Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Integrace časomíry pro požární sport s webovou aplikací

Bc. Karel Wiedermann

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Karel Wiedermann

Systémové inženýrství a informatika
Informatika

Název práce

Integrace časomíry pro požární sport s webovou aplikací

Název anglicky

Linked of timer for fire sport with web applications

Cíle práce

Cílem práce je vytvořit řídící program časomíry pro požární sport. Program bude komunikovat s časomírou a webovou aplikací. Program v PC bude plně ovládat časomíru, umožňovat export výsledků do CSV a komunikovat s webovou aplikací. Ve webové aplikaci budou zobrazeny startovací listiny a dosažené výsledky.

Metodika

Prvním krokem bude oslovení SDH vlastnící časomíru a analýza požadavků. V druhém kroku bude vytvořen návrh řešení a vybraní technologií. V následujícím kroku bude implementována propojení mezi časomírou a PC. Program v PC bude plně ovládat časomíru a komunikovat s webovou aplikací.

Doporučený rozsah práce

50 – 60 stran

Klíčová slova

požární sport, C#, PHP

Doporučené zdroje informací

ALBAHARI, Joseph a Ben ALBAHARI. C# 7.0 in a nutshell. 7th edition. Sebastopol: O´Reilly, 2018. ISBN 1491987650.

BLOKDYK, Gerardus. Json Web Token. Third Edition. CreateSpace Independent Publishing Platform, 2018. ISBN 978-1985676466.

GRUNDGEIGER, Dave. Programming Visual Basic .NET. Sebastopol, CA: O'Reilly, c2002. ISBN 0596000936.

HOHPE, Gregor a Bobby WOOLF. Enterprise integration patterns: designing, building, and deploying messaging solutions. Boston: Addison-Wesley, c2004. ISBN 978-0321200686.

JACKSON, Wallace. JSON quick syntax reference. Lompoc, CA: Apress, [2016]. Expert's voice in Web development. ISBN 978-1484218624.

Masse, M.: The REST API Design Rulebook. O'Reilly Media, 2011 vydání, ISBN 978-1449310509

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 9. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 9. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 06. 04. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Integrace časomíry pro požární sport s webovou aplikací" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 6.4.2020 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Markovi Píckovi, Ph.D. za odborné vedení práce, za praktické rady a za možnost využít jeho zkušeností v problematice vývoje softwaru. Také bych rád poděkoval Bc. Janu Adamcovi za umožnění otestovat nahrání výsledků na stránky firesport.eu. Dále bych rád poděkoval členům sborů dobrovolných hasičů Kostelec nad Černými lesy, Český Brod, Sendražice a Lžovice za konzultace problematiky pořádání závodů a zapůjčení časomír.

Integrace časomíry pro požární sport s webovou aplikací

Abstrakt

Tato práce řeší vytvoření aplikace pro zjednodušení organizace závodů v požárním útoku a následný export výsledků na webové stránky firesport.eu. Zaměřil jsem se hlavně na implementaci aplikace v programovacím jazyce C#. Podařilo se mi dosáhnout 100% úspěšnosti při zkoušce testovacích scénářů popsaných v této práci. V práci jsem vytvořil aplikaci pro organizaci závodů s funkcionalitou usnadňující tvorbu startovacích listin a následnou práci s výsledky, včetně exportů do CSV a na webové stránky firesport.eu. Přínosem této práce je vytvoření aplikace k organizaci závodů v požárním útoku a konektoru pro firesport.eu. Konektor je vytvořen jako separátní knihovna, kterou je možné použít i pro jiné projekty.

Klíčová slova: požární sport, firesport.eu, TVR elektronik, C#, LINQ, Windows Forms, PHP, Postman

Integration of timers for fire sport with web application

Abstract

This work solves the creation of an application for simplifying the organization of races in a fire attack and the subsequent export of results to the website firesport.eu. I focused mainly on implementation of the application in C# programming language. I managed to achieve 100% success in the test scenarios described in this work. In this work, I created an application for organizing races with functionality facilitating the creation of start lists and subsequent work with the results, including exports to CSV and to the website firesport.eu. The contribution of this work is the creation of an application for the organization of races in fire attack and connector for firesport.eu. The connector is designed as a separate library that can be used for other projects.

Keywords: fire sport, firesport.eu, TVR electronics, C#, LINQ, Windows Forms, PHP, Postman

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

API - Application Programming Interface
CLR - Common Language Runtime
CIL - Common Intermediate Language
HZS ČR - Hasičský záchranný sbor České republiky
LINQ - Language Integrated Query
WPF - Windows Presentation Foundation

Úvod

Hasičský záchranný sbor České republiky, jednotky sboru dobrovolných hasičů, Policie České republiky a zdravotnická záchranná služba jsou základními složkami Integrovaného záchranného systému. Jejich vzájemná součinnost se odráží v přípravě na mimořádné události, při provádění záchranných a likvidačních prací. Ve své práci jsem se zaměřil na první zmiňovanou složku – HZS ČR, konkrétně na její podporu při přípravě pro požární soutěže.

Jako člen Sboru dobrovolných hasičů Sendražice se aktivně věnuji požárnímu sportu na republikové úrovni. Vybral jsem si téma, kterým chci zjednodušit pořádání závodů v požárním útoku nejen mého mateřské sboru, ale velkého spektra ligy a sborů v České republice.

Tato práce se zabývá vytvořením aplikace pro zjednodušení a zautomatizování organizace závodů a následné nahrání výsledků na webové stránky firesport.eu. Aplikace umožnuje vytvoření prezenčních listin a následné generování startovních listin a automatické přiřazování časů čtených z časomír společnosti TVR elektronik.

Organizace závodů v požárním útoku je velmi specifická dle pravidel ligy, do které jsou dané závody zařazeny. V České republice a Slovenské republice se nachází více jak 50 lig. Ligy mají upravená pravidla oproti pravidlům postupových soutěží, které jsou vydány generálním ředitelem HZS ČR. V ligách dochází k značným rozdílům. Organizace závodů v požárním útoku je velmi různorodá. Dle typu ligy se závodů účastní týmy různých kategorií, na různě dlouhých tratích, na různém počtu základen a i v různém počtu pokusů na závodech.

Tato práce vytváří obecnou aplikaci, kterou je možné použít k organizaci všech známých alternativ závodů v požárním útoku.

1 Cíl práce a metodika

1.1 Cíl práce

Cílem diplomové práce je vytvoření aplikace k zjednodušení organizace závodů. V aplikaci bude možné vytvořit prezenční listinu, ze které bude následně generována startovní listina dle pravidel získaných od sboru dobrovolných hasičů, kteří pořádají závody v požárním útoku na okrese Kolín. Aplikace bude komunikovat s časomírami TVR elektronik, ze kterých budou čteny dosažené časy. Časy se budou automaticky přiřazovat k týmům dle pořadí ve startovní listině. Konečné výsledky bude možné exportovat do CSV nebo nahrát na webové stránky firesport.eu, které sdružují 54 lig z České republiky a Slovenské republiky. Aplikace bude umožňovat variabilní nastavení počtu kol a tratí na pořádaných závodech.

1.2 Metodika

Teoretická část práce bude obsahovat rozbor témat týkajících se požárního útoku a použitých technologií. Pro práci bude použita technologie .NET Framework a jeho programovací jazyk C#, jako grafická nadstavba bude použit Windows Forms. Ve zdrojovém kódu aplikace se velmi často bude používat jazyk LINQ pro iteraci nad kolekcemi a selekcí instancí.

Pro vytvoření požadavků na aplikaci bude proveden strukturovaným rozhovor, kterým budou shromážděny požadavky od sborů pořádajících závody v požárním útoku na okrese Kolín. Ze sesbíraných požadavků od budoucích uživatelů aplikace bude vytvořeno zadání. Pro návrh aplikace bude použito jazyka UML. Vytvořením use case diagramu budou zobrazeny obecné funkcionality aplikace. V práci budou vytvořeny aktivity diagramy popisující chování aplikace. Dále bude vytvořen Class diagram znázorňující třídy a vztahy mezi nimi.

Před zahájením programování aplikace bude pomocí aplikace Serial Port Monitor zjištěno, jak časomíry komunikují s počítačem přes sériový port. Dalším nutným krokem bude zjištění jak nahrát data na webové stránky firesport.eu. K zjištění přenosu dat bude použita Google konsole a následně aplikace Postman k vytvoření vzorových http požadavků.

Posledním krokem práce bude vyzkoušení testovacích scénářů pro ověření správné funkčnosti aplikace.

2 Teoretická východiska

2.1 Požární sport

Požární sport spojuje atletiku s určitými prvky a úkony z práce hasičů. Pro jeho zvládnutí je nutné spojit rychlost a obratnost, nebát se překážek, ani ohně, ani výšek. Požární sport se skládá ze čtyř základních disciplín, běh na sto metrů překážek, výstup do čtvrtého podlaží cvičné věže pomocí hákového žebříku, štafety a požárního útoku. Praktická část práce je zaměřena na vytvoření aplikace na organizaci závodů v požárním útoku. [1]

Běh na 100 metrů s překážkami je individuální disciplína, ve které závodník překonává 2 m vysokou bariéru, pak sbírá dvě hadice, se kterými běží přes kladinu (dlouhá 8 m, široká 18 cm a vysoká 1,2 m). Dalším úkolem je spojení hadic, napojení jedné z nich na rozdělovač a nakonec po napojení proudnice proběhnutí cílem. [1], [2]

Druhou individuální disciplínou požárního sportu je výstup do 4. podlaží cvičné věže pomocí hákového žebříku, vážícího 8,5 kg. [1], [2]

Členové štafety 4x100 metrů s překážkami musí překonávat na trati různé překážky a uhasit skutečný oheň. Úkolem je v co nejkratším čase donést do cíle hasičskou proudnici, která slouží jako štafetový kolík. [1]

Požární útok je mezi hasiči označován jako „královská disciplína“. Je to z toho důvodu, že se musí skloubit mnoho faktorů, vnitřních i vnějších, kterých je zapotřebí, aby se útok vydařil. Na soutěžích v požárním útoku se snaží sedmičlenná družstva co nejrychleji sepnou spínač na terčích. Voda nabraná z kádě putuje savicemi přes přenosnou hasičskou stříkačku do hadic přes rozdělovač až do proudnic. Při požárním sportu se používají dva typy terčů. Oba terče mají ve výšce 160 cm nad zemí umístěný otvor o průměru 5 cm. Prvním typem terčů jsou nástřikové, které jsou používány v postupových soutěžích. Proudaři při tomto typu terčů mají za úkol nastříkat 10 litrů vody do nástřikové nádrže a tím sepnout plovací spínač. Nejlepší proudaři stříkají poslepu a jen poslouchají zvuk terče, jelikož při správném úhlu vody k terči, kde voda přímo proudí do nástřikové nádoby, terč vydává dunivý zvuk. Druhý typ terčů používaný při hasičský soutěží je terč sklopný, kde má proudař své snažení velmi zjednodušené. Stačí jen proudem vody

převrátit závažný válec sklopného terče a tím sepnout spínač. Voda ze stříkačky putuje hadicemi přes rozdělovač až k proudnicím. [1], [2]

2.1.1 Složení družstva

- Košař
- Spojař (savičář)
- Strojník
- Béčkař
- Rozdělovač
- Levý proudař
- Pravý proudař

2.2 Časomíry TVR elektronik

Firma TVR elektronik působí na domácím i zahraničním trhu již řadu let. Byla založena v roce 1997 se specializací na vývoj i výrobu kontrolních a testovacích přístrojů pro rakouskou firmu, která se zabývala celosvětovou výrobou relé. S touto firmou spolupracuje dodnes. [3]

Během dalších let se působení firmy rozrostlo i do jiných odvětví. V sortimentu firmy se nacházejí časomíry pro různá odvětví různých sportů, časomíry s odečtem času, pomůcky pro dyslektické děti, zemědělská elektronika, elektronika pro modeláře a ukazatele skóre pro fotbal, florbal, basketbal, házenou, ragby. [3]

V současné době firma vyrábí několik typů digitálních časomír. Dále je schopna vyrobit časomíry dle konkrétních požadavků zákazníka v krátkých dodacích lhůtách a bez zvýšených nákladů na vývoj. Nabízí také různé LED displeje, časoměrná zařízení, nápisy, hodiny, ukazatele skóre, teplot. Společnost TVR elektronik má pro ovládání časomír vytvořenou jednoduchou aplikaci pro sběr dat. Tato aplikace je vytvořena velmi obecně tak, aby splňovala základní náležitosti pro všechny dodávané časomíry společnosti pro různé druhy sportů. [3]

2.2.1 Model S-150

Časomíra je základní verzí časomír, které jsou používány na závodech v hasičském sportu. Obsahuje 5 číslic složených z vysoce svítivých diod. [3]

Obrázek 1 - Časomíra model S-150 [3]

2.2.2 Model DUAL-150Z

Časomíra Dual je typickým příkladem, který se vyskytuje na většině požárních soutěžích v České republice a na Slovensku. Časomíra se skládá ze dvou displejů o velikosti číslic 150 mm. Je tedy možné zobrazovat časy obou terčů na ráz. K časomíře je možné připojit až čtyři terče a tím zřídit dvě trati pro požární útok. Tyto trati ale není možné startovat paralelně, ale vždy pouze jednu trať. Časomíra obsahuje pouze jedny stopky. [3]

2.2.3 Model DUAL-150Z - Triple

Jedná se o zatím prvním typ časomíry v České republice. Byla speciálně vyrobena pro potřeby Podlipanské ligy. Podlipanská liga jako jediná v České republice startuje na svých závodech ze tří základen a běhy měří jednou časomírou.

