | string | Název |
| Description | string | Popis |
| Note | string | Poznámka |
| State | Enum | Stav |
| LeftResolutionName | string | Název rozlišení levého snímače |
| RightResolutionName | string | Název rozlišení pravého snímače |
| LeftCameraName | string | Název levého snímače |
| RightCameraName | string | Název pravého snímače |
| SaveAll | bool | Ukládat všechny snímky |
| TargetDist | decimal | Délka ramene |
| DateTime | DateTime | Datum a čas spuštění |
| CreationDateTime | DateTime | Datum a čas vytvoření |
| LastChangeDateTime | DateTime | Datum a čas poslední úpravy |
| ProductId | int | Cizí klíč – Product |

Stav měření může být jedním z následujícího výčtu MeasureState:

- Running = 0 – probíhající; Completed = 1 – dokončeno; Aborted = 2 – ukončeno

**Třída Module**

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Name | string | Název |
| Type | Enum | Typ |
| ControllerName | string | Název kontroleru |

Typem modulu může být: All = 0 – přístup všech; Admin = 1 – přístup pouze administrátor

### Třída Point

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Timestamp | DateTime | Datum a čas snímku |
| GlobalTicks | int | Globální pořadí |
| LeftAbsX | decimal | Absolutní souřadnice X levého středu |
| LeftAbsY | decimal | Absolutní souřadnice Y levého středu |
| LeftRelX | decimal | Relativní přírůstek X levého středu |
| LeftRelY | decimal | Relativní přírůstek Y levého středu |
| LeftDistPx | decimal | Velikost pixelu levého snímku |
| RightAbsX | decimal | Absolutní souřadnice X pravého středu |
| RightAbsY | decimal | Absolutní souřadnice Y pravého středu |
| RightRelX | decimal | Relativní přírůstek X pravého středu |
| RightRelY | decimal | Relativní přírůstek Y pravého středu |
| RightDistPx | decimal | Velikost pixelu pravého snímku |
| Valid | Enum | Validita bodu |
| CreationDateTime | DateTime | Datum a čas vytvoření |
| ProductId | int | Cizí klíč – Product |
| MeasureId | int | Cizí klíč – Measure |
| IntervalId | int | Cizí klíč – Interval |

### Třída Product

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Number | string | Unikátní číslo |
| Name | string | Název |
| Description | string | Popis |
| Note | string | Poznámka |
| Published | bool | Viditelný |
| CreationDateTime | DateTime | Datum a čas vytvoření |
| LastChangeDateTime | DateTime | Datum a čas poslední úpravy |

### Třída QueuedEmail

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| From | string | Emailová adresa odesílatele |
| FromName | string | Název odesílatele |
| To | string | Emailová adresa příjemce |
| ToName | string | Název příjemce |
| Subject | string | Předmět |
| Body | string | Tělo zprávy |
| SentDateTime | DateTime? | Datum a čas odeslání |
| CreationDateTime | DateTime | Datum a čas vytvoření |
| LastChangeDateTime | DateTime | Datum a čas poslední úpravy |
| UserId | int? | Cizí klíč – User |

**Třída Role**

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Name | string | Název |
| SystemName | string | Systémový název |
| Description | string | Popis |

**Třída RolePermission**

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Permissions | Enum | Přidělená oprávnění |
| RoleId | int | Cizí klíč – Role |
| ModuleId | int | Cizí klíč – Modul |

**Třída User**

| Název | Datový typ | Komentář |
|---|---|---|
| Id | int | Primární klíč |
| Username | string | Unikátní login |
| Password | string | Heslo v podobě hashe |
| PasswordSalt | string | Sůl hesla |
| Firstname | string | Jméno |
| Lastname | string | Příjmení |
| Email | string | Emailová adresa |
| PhoneNumber | string | Telefonní číslo |
| Occupation | string | Zaměstnání |
| Residence | string | Bydliště |
| Note | string | Poznámka |
| UserType | Enum | Typ (Admin, Technik) |
| Published | bool | Aktivní |
| RoleId | int | Cizí klíč – Role |
| CreationDateTime | DateTime | Datum a čas vytvoření |
| LastChangeDateTime | DateTime | Datum a čas poslední úpravy |

### 6.2.2 Databázový kontext

Po definici datových entit byla vytvořena databáze pomocí příkazů konzole Visual Studia:

- *add-migration* [název migrace] – vygeneruje kód, který bude aplikován na SQL server.
- *update-database* – aplikuje všechny čekající migrace na zvolenou databázi.

Databázi zvolíme v konfiguračním souboru pomocí připojovacího řetězce ConnectionString. Pro každou třídu vytvoříme repozitář a zabalíme do třídy UnitOfWork podle vzoru uvedeného v kapitole 3.8. Tímto máme připraven datový kontext, který můžeme vkládat do konstruktorů tříd, kde jej chceme využívat. Pomůže nám v tom již dříve použitý framework Autofac.

## 6.3 Webové aplikační rozhraní

Struktura webového aplikačního rozhraní vychází z návrhu uvedeného v kapitole 4.3.1.