Časomíra designově vychází z typu časomíry Dual-150Z s tím rozdílem, že je možné k časomíře připojit až 6 terčů a tím vytvořit tři startovní tratě. Tratě také nejde startovat paralelně jako u předchozího typu, ale vždy jenom jednu po druhé. [3]

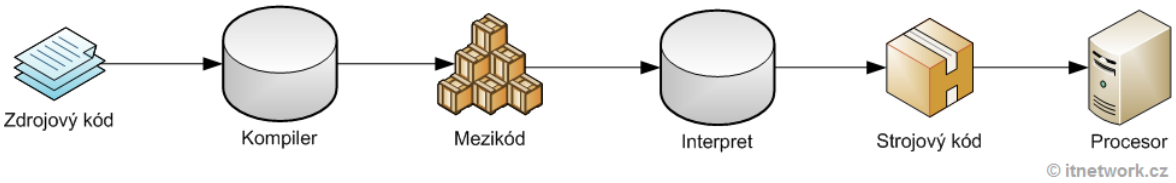
Obrázek 2 - Časomíra model DUAL-150 [3]

2.3 Jazyk C#

Jazyk C# je řazen do jazyků s virtuálním strojem podobně jako Java. Zdrojový kód je nejprve přeložen do takzvaného mezikódu, který je nazýván společností Microsoft Common Intermediate Language neboli CIL. Jedná se o binární kód, který má o poznání jednodušší instrukční sadu a přímo podporuje objektové programování. Tento mezikód je potom díky jednoduchosti relativně rychle interpretovatelný takzvaným virtuálním strojem. Výsledkem je strojový kód pro procesor. [19], [22]

Virtuální stroj v .NET je Common Language Runtime neboli CLR implementuje systém Virtual Execution System, který je definován standardem Common Language Infrastructure, původně vyvinutým samotnou společností Microsoft.

Všechny programy napsané pro framework .NET, bez ohledu na programovací jazyk, jsou prováděny v CLR. Všechny verze .NET framework zahrnují CLR. CLR poskytuje služby správy paměti, zpracování výjimek , bezpečnosti a správy vláken. [4], [19], [23]



Obrázek 3 - Kompilace zdrojového kódu [19]

2.3.1 .NET Framework

.NET Framework se skládá ze čtyř základních částí, které jsou dodávány společně. Jedná se o technologii, kde si vývojář může vybrat z řady jazyků. Vývojář má na výběr z objektových jazyků jako je například C# nebo funkcionálních jako je F#. Dalším částí je vývojové prostředí Visual Studio, které poskytuje řadu užitečných funkcionalit pro vývoj aplikacích na platformě .NET. Důležitým pilířem je Virtuální stroj CLR popisovaný v předchozí kapitole. Nedílnou součástí je sada standardizovaných knihoven dodaných společností Microsoft. [4], [19]

2.3.2 Knihovny

Knihovny jsou největší výhodou .NETu. Microsoft v podstatě dodává kompletní sadu knihoven, ve které je předdefinovaná řada struktur a komponent pro práci s konzolí, databázemi, formulářovými prvky a podobně. Řešení jsou kvalitní, aktuální a jsou sdílené mezi jednotlivými jazyky. Jednou z předností komponent je, že jsou odladěny pro vývoj pro operační systémy Windows, tento fakt je zapříčiněn tím, že Microsoft je i autorem operačních Windows. Pro běh aplikací je potom nutné, aby na koncové stanici byla ta samá verze .NET Frameworku, ve které byla aplikace vyvinuta. Windows v základu obsahují .NET Framework, nebo je možné vytvořením instalačního balíčku aplikace .NET Framework zabalit k aplikaci. [4], [19], [23]

2.3.3 Language Integrated Query

Language Integrated Query neboli LINQ je integrovaný jazyk .NET Frameworku pro dotazování. Přínosem LINQ je představení jednotné syntaxe pro přístup k datům bez ohledu na jejich zdroj, kterým může být databázové rozhraní, XML soubor nebo

objekt v paměti. LINQ usnadňuje transformaci, třídění a propojování dat a vyhledávání v nich. [26]

LINQ je navržen jako poměrně obecný nástroj. Je v něm možné manipulovat s různými daty. LINQ to Objects umožňuje dotazování nad normálními objekty, respektive jejich kolekcemi, LINQ to SQL přináší nový způsob pro práci s databázemi a LINQ to XML umožňuje pracovat s XML soubory. Na internetu jsou dostupné i další implementace LINQu, jako je například LINQ to Amazon, který slouží pro vyhledávání knih v tomto internetovém obchodě. [24], [26]

Výše jmenované jazyky byly rozšířeny o nová klíčová slova a další podpůrné jazykové konstrukce. Klíčové pro pochopení jazyka LINQ je zejména znalost těchto konstrukcí, především Lambda výrazů, které jsou nejjednodušší metodou zápisu anonymních metod. Důležitým pilířem jsou inicializátory objektů a kolekcí. Anonymní třídy umožňují rychlé vytvoření objektů přenášejících informace vyžádané z databáze přes LINQ. [24], [26]

Pro vývojáře je vhodná znalost rozšiřující metody jazyka LINQ. Metody by bylo možné rozdělit do šesti základních skupin:

1. Metody pro selekci jako je Select nebo Cast.
2. Metody pro omezení výběru jako je Where, Skip.
3. Metody pro agregaci jako jsou Sum, Count a tak dále.
4. Metody pro vytváření kolekce Empty, Repeat, Range.
5. Metody převádějící kolekce - do těchto metod by byly řazeny ToArray, ToList, ToHashSet a další.
6. Posledním skupinou jsou ostatní metody, které mění vstup sekvencí na výstup sekvencí. Řadila by se tam metoda pro spojení kolekcí Join nebo OrderBy pro specifické třídění a řada dalších.

Metody Where a OrderBy, jež v kontextu LINQ lze s úspěchem považovat za klíčové, je na poli čísel možné volat, protože jsou definovány pro všechny objekty implementující rozhraní IEnumerable, mezi něž pole čísel patří. Použité lambda výrazy,

jejichž poznávacím znamením je operátor =>, jsou znázorněny v metodách Where a OrderBy v příkladu na obrázku 4. [24], [26]

Pro názornou ukázku práce s LINQ je vytvořen příklad na Obrázku 4. Jsou zde znázorněny metody Where a OrderBy. Jejich mechanismus funguje tak, že za proměnou n se postupně dosadí každé číslo z pole numbers. Jak je patrné, díky této technolgii je psaný kód kratší a při rozsáhlých iteracích přehlednější. [24], [26]

```
static void SampleLambdaNumbers()
{
    int[] numbers = { 6, 3, 7, 9, 5 };

    var result = numbers.Where(n => n > 5).OrderBy(n => n);

    Console.WriteLine("Where and Order list of numbers:");
    foreach (int number in result)
        Console.WriteLine(number);
}
}
```

```
file:///C:/Users/KarelWiedermann/Google D
Where and Order list of numbers:
6
7
9
```

Obrázek 4 - LINQ příklad [25]

2.4 Vlákna

S rozhraním .NET je možné psát aplikace, které provádějí více operací současně. Operace s potenciálem zadržení dalších operací mohou být prováděny na samostatných vláknech, což je proces známý jako multithreading (více vláknový nebo volné vlákno).

Aplikace, které používají více vláknové zpracování, lépe reagují na vstup uživatele, protože uživatelské rozhraní zůstává aktivní, jelikož nároční úlohy na procesor se provádějí na samostatných vláknech. Více vláknové zpracování je užitečné také při vytváření škálovatelných aplikací, protože můžete přidávat vlákna při zvyšování pracovního vytížení. [12]

Více vláknové aplikace mohou zlepšit výkon aplikací pro Windows Forms, ale přístup k ovládacím prvkům Windows Forms není ze své podstaty bezpečný pro přístup z více vláken. Více vláken může vystavit v kódu velmi závažné a složité chyby. Dva nebo více podprocesů manipulující s ovládacím prvkem mohou vynutit ovládací prvek do nekonzistentního stavu a vést ke sporným stavům, zablokování a zamrznutí. Pokud je implementováno více vláken v aplikaci, je nutné volat ovládací prvky mezi vlákny bezpečným způsobem.

Existují dva způsoby jak bezpečně volat ovládací prvky Windows Forms z vlákna, které daný ovládací prvek nevytvořilo. Metodu System.Windows.Forms.Control.Invoke je možné použít k volání delegáta vytvořeného v hlavním vlákně, který volá ovládací prvek. Druhá možnost je implementace System.ComponentModel.BackgroundWorker, který používá model řízený událostmi k oddělení práce provedené ve vlákně na pozadí. [12],[13]

2.4.1 Nebezpečná volání mezi vlákny

Je nebezpečné volat ovládací prvek přímo z vlákna, které jej nevytvořilo. Následující fragment kódu ilustruje nebezpečné volání ovládacího System.Windows.Forms.TextBox prvku. Obslužná metoda Button1_Click WriteTextUnsafe události vytvoří nové vlákno, které přímo nastaví vlastnost text na komponentě textBox1 z vlákna, které tuto komponentu nevytvořilo. [13]

C# Kopírovat

```
private void Button1_Click(object sender, EventArgs e)
{
    thread2 = new Thread(new ThreadStart(WriteTextUnsafe));
    thread2.Start();
}
private void WriteTextUnsafe()
{
    textBox1.Text = "This text was set unsafely.";
}
```

Obrázek 5 - Nebezpečné volání mezi vlákny [13]

Ladící program sady Visual Studio zjistí InvalidOperationException neboli nebezpečné volání vlákna, vyvoláním funkce s neplatnou operací mezi vlákny.

2.4.2 Bezpečná volání mezi vlákny

Následující příklady kódu ukazují dva způsoby, jak bezpečně volat ovládací prvek Windows Forms z vlákna, které jej nevytvořilo:

1. Metoda System.Windows.Forms.Control.Invoke, která volá delegáta z hlavního vlákna pro volání ovládacího prvku.
2. Součást System.ComponentModel.BackgroundWorker, která nabízí model řízený událostmi.

2.4.2.1 Použití metody Invoke s delegátem

Následující příklad ukazuje vzor pro zajištění volání bezpečné pro přístup z více vláken ovládacího prvku Windows Forms. Dotazuje System.Windows.Forms.Control.InvokeRequired vlastnost, která porovnává ID vytvoření vlákna a ovládacího prvku s ID volajícího vlákna. Pokud ID podprocesu je stejné, volá se ovládací prvek přímo. Pokud se ID podprocesu liší, volá se metoda Control.Invoke s delegátem z hlavního vlákna, který provede skutečné volání ovládacího prvku.

Na obrázku 6 je znázorněn příklad. Delegát SafeCallDelegate nastaví vlastnost Text komponenty TextBox. Pokud InvokeRequired v metodě WriteTextSafe vrátí true, tak je metodě SafeCallDelegate předáno volání ovládacího prvku. Pokud InvokeRequired vrátí false, tak metoda WriteTextSafe nastaví přímo TextBox.Text. [13]

```
using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;

namespace InvokeThreadSafeForm
{
    public class InvokeThreadSafeForm : Form
    {
        private delegate void SafeCallDelegate(string text);
        private Button button1;
        private TextBox textBox1;
        private Thread thread2 = null;

        [STAThread]
        static void Main()
        {
            Application.SetCompatibleTextRenderingDefault(false);
            Application.EnableVisualStyles();
            Application.Run(new InvokeThreadSafeForm());
        }
        public InvokeThreadSafeForm()
        {
            button1 = new Button { Location = new Point(15, 55), Size = new Size(240, 20), Text = "Set text safely" };
            button1.Click += new EventHandler(Button1_Click);
            textBox1 = new TextBox { Location = new Point(15, 15), Size = new Size(240, 20) };
            Controls.Add(button1);
            Controls.Add(textBox1);
        }
        private void Button1_Click(object sender, EventArgs e)
        {
            thread2 = new Thread(new ThreadStart(SetText));
            thread2.Start();
            Thread.Sleep(1000);
        }
        private void WriteTextSafe(string text)
        {
            if (textBox1.InvokeRequired)
            {
                var d = new SafeCallDelegate(WriteTextSafe);
                textBox1.Invoke(d, new object[] { text });
            }
            else
                textBox1.Text = text;
        }
        private void SetText()
        {
            WriteTextSafe("This text was set safely.");
        }
        private void InitializeComponent()
        {
            this.SuspendLayout();
            this.ClientSize = new System.Drawing.Size(274, 229);
            this.Name = "InvokeThreadSafeForm";
            this.ResumeLayout(false);
        }
    }
}
```

Obrázek 6 - Použití metody Invoke s delegátem [13]

2.4.2.2 Použití obslužné rutiny události BackgroundWorker

Snadný způsob jak implementovat multithreading je s System.ComponentModel.BackgroundWorker, která používá model řízený událostmi. Podproces na BackgroundWorker.DoWork na pozadí spustí událost, která nepracuje s hlavním vláknem. Hlavní vlákno spustí BackgroundWorker.ProgressChanged BackgroundWorker.RunWorkerCompleted obslužné rutiny a události, které mohou volat ovládací prvky hlavního vlákna. [13]

Provede se bezpečné volání BackgroundWorker pro přístupu z více vláken pomocí vytvoření DoWork metody ve vlákně na pozadí. Je nutné ji spojit s událostí. V případě, že je požadováno zpuštění vlákna na pozadí, je nutné zavolat BackgroundWorker.RunWorkerAsyncna. [13]

Příklad na obrázku 7 používá RunWorkerCompleted obslužnou rutinu pro TextBox. Událost nastaví vlastnost text pro komponentu textBox1. [13]

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;

namespace BackgroundWorkerForm
{
    public class BackgroundWorkerForm : Form
    {
        private BackgroundWorker backgroundWorker1;
        private Button button1;
        private TextBox textBox1;

        [STAThread]
        static void Main()
        {
            Application.SetCompatibleTextRenderingDefault(false);
            Application.EnableVisualStyles();
            Application.Run(new BackgroundWorkerForm());
        }
        public BackgroundWorkerForm()
        {
            backgroundWorker1 = new BackgroundWorker();
            backgroundWorker1.DoWork += new DoWorkEventHandler(BackgroundWorker1_DoWork);
            backgroundWorker1.RunWorkerCompleted += new RunWorkerCompletedEventHandler(BackgroundWorker1_RunWorkerCompleted);
            button1 = new Button { Location = new Point(15, 55), Size = new Size(240, 20), Text = "Set text safely with BackgroundWorker" };
            button1.Click += new EventHandler(Button1_Click);
            textBox1 = new TextBox
            {
                Location = new Point(15, 15),
                Size = new Size(240, 20)
            };
            Controls.Add(button1);
            Controls.Add(textBox1);
        }
        private void Button1_Click(object sender, EventArgs e)
        {
            backgroundWorker1.RunWorkerAsync();
        }

        private void BackgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            // Sleep 2 seconds to emulate getting data.
            Thread.Sleep(2000);
            e.Result = "This text was set safely by BackgroundWorker.";
        }