Základem jsou API třídy ve složce Controllers, které jsou přímými potomky třídy ApiController. Prvním krokem ke zprovoznění je definice URL šablony, podle které bude rozhraní obsluhovat požadavky. Tvar použité standartní šablony je zobrazen na obr. 31.

```
1 reference | Martin Hortvík, 22 days ago | 1 author, 2 changes
public static void Register(HttpConfiguration config)
{
    // Web API routes
    config.MapHttpAttributeRoutes();

    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
    );
}
```

*Obr. 31 Definice URL šablon API rozhraní*

Pro úspěšné vyvolání metody kontroleru musí být URL adresa ve tvaru:

api/název kontroleru/případné parametry

### 6.3.1 Kontrolery

K realizaci navržených funkcí byly založeny následující kontrolery:

- *AppController*
  - GET – vrací pravdivostní hodnotu, zda nechat měření na stanici běžet
  - POST – měření na stanici bylo zrušeno, nastav stav v databázi
- *IntervalsController*
  - GET – vrací extra interval, pokud v databázi existuje – stanice bude pokračovat v měření
  - POST – na stanici byl přidán extra interval, obnov stav měření na probíhající
- *MeasuresController*
  - POST – založí měření v databázi na základě přijatých parametrů
- *PointsController*
  - POST – založí bod měření v databázi
- *ProductsController*
  - GET – vrací seznam produktů v databázi

Všechny uvedené kontrolery obsahují řídicí atribut RequireSecurityHash, který před přístupem do třídy zjistí, zda HTTP požadavek obsahuje hlavičku SecurityHash se správnou hodnotou. V případě, že ji neobsahuje, obslužný kód třídy nebude vyvolán.

Přijímané intervaly a body ze stanice musí obsahovat Id měření, které tvoří cizí klíč těchto záznamů. K tomu slouží hlavička HTTP požadavku MeasureId.

Obslužné metody POST vrací HTTP odpověď ve formě status kódu, který má pro úspěšnou akci hodnotu 200 OK, pro vytvoření záznamu hodnotu 201 Created a pro neúspěšnou akci hodnotu 400 BadRequest.

### 6.3.2 Funkce

Kontrolery neobsahují žádnou vnitřní logiku, ani přístup k databázi – k tomu je určena funkční vrstva tvořena třídami, jejichž metody z kontroleru voláme. Pokud to není nutné, vkládáme funkční třídy do sdílené serverové knihovny, abychom k nim měli přístup i z prezentační webové aplikace. Mezi zde využívané funkční třídy patří:

- *IntervalManager* – obsahuje funkce ke čtení a vytváření entity intervalu v databázi.
- *MeasureManager* – slouží k široké manipulaci s entitou měření v databázi.
- *PointManager* – ukládá přijaté body do databáze.

Pokud dojde při zpracování požadavku k výjimce, je třeba mít přehled o původu chyby a na základě něj kód vyladit. K tomu byla použita knihovna Elmah, která při správné implementaci zachytává a ukládá veškeré informace o vyvolané výjimce. Ukládá je do vlastní databáze, kterou vytvoříme SQL scriptem před publikováním projektu. Výpis chyb je přístupný na adrese /elmah.axd.

## 6.4 Webová prezentační aplikace

Realizace webové aplikace probíhala dle návrhu uvedeném v kapitole 4.3.2. Použité technologie při vývoji webových stránek ASP.NET jsou popsány v kapitole 3.4.

Struktura aplikace vychází z MVC vzoru, kdy máme kontrolery ve složce *Controllers*, pohledy ve složce *Views* a modely definované funkčními třídami ve složce *Infrastructure*. Ostatní složky kořenového adresáře projektu tvoří:

- *Content* – obsahuje obrázky, ikony a CSS soubory (design stránek)
- *fonts* – souhrn fontů
- *Libs* – obsahuje Javascriptové frameworky, knihovny
- *Scripts* – základní a vlastní Javascriptové soubory
- *ViewModels* – prezentační modely

Přistoupit do portálu lze pouze s platným přihlášením, proto v aplikaci nastavujeme autentifikaci. Do konfiguračního souboru přidáme metodu autentifikace Forms. Při každém požadavku aplikace zjistí, zda je uživatel ve svém prohlížeči přihlášen na základě hodnoty cookies. V aplikaci pomocí atributu AllowAnonymous umožníme vstup do metody i když není uživatel přihlášen – vstupní metodě Login v kontroleru Account.

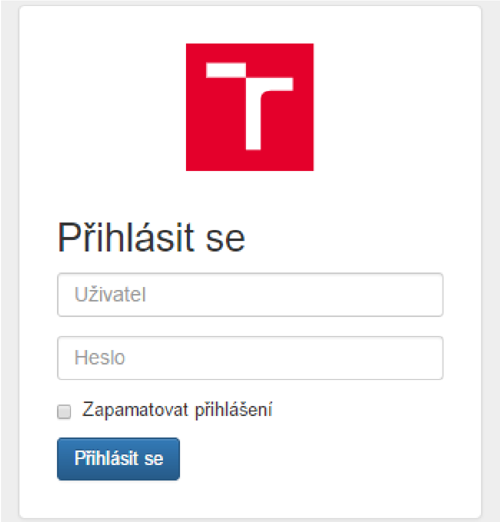
Webové stránky jsou dle návrhu rozděleny do jednotlivých modulů.