        private void BackgroundWorker1_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
        {
            textBox1.Text = e.Result.ToString();
        }
    }
}

```

Obrázek 7 - Použití metody BackgroundWorker [13]

2.5 Grafické možnosti

Tvorba grafického rozhraní v desktopových aplikacích na platformě Microsoft .NET v jazyku C# je možná ve více technologiích. Historicky je možné používat Windouws API nebo jeho zapouzdření pomocí knihovny Microsoft Foundation Class Library. Další z možností je použití Windows Forms nebo nejnovější technologie Windows Presentation Foundation. [4], [18], [19]

2.5.1.1 Windows Forms

Windows Forms je grafická knihovna zahrnutá jako součást Microsoft .NET Framework a Mono .NET Frameworku. Mono je softwarová platforma, která vývojářům umožňuje snadno vytvářet aplikace napříč platformami.

Windows Forms poskytuje platformu pro psaní rozsáhlých klientských aplikací pro počítače a tablety. Považuje se za náhradu dřívější a složitější knihovny Microsoft Foundation Class Library založenou na C++, nenabízí srovnatelné paradigma a ve víceúrovňovém řešení funguje pouze jako platforma pro úroveň uživatelského rozhraní. [15], [17], [18]

2.5.1.1.1 Architektura

Aplikace Windows Forms je řízená událostmi podporovanými Microsoft .NET Frameworkem. Aplikace využívající Windows Forms na rozdíl od dávkového programu tráví většinu času jednoduše čekáním, až uživatel něco udělá, například vyplní textové pole nebo klepne na tlačítko.

Windows Forms poskytuje přístup k nativním ovládacím prvkům uživatelského rozhraní Windows zabalením existujícího rozhraní Windows API do spravovaného kódu. Každý ovládací prvek v aplikaci Windows Forms je konkrétní instancí třídy. [16], [17], [18]

2.5.1.1.2 Funkce

Všechny vizuální prvky v knihovně Windows Forms pocházejí z třídy Control. Knihovna poskytuje minimální funkčnost prvků uživatelského rozhraní, jako je umístění, velikost, barva, písmo, text a také běžné události, jako je kliknutí a přetažení. Třída Control

umožňuje, aby ovládací prvky bylo možné za běhu aplikace přeuspořádat. Dále nabízí podporu Microsoft Active Accessibility, které pomáhá uživatelům se sníženou schopností používat Windows Forms lépe. [15], [16], [17]

Kromě poskytování přístupu k nativním ovládacím prvkům Windows, jako je tlačítko, textové pole, zaškrtávací políčko, seznam, Windows Forms přidal své vlastní ovládací prvky a komponenty na zobrazení dat jako je například DataGrid, TreeView a další. Tyto ovládací prvky jsou vykreslovány pomocí GDI+. [15], [16], [17]

2.5.1.2 Windows Presentation Foundation

Windows Presentation Foundation neboli WPF je knihovna tříd pro tvorbu grafického rozhraní, která je součástí .NET frameworku od verze 3.0. Windows WPF je brán jako nástupce Windows Forms. [20], [21]

WPF je součástí Windows Vista, Windows 7 a Windows Server 2008 a je možné ji doinstalovat do Windows XP SP2/SP3 a Windows Server 2003.

Pro vytvoření uživatelsky bohatého rozhraní využívá WPF značkovací jazyk XAML, který umožňuje oddělit funkčnost a vzhled aplikace. Cílem WPF je sjednotit uživatelské rozhraní pro 2D a 3D grafiku, vektorovou a rastrovou grafiku, animace, audio a video. [20], [21]

2.5.1.2.1 Grafika

Všechna grafika včetně samotných WPF oken funguje pomocí Direct3D knihoven. To umožňuje hardwarovou akceleraci pomocí grafické karty a pokročilejší grafické schopnosti. Využitím vektorové grafiky jsou objekty matematicky popsány a díky tomu při plynulém přibližování nedochází k rozmazání. [20], [21]

2.6 Návrhové vzory

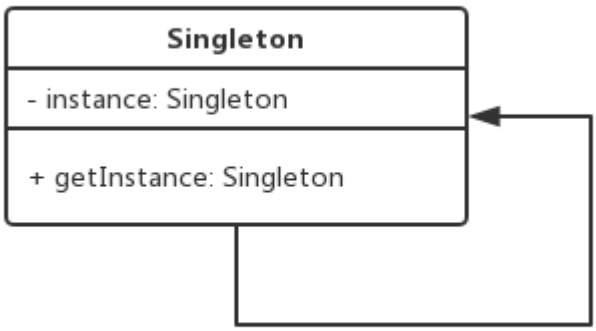
Návrhové vzory jsou nedílnou součástí objektově orientovaného programování. Návrhový vzor (anglicky design pattern) v softwarovém inženýrství představuje obecné řešení problému, které se využívá při návrhu počítačových programů. Návrhový vzor není knihovnou nebo částí zdrojového kódu, která by se dala přímo vložit do programu. Jedná se o popis řešení problému nebo šablonu, která může být použita v různých situacích. Objektově orientované návrhové vzory typicky ukazují vztahy a interakce

mezi třídami a objekty, aniž by určovaly implementaci konkrétní třídy. Algoritmy nejsou považovány za návrhové vzory, protože řeší konkrétní problémy, a nikoliv problémy návrhu. [5], [6]

Návrhové vzory nepocházejí ze softwarového inženýrství. Jsou zcela běžné v každodenním životě. K asi nejznámějším a nejstarším příkladům patří architektura. [5], [6]

2.6.1 Jedináček

Návrhový vzor jedináček (anglicky singleton) je využíván, když je v aplikaci potřeba sdílet jednu instanci mezi několika bloky, objekty, aniž by bylo nutné stále předávat v konstruktoru. Ukázkový příklad je databázové připojení. Celý program pracuje s jedním připojením a bylo by nepraktické ho stále předávat. Nabízí se řešení udělat třídu poskytující databázové API jako statickou. Může však nastat případ, kdy se hodí ji mít instanciovatelnou. Například někdy aplikace může pracovat s více připojeními nebo aplikace používá třídu, která není statická. Ve zmíněných případech je vhodné použít návrhový vzor jedináček. [5],[6],[7]



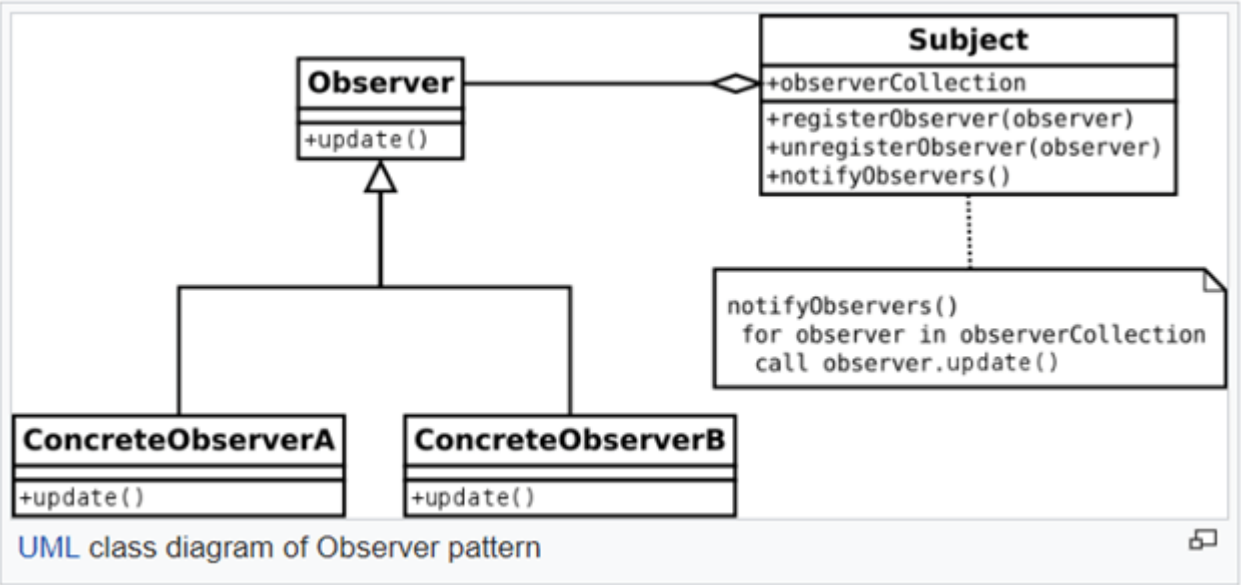
Obrázek 8 – Diagram jedináček [9]

2.6.2 Pozorovatel

Pozorovatel známý pod anglickým názvem observer je návrhový vzor pozorovatele, který umožňuje předplatiteli zaregistrovat se a přijímat oznámení od poskytovatele. Je vhodný pro jakýkoli scénář, který vyžaduje oznámení na základě nabízených oznámení.

Vzor definuje poskytovatele a jednoho nebo více pozorovatelů. Pozorovatelé se registrují u poskytovatele a pokaždé, když dojde ke změně předdefinované podmínky, události nebo stavu, poskytovatel automaticky upozorní všechny pozorovatele tím, že zavolá jednu z jejich metod. V rámci tohoto volání metody může poskytovatel také poskytnout informace o aktuálním stavu pozorovatelům. V .NET Framework se vzor návrhu pozorovatele používá implementací obecných rozhraní System.IObservable<T> a System.IObserver<T>. Parametr obecného typu představuje typ, který poskytuje informace o oznámení. [4], [5], [8]

Událost je zpráva odeslaná objektem k signalizaci výskytu akce. Tato akce může být způsobena zásahem uživatele, například kliknutím na tlačítko, nebo může být výsledkem některé jiné logiky programu, jako je například změna hodnoty vlastnosti. Objekt, který vyvolává událost, se nazývá odesílatel události. Odesílatel události neví, který objekt nebo metoda obdrží (zpracuje) události, které vyvolá. [4], [5], [6], [8]



Obrázek 9 - Diagram pozorovatel [10]

2.7 Číselníky

Číselník popisuje obsah, strukturu a rozvržení kolekce dat. Dobře zdokumentovaný číselník obsahuje informace, které mají být úplné a samy vysvětlující každou proměnnou v datovém souboru. [11]

Číselníky začínají základními úvodními materiály, včetně názvu studie, jména hlavního řešitele, obsahu a úvodem popisujícím účel a formát číselníku. Některé číselníky také obsahují metodologické podrobnosti, jako je způsob výpočtu hmotností a nástroje pro sběr dat, zatímco jiné, zejména s většími nebo složitějšími datovými sbírkami, ponechávají tyto podrobnosti pro samostatnou uživatelskou příručku a nástroj pro sběr dat. [11]

3 Vlastní práce

3.1 Shromáždění požadavků na ovládací aplikaci časomír

Požadavky na vytvoření ovládací aplikace byly získány pomocí strukturovaného rozhovoru. Strukturovaný rozhovor byl veden se členy sborů starající se o dvě různé hasičské ligy v požárním útoku na okrese Kolín. Všem byly položeny stejné otázky.

Prvními tázanými byli členové sboru dobrovolných hasičů Sendražice a sboru dobrovolných hasičů Lžovice. Zmíněné sbory zajišťují materiální a organizační zajištění Polabské hasičské Ligy. Závodů Polabské hasičské ligy se účastní přibližně 25 týmů v kategoriích mužů, žen, starších žáků a mladších žáků.

Druhými tázaným byli členové sboru dobrovolných hasičů Kostelec nad Černými lesy a sboru dobrovolných hasičů Český Brod. Zmíněné sbory zaštiťující organizaci Podlipanské ligy, z. s. Závody podlipanské ligy jsou na počet závodníků vice jak dvojnásobné než závody polabské hasičské ligy, závodů se účastní přibližně 60 týmů v kategoriích mužů, žen, smíšeného dorostu, starších žáků, mladších žáků a přípravky.

3.1.1 Řízený rozhovor se sbory zajišťující polabskou hasičskou ligu

Na kolik základen (tratí) se běhají závody polabské hasičské ligy?

Liga je běhána převážně na dvě základny. Některá kola ligy jsou běhána pouze na jednu základnu z důvodu velikosti hasičského areálu sboru, u kterého je soutěž pořádána a kde není prostor k postavení dvou základen.

Na jakých tratích běhají vaše kategorie?

Oproti pravidlům postupových soutěží muži běhají pouze na 2 B hadice a 4 C hadice, tedy na stejné tratě jako ženy. Tato úprava je z toho důvodu, aby nemusela být přestavována trať pro dospělou kategorii. Děti běhají dle pravidel hry Plamen, tedy na dvě 2 B hadice a 4 C hadice poloviční délky než dospělá kategorie.

Jak na závodech děláte startovní listiny?

Sbor dobrovolných hasičů Lžovice daroval lize školní tabuli, kde je ručně vytvořeno startovní pořadí. Vytváření startovního pořadí občas není úplně jednoduché, když na soutěž přijede sbor, který přihlašuje více týmů do soutěže. Když tato situace nastane je nutné, aby týmy startovaly ze stejné základny. Týmy nemohou startovat přímo za sebou,

jelikož většina sborů má pro kategorii mužů i žen nebo pro „tým A¨ a „tým B“ pouze jednu sadu hadic, kterou musejí pro další pokus znovu smotat a připravit. Tato činnost trvá v řádech minut.

Jak vyhodnocujete časy zobrazené na časomíře?

Časy jsou opisovány obsluhou z časomíry nebo hlavním rozhodčím na papír a tabuli se startovací listinou. Po dokončení všech běhů v kategorii je počítáno pořadí družstev.

Jak nahráváte dosažené výsledky na firesport?

Dosažené výsledky na závodech jsou opsány z vyplněného papíru od hlavního rozhodčího do formuláře pro nahrání na firesport.eu. Jelikož firesport.eu má specifický formát pro zadání výsledků občas dochází k chybám při zadání výsledků.

3.1.2 Řízený rozhovor se sbory zajištující podlipanskou ligu, z.s

Na kolik základen (tratí) se běhají závody podlipanské ligy?

Závody ligy jsou od roku 2019 převážně běhány na tři základny, které je možné rozestavit na fotbalových hřištích, kde se většina kol podlipanské ligy pořádá. Některá kola ligy jsou běhána v hasičský areálech, kde se vejdou pouze dvě základny.

Na jakých tratích běhají vaše kategorie?

Organizačně je velmi složité pořádání kola podlipanské ligy. Tratě na závodech jsou třikrát přestavovány. Většina závodů začíná ráno dětskou kategorií, která je běhána dle pravidel hry Plamen. Po nich následuje kategorie žen s kategorií smíšeného dorostu. Obě kategorie běhají na 2 B hadice a 4 C hadice. Závody uzavírá mužská kategorie na 3B hadice a 4 C hadice.

Jak na závodech děláte startovní listiny?

Startovní listiny jsou dělány po prezenci sborů jednotlivých kategorii. Vytvořit startovní listinu pro dětské kategorie není jednoduchá záležitost. Některé sbory jezdí na závody i s pěti týmy v dětské kategorii. Tyto týmy je nutné rozlosovat vždy na stejnou trať, aby nebylo nutné stěhovat požární stříkačku daleko. Dále je nutné, aby byly vytvořeny

rozestupy mezi startovními týmy. Rozestupy mezi týmy se snažíme dělat alespoň přes dva týmy, ale občas to není možné.

Jak vyhodnocujete časy zobrazené na časomíře?

Časy jsou opisovány z časomíry obsluhou časomíry na papír. Dle zapsaných časů je vytvořena výsledkový listina.

Jak nahráváte dosažené výsledky na firesport.eu?

Výsledky jsou opisovány z výsledkových listin do excelového souboru, ze kterého jsou pak importovány na firesport.eu. Excelový soubor je dále uložen ve formátu pdf. Tento soubor je pak umístěn na webové stránky ligy. Jelikož dochází k ručnímu opisování výsledků občas dochází k zanesení chyby do výsledků.

3.1.3 Vyhodnocení rozhovorů

Organizátoři lig se shodli v tom, že vytváření startovních listin je velmi pracné, protože jsou na obou ligách vytvářeny ručně. Při vytváření startovních listin se organizátoři závodů snaží dbát na to, aby týmy z jednoho sboru, které soutěží na stejné trati, startovaly vždy ze stejné základny a neměly startovní čísla jdoucí po sobě. Další shoda dotazovaných nastala ve vytváření výsledkových listin, které jsou pořadateli opisovány na papír a následně z papíru zadávány na firesport.eu.

Rozdíl v odpovědích nastal při zadání otázky ,,Na jakých tratích běhají vaše kategorie?“. V polabské hasičské lize oproti podlipanské lize, z.s. týmy mužů startují pouze na 2 B hadice a 4 C.

Z dosažených odpovědí je navrhnuta aplikace na správu prezenčních listin, startovacích listin, výsledkových listin, export výsledků do CSV a na webové stránky firesport.eu.

3.1.4 Formulování požadavků

Pro aplikaci je požadováno vytvoření možnosti nastavení počtu startovních tratí, na kterých bude závislá řada funkcionalit. Aplikace bude umožňovat prezenci týmů na soutěži a tvorbu startovacích listin, které budou mí následující podmínky. V případě, že je do soutěže přihlášeno víc týmů od jednoho sboru, vždy tyto týmy budou startovat na stejné trati (základně) a pokud to bude možné, tak startovní čísla pro týmy budou s rozestupem

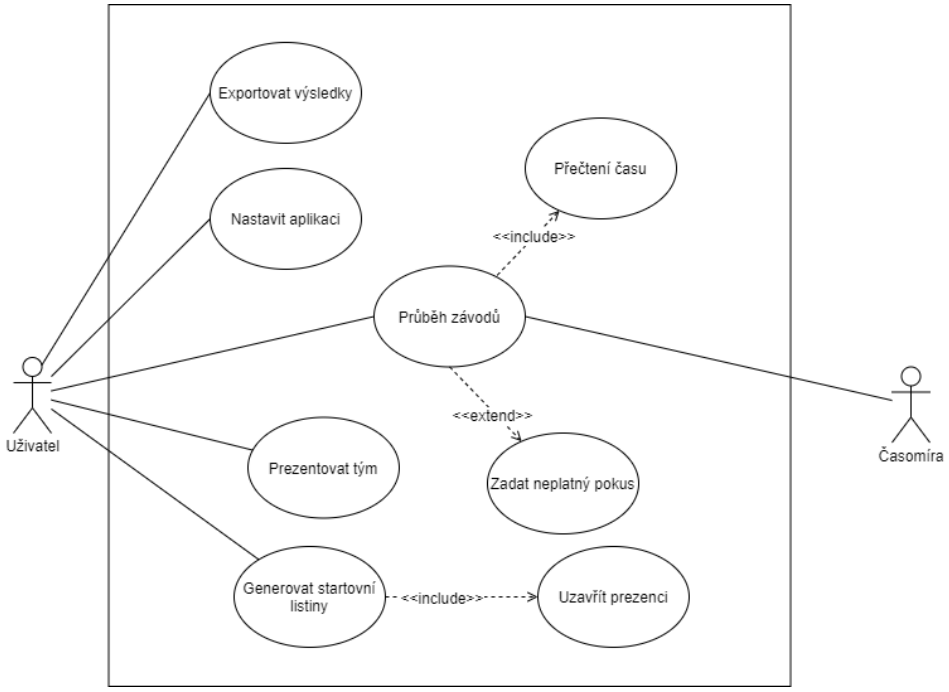
dvou dalších týmů. Pokud nebude přihlášeno dostatek týmů, aby byla tato podmínka splněna, je tým umístěn na místo ve startovním poli tak, aby nedocházelo k vypadávání startovních čísel. Výsledky bude možné exportovat do CSV a na webové stránky firesport.eu.

Tabulka 1 – Požadavků

Číslo požadavku	Popis
1.	Tvorba prezenčních listin
2.	Tvorba startovacích listin
3.	Generování startovních listin. Týmy jednoho sboru budou startovat vždy ze stejné základny a pokud to bude možné v rozestupu dvou jiných týmů.
4.	Nahrání výsledků na webové stránky firesport.eu
5.	Export výsledků do CSV

3.2 Obecný use case diagram

Obecný Use Case diagram zobrazený na obrázku 10 znázorňuje základní případy použití aplikace. Jsou v něm modelováni aktoři. Aktory aplikace jsou uživatel pracující s aplikací a hardwarová časomíra poskytující data.



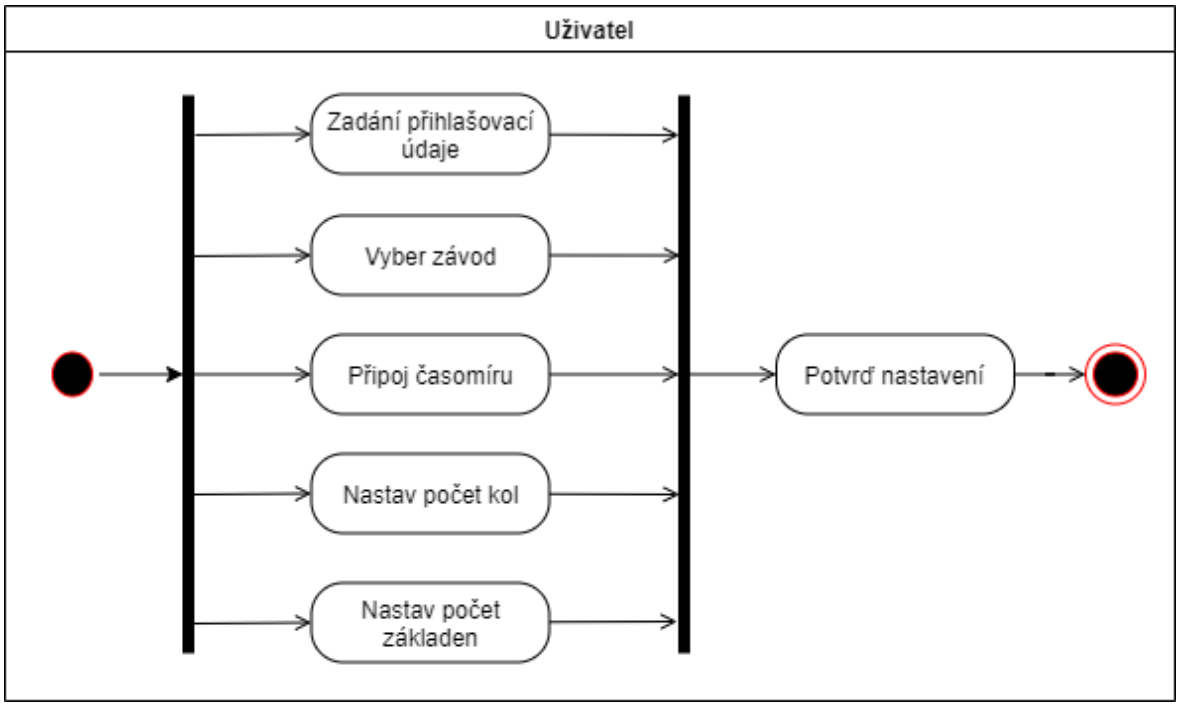
Obrázek 10 - Obecný Use Case

3.3 Aktivity diagramy

Aktivity diagramy následujících podkapitolách popisují nejdůležitější use case zobrazené na obrázku 10.

3.3.1 Nastavit aplikaci

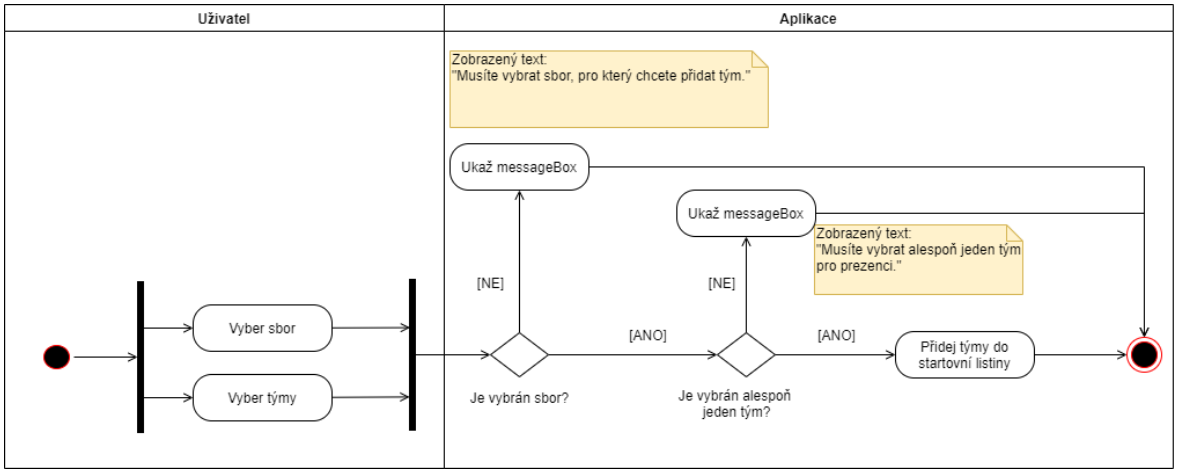
Diagram na obrázku 11 znázorňuje nastavení aplikace, které je nutné k jejímu správnému fungování. Pro nastavení je v aplikaci vytvořeno okno nastavení, kde jsou vytvořena pole pro nastavení vlastností aplikace.



Obrázek 11 - Aktivity diagram nastavení aplikace

3.3.2 Prezentovat tým

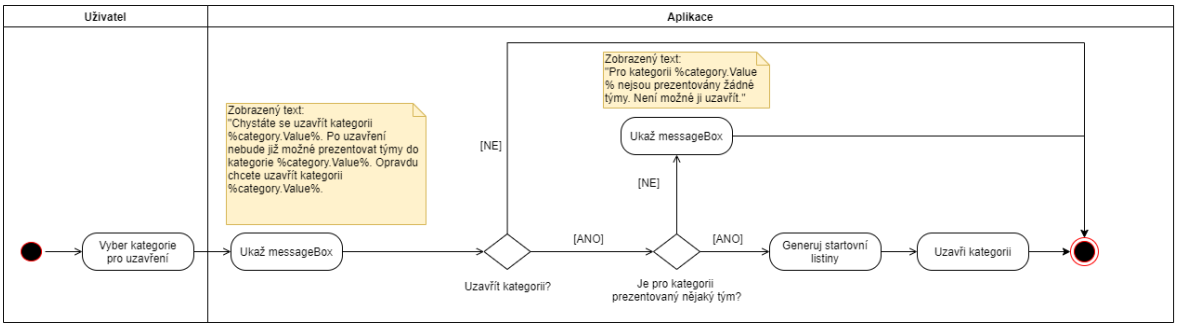
Prezentování týmu je zobrazeno na obrázku 12. Diagram znázorňuje prezentování týmů před každou soutěží. Bez prezence týmů není možné generovat startovní listiny a následně automaticky přiřazovat výsledky k týmům.