### 6.4.1 Modul Obecný

Obecné stránky portálu jsou přístupné všem typům uživatele. Patří mezi ně tyto stránky:

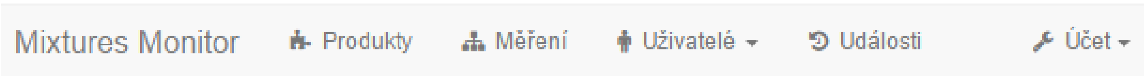
**Přihlášení**
Stránka obsahuje logo a formulář s poli Uživatel, Heslo a Zapamatovat přihlášení. Při zadání nevalidních údajů je uživatel upozorněn, že zadává špatné údaje. Přihlašovací stránka je zobrazena na obr. 32. Po přihlášení je uživatel přesměrovn na úvodní stranu.

Přihlašovací formulář také odešle parametry v adresním řádku, kde může být vyplněna adresa, na kterou bude uživatel po úspěšném přihlášení přesměrován – tzv. ReturnUrl.




*Obr. 32 Přihlašovací formulář portálu*

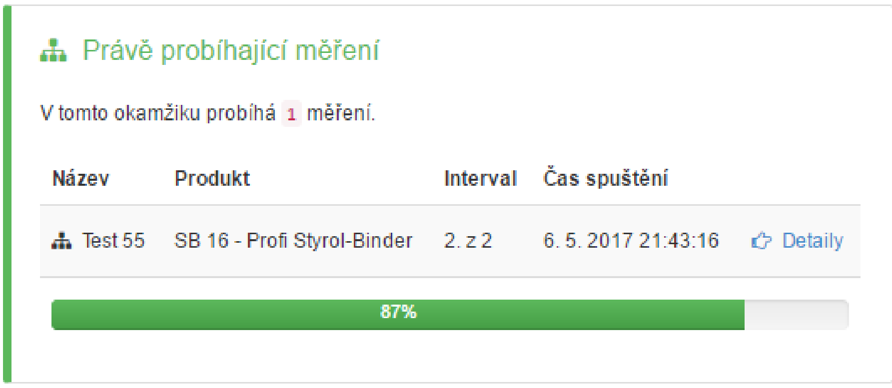
Vnitřní stránky obsahují hlavní menu v horní části okna. Obsahuje odkazy do jednotlivých modulů portálu. Hlavní menu je zobrazeno na obr. 33.




*Obr. 33 Hlavní navigace portálu*

**Úvodní strana**

Na úvodní straně se nachází panely s navigací na moduly Produkty, Měření a Uživatelé. V pravé části je informační panel právě probíhajících měřeních (viz obr. 34). Pokud žádné neprobíhá, panel je označený rudým okrajem a zůstává prázdný.




*Obr. 34 Informační panel s aktuálně probíhajícím měřením Test 55*

**Události**

Na stránce je seznam, který obsahuje všechny akce a změny, které se v portálu udály. Má pouze informativní charakter. Historicky lze dohledat, který uživatel je za co zodpovědný.

**Můj profil a změna hesla**

Samostatné stránky obsahují jednoduché formuláře. Změna hesla vyžaduje zadání původního a nového i s potvrzením. Můj profil obsahuje základní informace o přihlášeném uživateli s možností editace základních údajů.

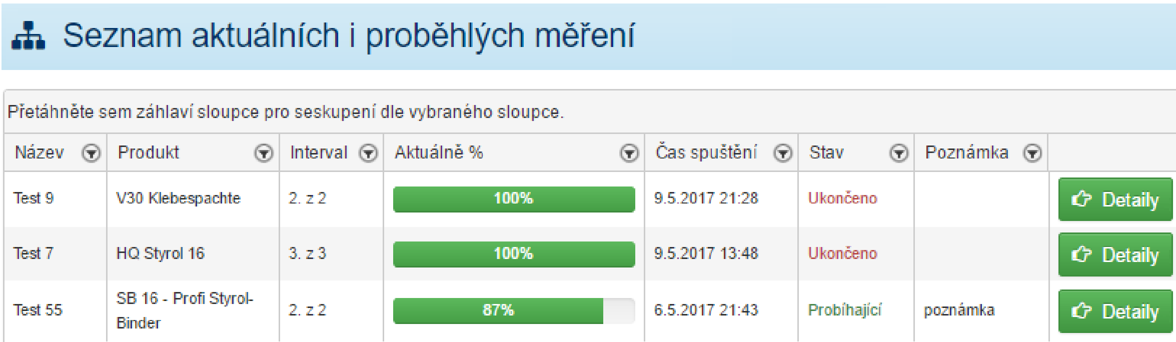
### 6.4.2 Modul Produkty

Modul tvoří samostatná stránka s interaktivním seznamem produktů. S položkami v seznamu lze manipulovat (přidávat, upravovat i mazat) dle přiřazených oprávnění. Seznam je realizován v Javascriptovém frameworku Kendo UI, který nabízí sestavení widgetu přímo v syntaxi Razor (syntaxe dynamických prvků v pohledech aplikace).