Obrázek 12 - Aktivity diagram prezence týmu

3.3.3 Generovat startovní listiny

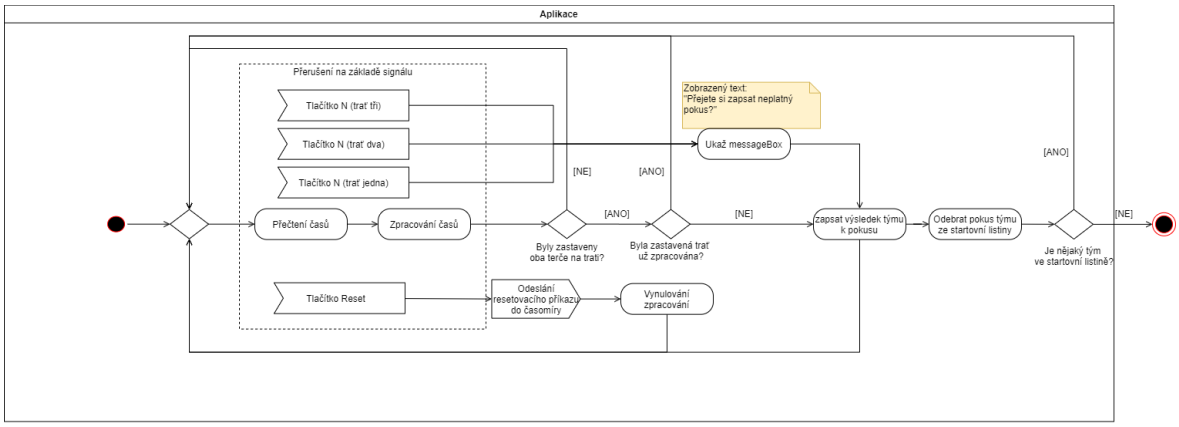
Diagram na obrázku 13 znázorňuje generování startovní listiny. Pro generování startovní listiny je nutné, aby v aplikaci byl prezentován nejméně jeden tým pro skupinu kategorií, které mají být uzavřeny. Skupiny kategorií jsou popsány v kapitole 4.6.2 Číselník kategorií. Generování startovních listin je i závislé na počtu tratí, které byly nastaveny.



Obrázek 13 - Aktivity diagram generování startovní listy

3.3.4 Průběh závodů

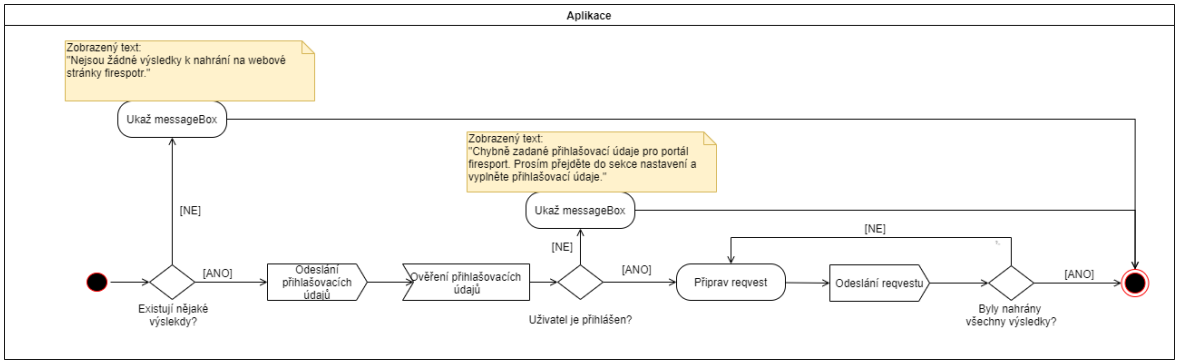
Průběh závodů je nejdůležitější use case aplikace. V rámci diagramu na obrázku 14 jsou znázorněny i interakce s uživatelem při klikání na tlačítka. Průběh závodů je závislý na vygenerování startovacích listin a připojení časomíry. Přečtené časy jsou automaticky připojovány k týmům, které se nacházejí na řadě ve startovní listině.



Obrázek 14 - Aktivity diagram průběh závodů

3.3.5 Exportování výsledků

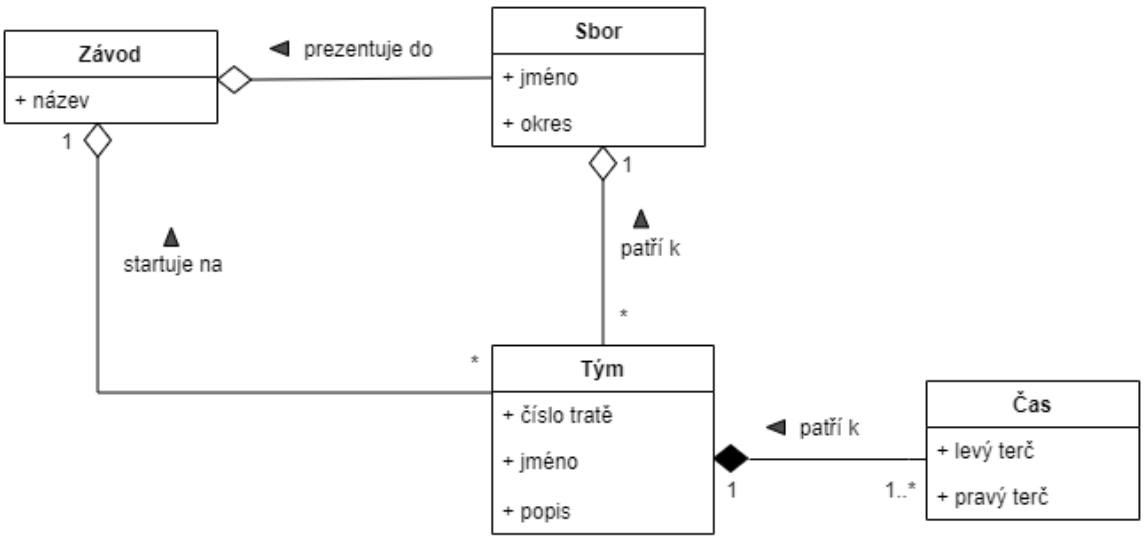
Diagram na obrázku 15 znázorňuje export výsledků na webové stránky firesport.eu. Výsledky mohou být exportovány až při dokončení use case průběhu závodů.



Obrázek 15 - Aktivity diagram exportování výsledků

3.4 Doménový model

Doménový model na obrátku 16 znázorňuje základních entity vyvíjené aplikace a vztahů mezi nimi. V modelu jsou zobrazeny entity jako závod, sbor, tým a entita pro dosažené časy týmu.



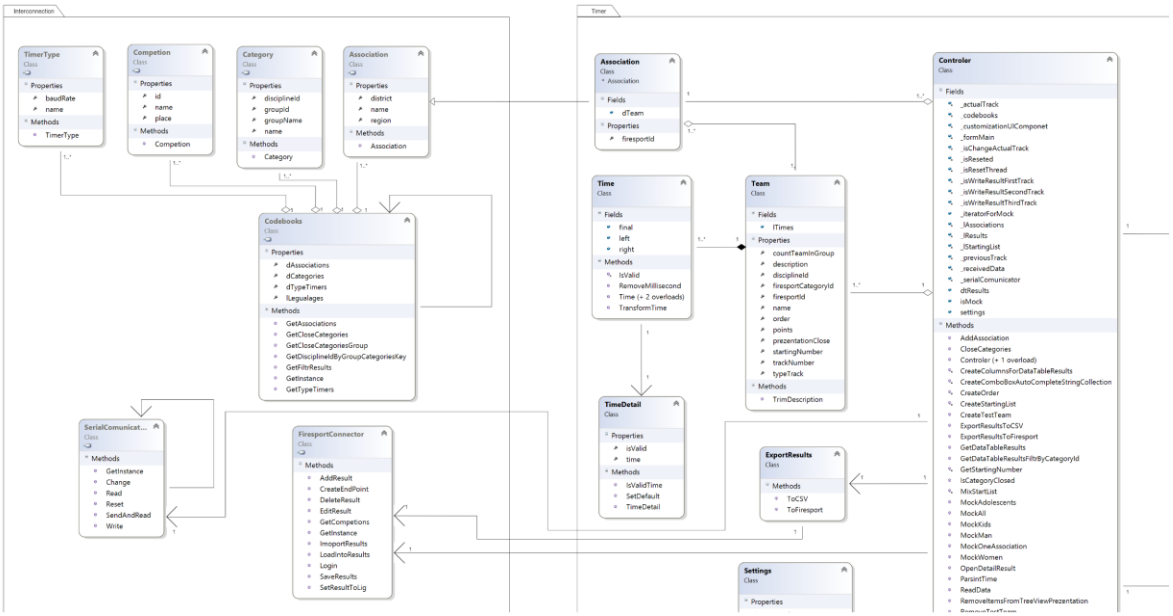
Obrázek 16 - Doménový diagram

3.5 Architektura aplikace

Aplikace se skládá z dvou částí, z hlavních aplikace Timer, která implementuje grafické rozhraní ve windows forms a práci se zadanými daty a z druhé části knihovny Interconection, která obsahuje třídu Codebooks pro práci s číselníky, třídu FiresportConnector s metodami pro práci s webovými stránkami firesport.eu a třídu SerialComunicator pro komunikaci přes sériový port s časomírami, které byly popsány v kapitole 3.2 Časomíry TVR elektronik.

3.5.1 Class diagram

Na obrázky 17 je zobrazen výřez z class diagramu aplikace Timer a knihovny Interconnection. Kompletní class diagram je přiložen jako příloha práce.



Obrázek 17 - Class diagram

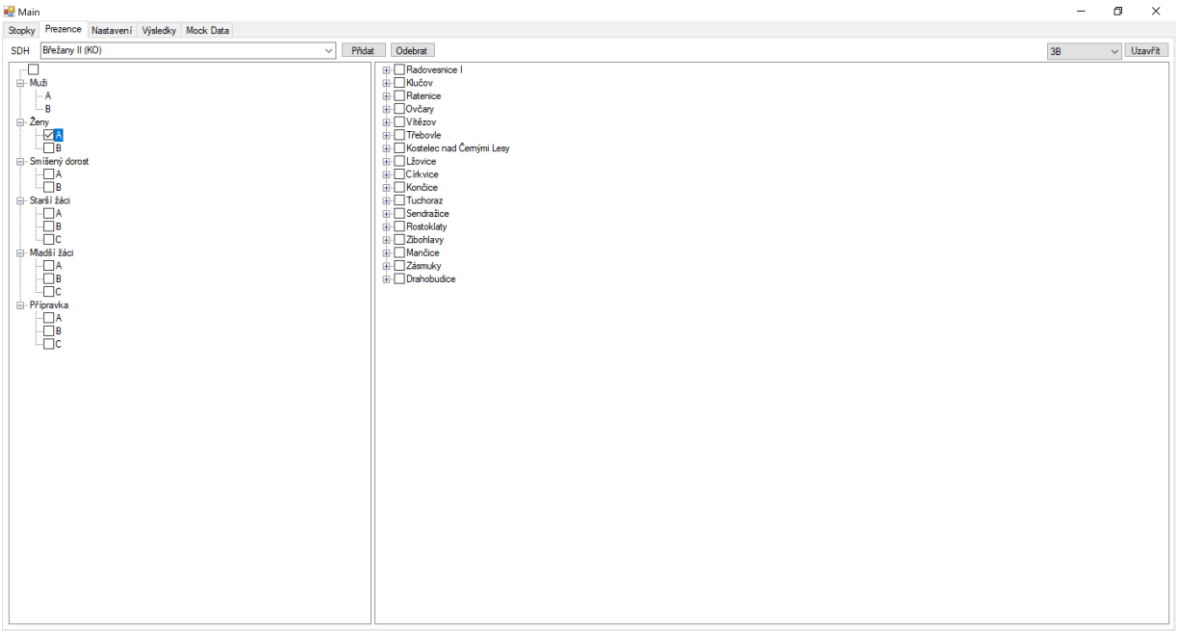
3.5.2 Timer

Hlavní část práce je implementována v aplikaci Timer. Aplikace Timer sdružuje funkcionality popsané v následujících podkapitolách.

3.5.2.1 Prezence

Prezence týmů na závodech potažmo v aplikaci je nutná událost, bez které nelze vytvořit startovací listiny. Na závodech o velkém počtu týmu a kategorií jsou týmy z jednotlivých kategorií prezentovány samostatně v průběhu dění závodů.

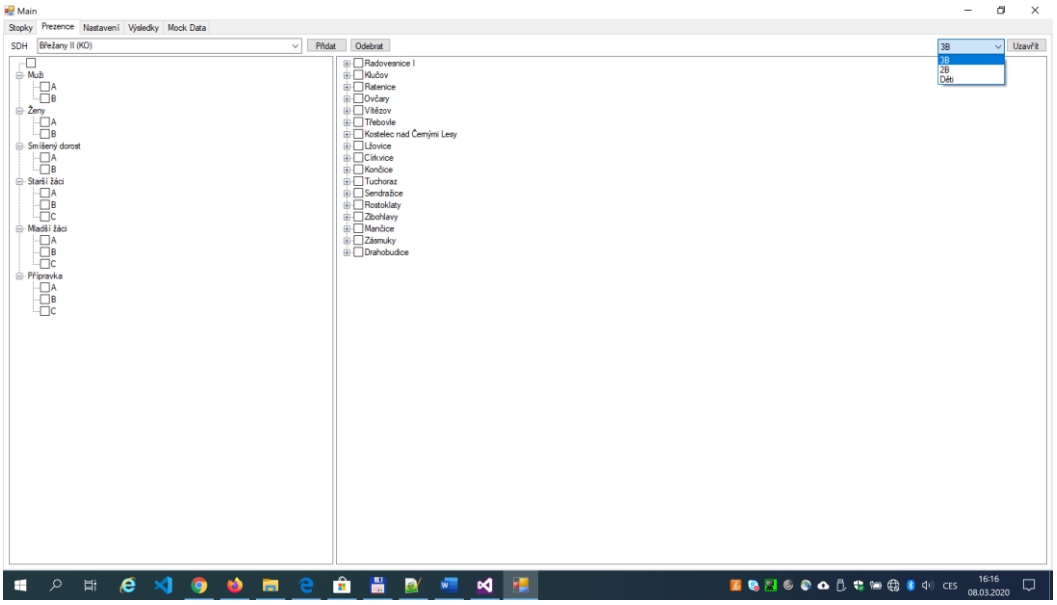
S popsaným stavem je i počítáno v možnosti prezence v aplikaci Timer, kde do uzavření kategorie je možné přidávat a odebírat prezentované týmy. Pro prezenci je nutné vybrat sbor, který prezentuje svůj tým do dané kategorie. Prezence týmů je zobrazena na obrázku 18.



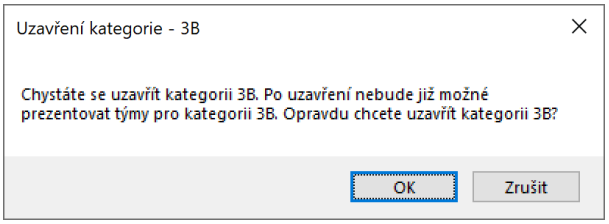
Obrázek 18 - Okno prezence týmů

3.5.2.2 Startovací listina

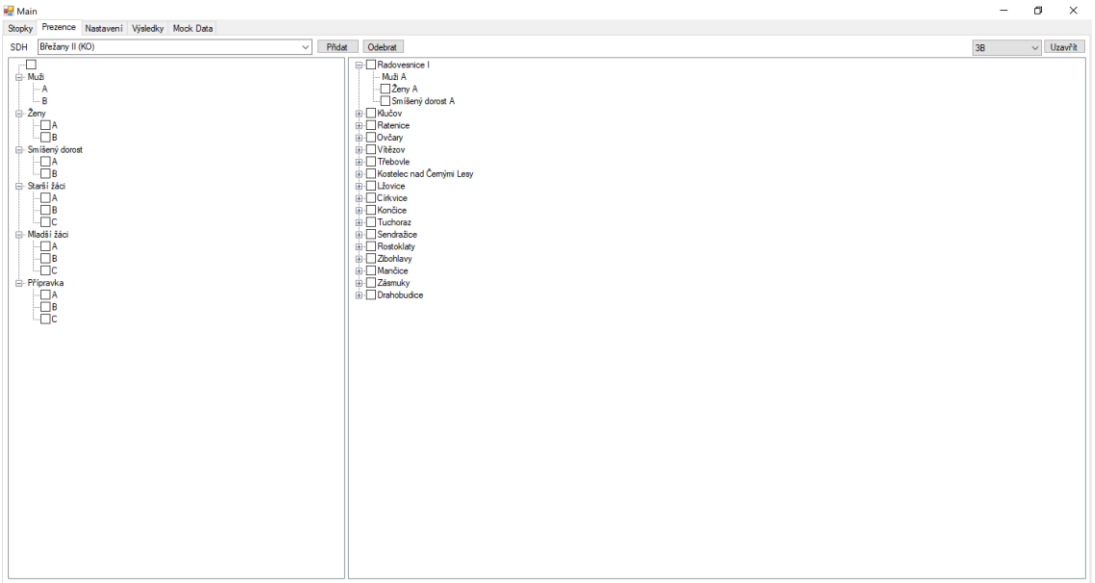
Startovní listina je generována na základně nastavení. Důležitými parametry nastavení je počet kol, ve kterých jsou závody pořádány neboli počet pokusů. Další důležitý parametr je počet tratí, na kterých je startováno. Startovní listina je generována při zavření prezence dané skupiny kategorií, které jsou blíže popsány v kapitole 4.6.2 Číselníky kategorií. Obrázek 19 zobrazuje uzavření skupiny kategorií, na obrázku 20 je zobrazena varovná hláška před uzavřením skupiny kategorií. Obrázek 21 znázorňuje změnu možnosti přidávání uzavřené kategorie, v TreeView jsou odebrány CheckBoxy, pomocí kterých se přidávají a odebírají týmy. Vygenerované startovní listiny jsou následně zobrazeny na hlavní kartě aplikace Stopky, která je zobrazena na obrázku 22.



Obrázek 19 - Generování startovních listin



Obrázek 20 - Hláška upozorňující před generování startovní listiny



Obrázek 21 - Změna TreeView po generování startovní listiny

Obrázek 22 – Okno Stopky se startovními listinami

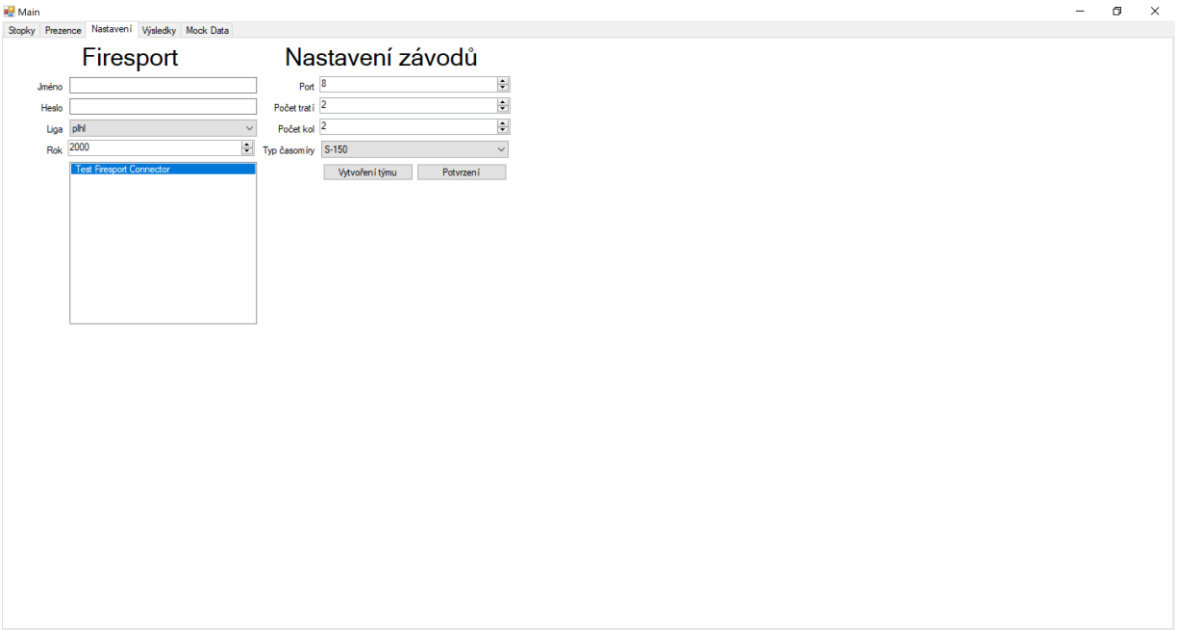
3.5.2.3 Nastavení

Záložka nastavení slouží ke konfiguraci aplikace. Je rozdělena na dvě části, nastavení pro webové stránky firesport.eu a nastavení závodů.

Pro firesport.eu se zde nacházejí konfigurační parametry pro připojení. Pro připojení je nutné zadat přihlašovací jméno a heslo. Testovací uživatel a testovací závod je popsán v kapitole 4.9 Testování aplikace. V dalších polích je vybírána liga a rok, pro který mají být načteny registrované závody. Načtení registrovaných závodů je nutné pro export výsledků z aplikace na webové stránky firesport.eu. Aby bylo možné správně nahrát výsledky na firesport.eu, je nutné, aby uživatel vybral závod, kterému mají být výsledky nahrány. Vybraný závod v aplikaci je prezentován jedinečným identifikátorem, který je nutným parametrem pro zapsání výsledků na firesport.eu.

Nastavení závodů slouží ke konfiguraci závodů dle přání uživatele. Okno nastavení je zobrazeno na obrázku 23. Je zde možné nastavit na kolik kol budou dané závody pořádány neboli kolik bude umožněno týmům startovních pokusů. Standartně dle pravidel požárního sportu jsou na závodech povoleny dva pokusy, ale je řada soutěží, kde je pouze jeden pokus. Maximální počet pokusů je 4. Dále je zde možné nastavit, na kolik tratí daná soutěž bude pořádána. Pro nastavení komunikačního portu s časomírou je umístěn NumericUpDown. Je potřeba vybrat číslo portu, ke kterému je připojena časomíra k PC a pole pro vybrání typu časomíry, které jsou popsány v teoretické části práce.

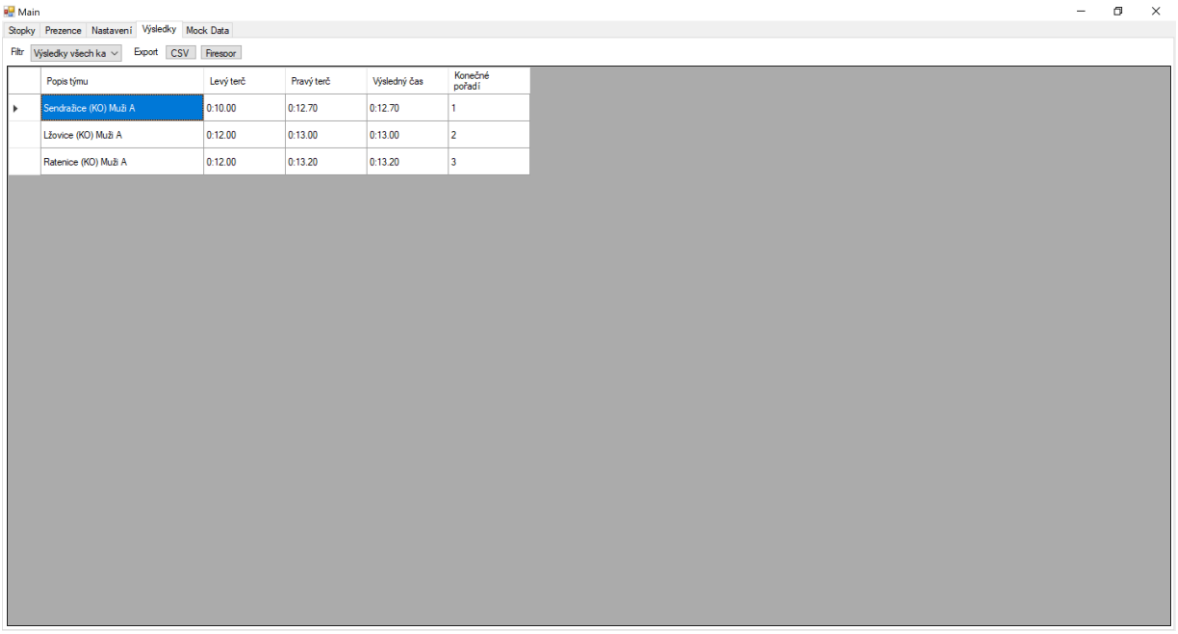
V nastavení se nachází i tlačítko pro vytvoření testovacího týmu v případě problému s časomírou, kdy by bylo nutné do startovních listin vložit testovací start. Tato funkcionalita je popsána v kapitole 4.9.5 Zkušební start.



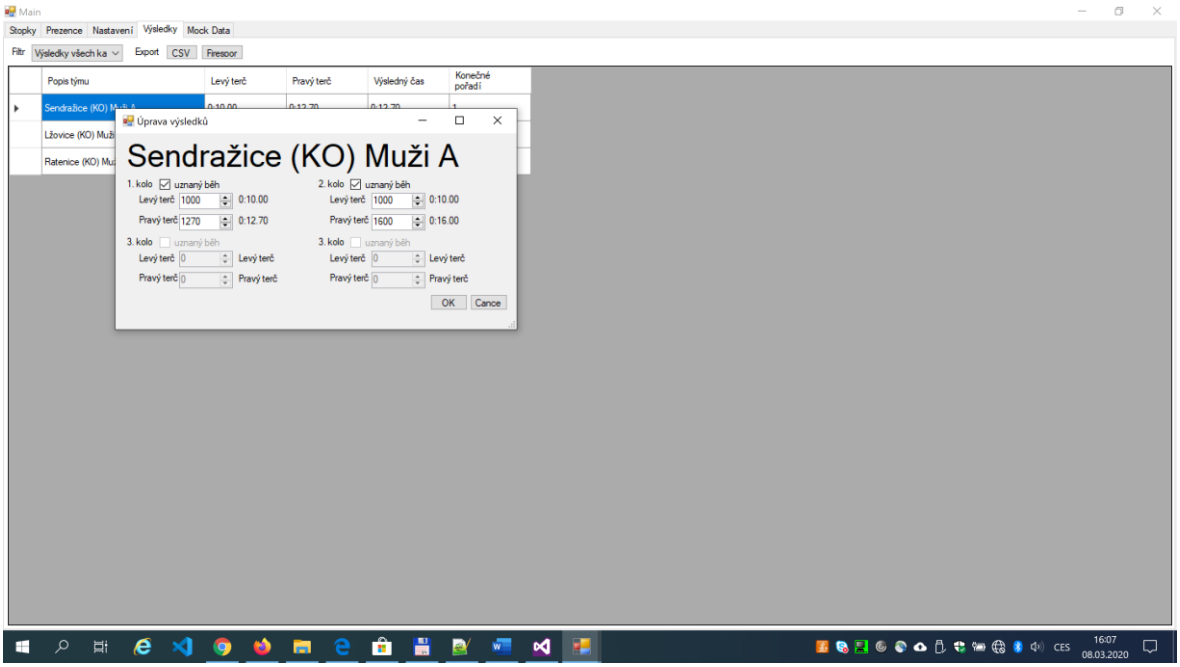
Obrázek 23 - Okno nastavení aplikace

3.5.2.4 Zobrazení výsledků

Všechny dosažené výsledky jsou zobrazeny na stejnojmenné záložce výsledky, která je prezentována obrázkem 24. Výsledky je možné filtrovat podle kategorie, pro kterou byly dosaženy. Výsledky jsou automaticky přiřazovány k týmům dle pořadí ve startovací listině. Pro zobrazení výsledků je použita komponenta DataGridView. První vykreslený sloupec na komponentě DataGridView je „Popis týmu“, který zobrazuje sdružené vlastnosti z instance Teamu. Ve druhém sloupci je zobrazen dosažený čas levého terče, ve třetím sloupci dosažený čas pravého terče. Čtvrtý sloupec zobrazuje výsledný čas a v posledním sloupci je zobrazeno aktuální pořadí v soutěži, které je vždy přepočítáváno při zadání nového výsledku. V případě, že je soutěž běhána na více kol, je vždy zobrazen nejlepší pokus v rámci kol. Klikem na výsledky je otevřeno okno pro úpravu výsledků, které je zobrazeno na obrázku 25.



Obrázek 24 - Okno všech výsledků



Obrázek 25 - Okno detailu výsledků týmu

3.5.2.5 Ukázka kódu

Ve zdrojovém kódu aplikace je velmi často použit jazyk LINQ s využití možností Lamda kalkulu pro vybrání, seřazení a následnou transformací kolekcí. Na obrázku 26 je znázorněno vybrání instancí týmů ze startovní listiny pro vypsaní do jednotlivých grafických komponent aplikace.