Výběr produktu je jedním z parametrů při spouštění měření na stanici. Takto má obsluha přehled o tom, jaké produkty byly zkoumány.

### 6.4.3 Modul Měření

Modul měření je hlavní částí serverového portálu. Skládá se ze seznamu a samostatné stránky s detaily pro každé měření. V seznamu jsou uvedeny základní informace, k detailu se uživatel dostane po kliknutí na tlačítko Detaily (viz obr. 35).




*Obr. 35 Seznam měření se základními informacemi a odkazem na detaily*

Karta měření je rozdělena na jednotlivé podstránky přepínatelné navigačními tlačítky.

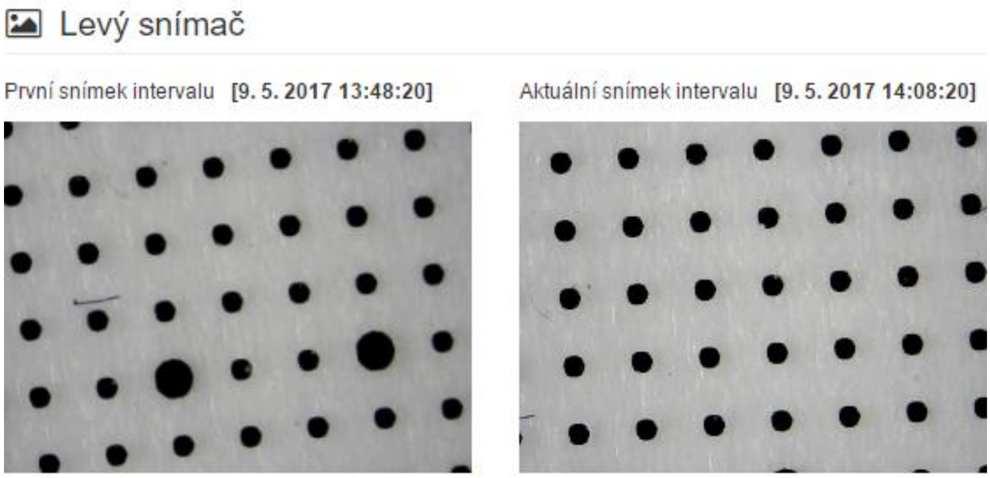
**Podstránka Detaily**

Obsahuje všechny podstatné informace měření. Aktuální stav měření v procentech a informace o aktuálním intervalu (viz obr. 36). V prostoru mezi nimi se nachází ovládací tlačítka. Jejich zobrazení záleží na stavu, v jakém se měření nachází. V probíhajícím stavu lze měření přerušit nebo si nechat exportovat dílčí výsledky.

V další části jsou zobrazeny zadané parametry měření s poznámkou, kterou může uživatel editovat. Editace poznámky je realizována přes Javascriptovou technologii AJAX.


*Obr. 36 Aktuální stav a interval měření, ovládací tlačítka na stránce Detaily*

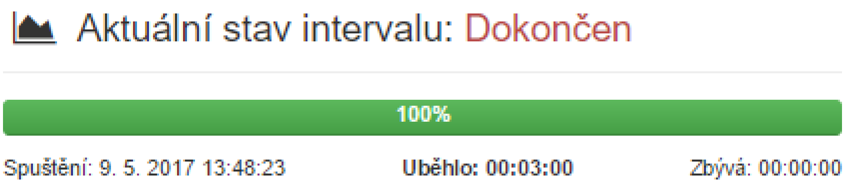
Snímky z levého a pravého snímače jsou zobrazeny ve spodní části okna (viz obr. 37). Vedle nich jsou aktuální výsledky absolutní polohy středu dilatometrického terče a relativní přírůstky v µm. Prostor pod výsledky obsahuje název a rozlišení snímače.




*Obr. 37 Snímky z levého snímače na stránce Detaily (obdobně i pravé snímky)*

**Podstránka Intervaly**

Slouží k vizuálnímu přehledu stavu jednotlivých měřících intervalů. Pro každý interval jsou vypsány informace (stejné jako na obr. 36) a navíc s procentuálním stavem (viz obr. 38).




*Obr. 38 Procentuální stav intervalu*

**Podstránky Levé a Pravé snímky**

Obsahují snímky seskupené dle intervalů měření. Snímky lze kliknutím otevřít v modálním okně a listovat pomocí tlačítek nebo kolečkem myši. Funkce náhledu byla realizována přes populární Javascriptovou knihovnu Fancybox. Pod každým snímkem je datum a čas vytvoření.

**Podstránka dokumenty**

Slouží k nahrávání příloh týkajících se konkrétního měření. Skládá se z interaktivního seznamu, který zobrazuje soubory ve složce, která byla vytvořena při založení měření. Do ní lze přes grafický uploader nahrávat libovolné přílohy. K realizaci uploaderu byla použita Javascriptová knihovna plupload.

**Podstránka Historie**

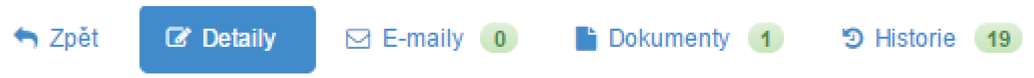
Obsahuje seznam všech akcí a změn, které se týkají zobrazeného měření. Například zda byl vytvořen nový interval, uložen nový bod nebo bylo měření zrušeno z experimentální stanice.

### 6.4.4 Modul Uživatelé

K následujícímu modulu může přistupovat pouze administrátor. Slouží ke správě uživatelských účtů, které mají přístup do portálu na základě ověření totožnosti zadáním přihlašovacích údajů.

První realizovanou stránkou je formulář Přidat nového uživatele, který obsahuje pole k vyplnění všech položek datového modelu User (vlastnosti uvedeny v kapitole 6.2.1). Před odesláním formuláře lze zaškrtnout volbu Odeslat přihlašovací údaje. Po založení uživatelského účtu bude odeslán email s přihlašovacími údaji na uvedený email.

Další stránka obsahuje seznam uživatelů portálu, který obsahuje základní informace. Přístup ke všem detailům včetně editace poskytuje samostatná stránka Detaily. Karta je rozdělena obdobně jako karta měření na jednotlivé podstránky, které jsou dostupné z navigace (viz obr. 39).




*Obr. 39 Navigace podstránek v detailech uživatelského účtu*

**Podstránka detaily**

Obsahuje editační formulář složený z jednotlivých polí. Vedle něj se nachází panel k nastavení hesla. Po kliknutí na tlačítko Nastavit heslo se objeví pole na zadání nového hesla. Pod polem se je tlačítko, po jehož stisku bude vygenerován náhodný řetězec znaků. Další tlačítko slouží k potvrzení změny. Odeslání hesla na server je zajištěno Javascriptovou technologií AJAX. Taktéž lze zaškrtnutím volby odeslat přihlašovací údaje emailem. Podstránka Detaily je zobrazena na obr. 40.


*Obr. 40 Editační formulář uživatelského účtu s nastavením hesla*

**Podstránka E-maily**
Obsahuje seznam odeslaných emailů na adresu uživatele. Sloupec Odesláno informuje o tom, zda došlo k úspěšnému odeslání emailu. Po najetí myší do buňky s obsahem zprávy bude tělo emailu zobrazeno v náhledovém okénku (widget Tooltip).

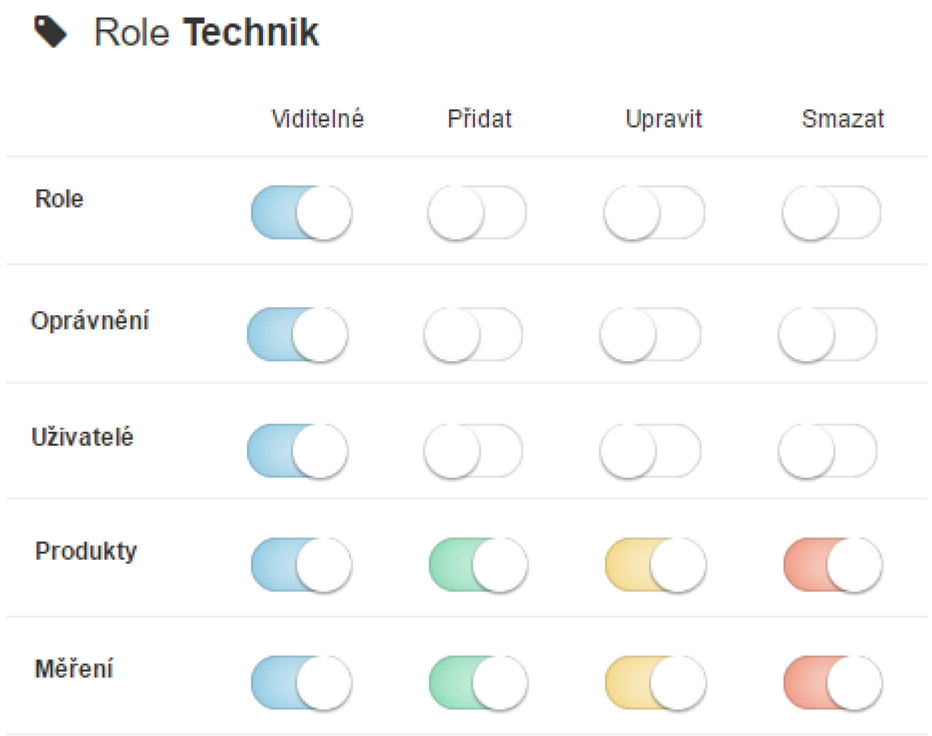
**Podstránka Dokumenty**
Podobně jako u detailu měření lze nahrávat přílohy do vytvořené složky při založení uživatele.

**Podstránka Historie**
Obsahuje seznam všech akcí a změn, které uživatel v portálu provedl.

### 6.4.5 Modul Role a oprávnění

Pomocí modulu Role a oprávnění přiřazuje administrátor uživatelským účtům práva, pomocí kterých mohou v portálu přistupovat a modifikovat položky jednotlivých modulů. Základem je interaktivní seznam, kde administrátor provádí správu rolí.