```
public void WriteStartingList(int groupCategoriesKey)
{
    int disciplineId = this._codebooks.GetDisciplineIdByGroupCategoriesKey(groupCategoriesKey);
    WriteStartingList(this._lStartingList.Where(team => team.trackNumber == 1 && team.disciplineId == disciplineId && team.order == 0)
        .OrderBy(team => team.startingNumber).ToList(), this._formMain.listBoxFirstTrack);
    if (this.settings.trackCount >= 2)
        WriteStartingList(this._lStartingList.Where(team => team.trackNumber == 2 && team.disciplineId == disciplineId && team.order == 0)
            .OrderBy(team => team.startingNumber).ToList(), this._formMain.listBoxSecondTrack);
    if (this.settings.trackCount >= 3)
        WriteStartingList(this._lStartingList.Where(team => team.trackNumber == 3 && team.disciplineId == disciplineId && team.order == 0)
            .OrderBy(team => team.startingNumber).ToList(), this._formMain.listBoxThirdTrack);
}
```

Obrázek 26 - Ukázka použití LINQ

3.5.3 Interconnection

Knihovna Interconnection byla vytvořena z důvodu použitelnosti dále popisovaných tříd i v jiných projektech. Pro Implementaci tříd FiresportConnector, SerialComunicator a Codebooks je použit návrhový vzor singleton neboli jedináček. Tento návrhový vzor byl zvolen z důvodu zajištění pouze jedné instance zmíněných tříd.

Je předpokládáno využití třídy FiresportConnector pro již hotová řešení jiných časomír. Třídu SerialComunicator bude možné použít i v jiných projektech pro programování aplikací k časomírám TVR elektronik.

V České republice se nacházejí i hasičské ligy, které používají časomíry vlastní výroby, které mají implementované své ovládací rozhraní v počítači. Jeden z důvodů byl vytvoření knihovny i pro integraci již hotových nebo nově vznikajících řešení časomír na webové stránky firesport.eu.

Knihovna obsahuje dále třídu SerialComunicator, která má implementované metody pro komunikaci časomír TVR elektronik přes sériový port. Použitelnost třídy je i v dalších tvořených aplikacích.

3.5.3.1 Ukázka kódu

V knihovně Interconnection je pro řadu tříd využito návrhového vzoru singleton. Jeho konstrukce byla popisována v teoretické části práce v kapitole 3.6.1 Jedináček. Na obrázku 27 je ukázka kódu z knihovny, přesněji třídy FiresportConnector, která zmíněný návrhový vzor využívá. Třída má privátní konstruktor, který je volán ze statické metody GetInstance v případě, že není vytvořena instance třídy FiresportConnector.