Další částí je stránka Správa oprávnění. Ta je realizována výběrem role a výpisem modulů portálu v řádcích s jednotlivými právy ve sloupcích. Na obr. 41 je zobrazena tabulka udělování oprávnění role Technik.




*Obr. 41 Udělování oprávnění roli Technik*

# 7 NASAZENÍ ŘEŠENÍ V REÁLNÉM PROVOZU

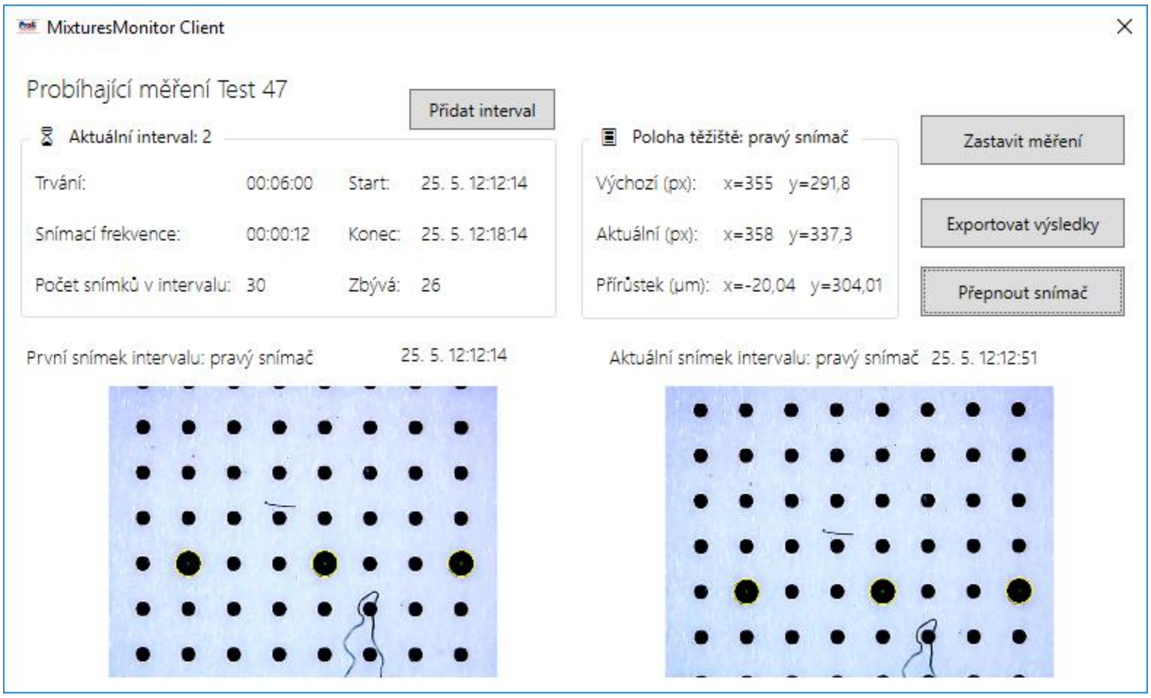
Po návrhu a realizaci obou částí řešení byly naprogramované aplikace testovány. Před nasazením desktopové aplikace na experimentální stanice zadavatele byla funkčnost ověřena provedením zkušebních měření (viz obr. 42).

## 7.1 Desktopová aplikace

Důkladné otestování stability desktopové aplikace probíhalo na počítačích poskytnutých softwarovým dodavatelem. Jednalo se o notebooky se starším i novějším operačním systémem Windows.

Při testování byly odladěny nedostatky při validaci vstupních parametrů měření. Mezi další ověřované funkčnosti patřily zejména:

- Zajištění obrazových dat ze snímačů při kalibraci a přepínání mezi rozlišeními
- Úspěšná detekce centrálních teček dilatometrických terčů
- Robustnost aplikace při akcích uživatele
- Export naměřených dat do excelových souborů s konstrukcí grafů




*Obr. 42 Desktopová aplikace při zkušebním měření*

V průběhu testování desktopové aplikace pracovníky vývojového centra zadavatele byly provedeny drobné úpravy. Testování spolehlivosti aplikace z hlediska výdrže pracovat i 4 týdny v provozu bez přerušení bylo úspěšné. Pouze u jednoho staršího notebooku došlo k pádu aplikace přibližně po dvou týdnech provozu. Z výpisu vzniklé výjimky byl zjištěno, že byl pád způsoben při komunikaci aplikace se snímačem.

Abychom mohli otestovat kompletní funkčnost aplikace nejen na počítači, kde bylo řešení vyvíjeno, bylo nutné zprovoznit serverovou část řešení na fyzickém serveru softwarového dodavatele. Tento server bude také použit při nasazení aplikace do reálného provozu vývojového centra zadavatele.

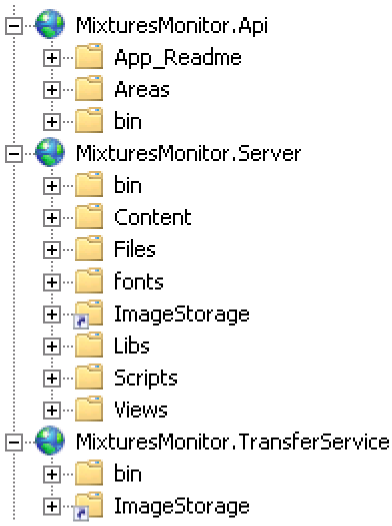
## 7.2 Serverový portál

Před publikováním webových aplikací je nutné založit jednotlivé stránky v IIS. Následně je třeba definovat procesy, pod kterými budou jednotlivé aplikace spuštěny.

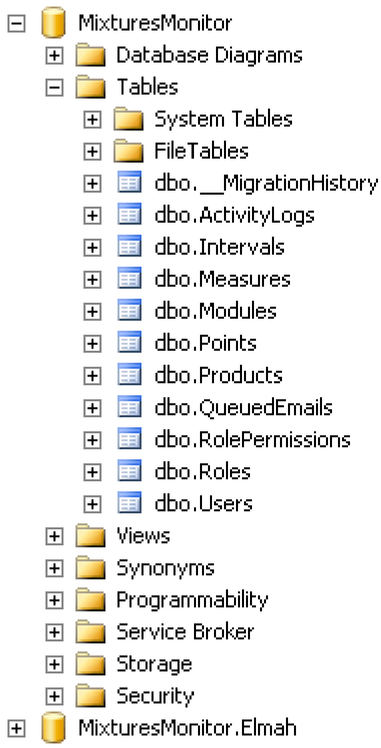
Záložka Application Pools v administraci IIS obsahuje seznam procesů, kde se vytvoří jednotlivé procesy s identitou webové služby (Network Service). S touto identitou je zajištěno, že procesy poběží i v případě, že administrátor serveru změní přístupová hesla.

Následně lze v kořenovém elementu Sites založit jednotlivé stránky (viz obr. 43.1), do kterých bude publikován obsah projektů pomocí Visual Studia.

Před publikací musí být připraveny databáze. Na serveru je k dispozici vývojové prostředí Visual Studio 2013, které je připojené na službu VSTS. Pomocí této služby byly webové projekty namapovány na disk. Následně lze přes konzoli Visual Studia vytvořit databázi MixturesMonitor. K zajištění funkce uvedené v závěru kapitoly 6.3.2 je třeba navíc vytvořit databázi MixturesMonitor.Elmah pro zachytávání případných výjimek webových aplikací. Výsledná struktura databází v prostředí Microsoft SQL Server Management Studio je zobrazena na obr. 43.2.




*Obr. 43.1 Struktura projektů založených v IIS*




*Obr. 43.2 MSSQL databáze na serveru*

Po úspěšném publikování všech webových projektů byla testována kompletní funkčnost celého řešení.

Testování komunikace desktopové aplikace a serverové části, včetně přenosu a ukládání snímků, proběhlo úspěšně.

Po provedení zkušebních měření obsluhou vývojového centra bylo rozhodnuto o nasazení desktopové aplikace do reálného provozu na všech experimentálních stanicích. V budoucnu se počítá s automatizací kalibrace snímačů, které obsluze ušetří čas při přípravě měření.

# 8 ZÁVĚR

Cílem této diplomové práce bylo navrhnout a realizovat desktopovou aplikaci určenou pro zjišťování rozměrových změn betonových směsí při tuhnutí, včetně implementace serverového portálu pro vyhodnocení získaných dat. Obě části řešení se podařilo realizovat a nasadit do reálného provozu vývojového centra zadavatele.

Úvodní část práce se věnuje analýze úlohy, která rozvádí požadavky zadavatele na vyvíjený software. V rozboru úlohy je vysvětlena podstata procesu zjišťování rozměrových změn materiálových směsí při tuhnutí. Dále následuje popis potřebného vybavení. Součástí analýzy je popis původního řešení, jehož zásadní nedostatky budou novým softwarem odstraněny. Poslední část úvodu se zabývá formou reprezentace naměřených dat.

Následuje teoretická část práce, ve které jsou stručně popsány současné softwarové technologie pro platformu Windows, které budou použity při vývoji desktopové i serverové aplikace. Přehled začíná stručnou charakteristikou vývojového prostředí Visual Studia a doplňkových služeb, které usnadňují proces tvorby aplikací. Následuje představení programovacích jazyků. V závěru jsou popsány architektury známých frameworků, knihoven a programovacích vzorů používaných při vývoji moderních aplikací.

Další kapitola práce se zabývá návrhem obou částí řešení. Úvod obsahuje vytyčení základní struktury řešení. Následuje popis návrhu desktopové aplikace, který zahrnuje obsah jednotlivých oken a soupis požadovaných funkcí s vysvětlením, pomocí kterých technologií budou realizovány. Závěr kapitoly tvoří návrh serverové části řešení se schématem popisující propojení jednotlivých částí.

V další části práce je popsána realizace desktopové aplikace v programovacím jazyce C#. S použitím architektury MVVM bylo realizováno grafické uživatelské prostředí ve formulářovém frameworku WPF. K zajištění přenositelnosti funkcí byla vytvořena sdílená knihovna obsahující datový model. Při realizaci algoritmu měření bylo použito technické vybavení zadavatele. Jednalo se o mikroskopy řady Levenhuk DTX 50, se kterými aplikace komunikuje přes rozhraní USB. Řízení mikroskopů a následné zpracování snímků bylo realizováno pomocí knihovny AForge. Naměřená data jsou z aplikace exportována v podobě excelového souboru obsahující sestrojené dilatometrické křivky pomocí knihovny EPPlus.

Další kapitola práce se věnuje implementaci serverové části řešení. Skládá se z webového rozhraní realizovaného aplikací ASP.NET Web API, které tvoří komunikační vrstvu mezi desktopovou aplikací a serverovou MSSQL databází. Webový portál určený k prezentaci výsledků měření byl realizován ASP.NET aplikací s architekturou MVC. K přístupu a práci s databází byl použit Entity framework. Přenos snímků z desktopové aplikace byl realizován přes webovou službu WCF.

V závěrečné kapitole je popsáno testování funkčnosti řešení a nasazení desktopové aplikace na stanice vývojového centra zadavatele. Kapitola obsahuje postup při nasazování serverové části na webový server poskytnutý softwarovým dodavatelem. Realizované řešení při testování splnilo požadavky zadavatele. V budoucnu se počítá s rozšířením desktopové aplikace o automatickou kalibraci snímačů.

# 9 SEZNAM POUŽITÉ LITERATURY


[1] NOVÁČEK, Jaroslav. *Ústní sdělení*. [cit. 2017-02-06].

[2] KATEDRA FYZIKY MATERIÁLŮ. *Dilatometrie*. [online]. [cit. 2017-05-08]. Dostupné z: https://material.karlov.mff.cuni.cz/cs/pristroje/dilatometrie.

[3] LEVENHUK DTX 50. *Návod k použití*. [online]. [cit. 2017-05-10]. Dostupné z: https://www.levenhuk.cz/products/materials/0/LVH-DTX-30-50-UM-ml-2017.pdf.

[4] VISUAL STUDIO IDE. *Learn to Develop with Microsoft Developer Network*. [online]. [cit. 2017-05-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/dn762121.aspx.

[5] VISUAL STUDIO IDE. *DevOps overview for Team Services and TFS*. [online]. [cit. 2017-05-12]. Dostupné z: https://www.visualstudio.com/cs-cz/docs/devops-alm-overview.

[6] MICROSOFT DOCS. *NuGet Package Manager UI*. [online]. [cit. 2017-05-12]. Dostupné z: https://docs.microsoft.com/en-us/nuget/tools/package-manager-ui.

[7] SHARP, John. *Microsoft Visual Studio C# 2010: Krok za krokem*. Brno: Computer Press, 2010. Krok za krokem (Computer Press). ISBN 978-80-251-3147-3.

[8] ITNETWORK.CZ. *1. díl – Úvod o C# a .NET frameworku*. [online]. [cit. 2017-05-12]. Dostupné z: https://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-uvod-do-jazyka-a-dot-net-framework.

[9] INTERVAL.CZ. *Kurz HTML – úvodní seznámení s HTML*. [online]. [cit. 2017-05-12]. Dostupné z: https://www.interval.cz/clanky/kurz-html-uvodni-seznameni-s-html/.

[10] ITNETWORK.CZ. *1. díl – Úvod do JavaScriptu*. [online]. [cit. 2017-05-14]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk/.

[11] ITNETWORK.CZ. *1. díl – Úvod do WPF (Windows Presentation Foundation)*. [online]. [cit. 2017-05-14]. Dostupné z: https://www.itnetwork.cz/csharp/formulare/wpf/c-sharp-tutorial-wpf-uvod-a-prvni-formularova-aplikace/.

[12] DOTNETPORTAL.CZ. *mvvm: model-view-viewmodel*. [online]. [cit. 2017-05-14]. Dostupné z: http://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel.

[13] ITNETWORK.CZ. *1. díl – Úvod do ASP.NET*. [online]. [cit. 2017-05-15]. Dostupné z: https://www.itnetwork.cz/csharp/asp-net/tutorial-uvod-do-asp-dot-net.

[14] AForge.NET Framework. *AForge.NET :: Computer Vision, Artificial Intelligence, Robotics*. [online]. [cit. 2017-05-15]. Dostupné z: http://www.aforgenet.com/framework/.

[15] CODE PROJECT. *Dependency Injection Pattern with Autofac*. [online]. [cit. 2017-05-15]. Dostupné z: https://www.codeproject.com/Articles/690391/Dependency-Injection-Pattern-with-Autofac.

[16] Entity Framework *Tutorial. What is Entity Framework*. [online]. [cit. 2017-05-15]. Dostupné z: http://www.entityframeworktutorial.net/what-is-entityframework.aspx.

[17] MICROSOFT DOCS. *Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application (9 of 10).* [online]. [cit. 2017-05-15]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application.