```
namespace Interconnection
{
    public class FiresportConnector
    {
        private static FiresportConnector instance = null;
        private HttpClient _client;
        private string _endPoint;
        private CookieContainer _cookieContainer;
        private HttpClientHandler _handler;
        private Uri _baseAddress;

        private FiresportConnector(string domain)
        {
            SetVariables(domain);
        }

        public static FiresportConnector GetInstance(string domain)
        {
            if (instance == null)
                instance = new FiresportConnector(domain);
            else
                instance.SetVariables(domain);

            return instance;
        }
```

Obrázek 27 - Ukázka kódu

3.5.4 Rozdělení výkonu

Pro rozdělení výkonu aplikace je použito programování ve více vláknech. V hlavním vlákně aplikace běží především funkcionality, které potřebují interakci s uživatelem přes grafické rozhraní, jako je tvorba prezenčních listin, startovacích listin export výsledků. Z hlavního vlákna aplikace je zpuštěno druhé vlákno, které čte data ze sériového portu a pomocí delegáta předává data do hlavního vlákna. Přečtená data jsou transformována do objektů Time a přidána k týmu, který je právě v pořadí na startovní listině pro danou trať.

3.6 Číselníky

Základním konfiguračním prvkem aplikace jsou vytvořené číselníky, které jsou nutné pro běh aplikace Timer a správné fungování knihovny Interconnection. Sloupce jsou v číselnících odděleny znakem středník. Z důvodu optimalizace aplikace pro číselníky byl použit formát CSV.

Pro vytvoření číselníků byl zvažován i formát XML. Třída XmlSerializer, která je dostupná v .NET frameworku, snadně serializuje a deserializuje instance. Tento formát nebyl použit z důvodu zvýšení náročnosti načtení dat, kde by bylo nutné minimálně načíst o 7+(24*n) znaků navíc s předpokladem velikosti všech tagu XML o jednom jmenném znaku. Proměnná „n“ zastupuje počet načítaných sborů z číselníku. Pro správný běh aplikace není nutné v číselníku mít umístěny všechny sbory v České republice, ale pouze předpokládané sbory, které se budou účastnit soutěží. Většina lig před svým zahájením vytváří seznam účastněných sborů na soutěžích viz obrázek 23 a proto není nutné načítat všechny sbory do aplikace. V případě načítání všech možných sborů v České republice, kterých je 7 662, by bylo nutné načíst o 183 895 znaků více. [27]

SDH	OSH	muži	ženy	dorost	žáci		přípravka
					starší	mladší	
Bulánka	KO						
Dobré Pole	KO						
Dolní Břežany	P-Z						
Dolní Měcholupy	PHA						
Doubek	P-V						A, B
Horní Kruty	KO						
Horní Měcholupy	PHA						
Klučov	KO	A, B				A, B, C	
Kostelec n/Č lesy	KO					A, B	
Křížkový Újezdec	P-V						
Kšely	KO						
Lány	KL						
Nové Jirny	P-V						
Ovčáry	KO					A, B, C	
Radovesnice I	KO						
Ratenice	KO						
Rostoklaty	KO						
Třebovle	KO					A, B	
Zásmuky	KO						
Zibohlavy	KO					A, B	
Počet družstev		12	8	6	11	18	6

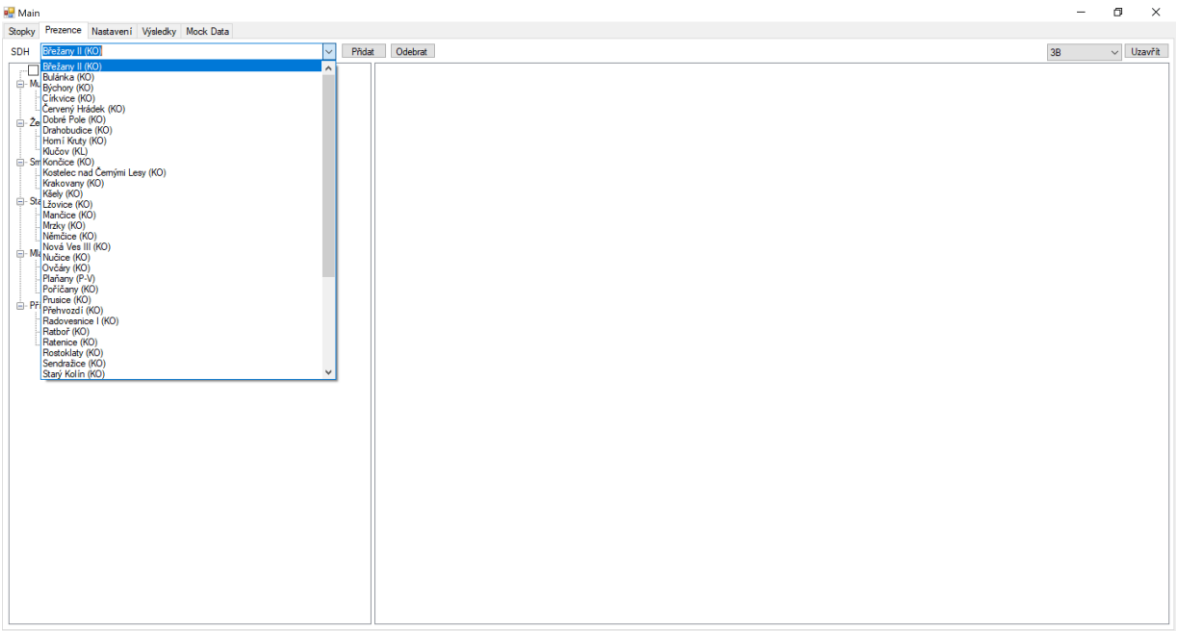
Obrázek 28 - Evidence družstev [14]

3.6.1 Číselník sborů - CB_Associations

Jednotlivé sbory jsou popsány v číselníku CB_Associations. Význam sloupců v číselníku je následující. První pozice určuje jedinečný identifikátor pro insert výsledků na firesport.eu neboli ve třídě Associations vlastnost firesportId. V dalším sloupci je název sboru. V posledním sloupci zkratka okresu, kde je sbor registrovaný. Okres je nutné rozlišit z důvodu přesnějšího určení sboru. V České republice se nachází řada sborů stejných jmen. Nejznámějšími jmenovci mezi sbory je Nová Ves.

Číselník neobsahuje všechny možné sbory v České republice, ale pouze určitou testovací množinu sborů z okresu Kolín, která byla tvořena především ze sborů účastnících se Podlipanské ligy, z.s. a Polabské hasičské ligy.

Příklad dat z číselníku „147397;Sendražice;KO“. Na obrázku 29 je znázorněno mapování dat z číselníku do ComboBoxu aplikace na záložce Prezence.

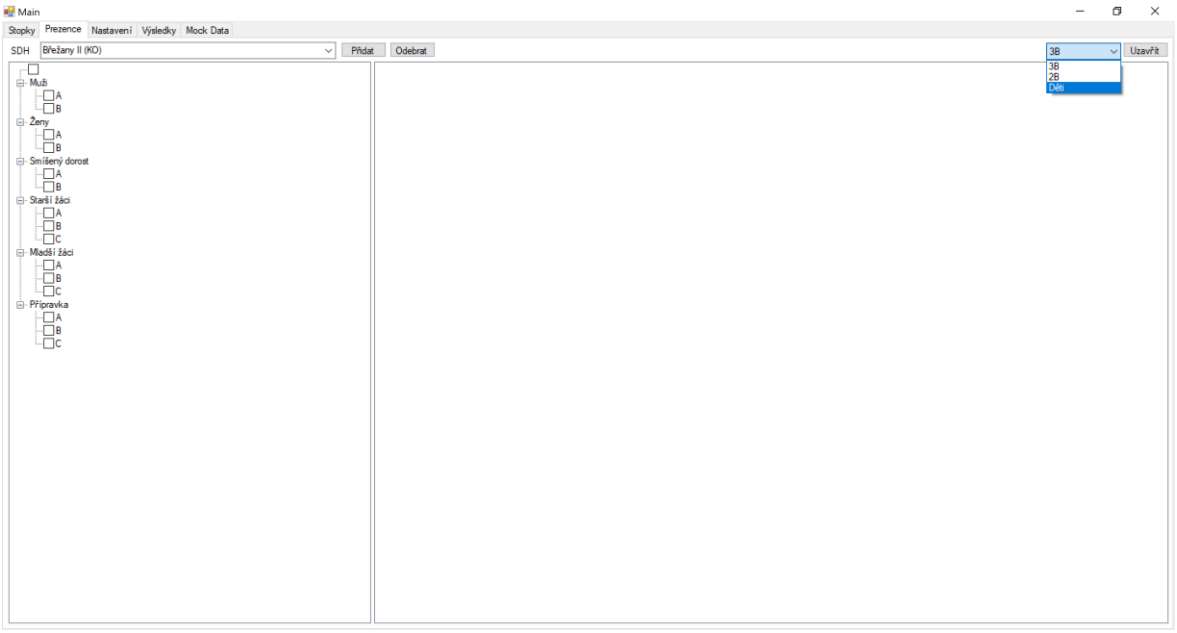


Obrázek 29 - Mapování číselníku obsahující sbory do aplikace

3.6.2 Číselník kategorií - CB_Categories.csv

Číselník kategorií definuje kategorie typy závodů, které je možné na daných závodech vyplňovat. Variabilita lig je velmi rozsáhlá. Řada lig má ve svých pravidlech úpravy pro plynulost průběhu závodů či atraktivnost závodu pro diváka. Je tedy nutné dynamicky tyto údaje měnit.

Číselník má následují strukturu. První sloupec určuje, zda je řádek načten do komponet aplikace. Jsou možné dvě hodnoty ve sloupci Y a N. Y značí načtení do aplikace, N značí nenačtení do aplikace. V dalším sloupci je hodnota firesportCategoriId, které je nutná k insertu výsledků na webové stránky firesport.eu. Následují sloupec je textová reprezentace číselné hodnoty. Například hodnota 1 se rovná kategorii mužů. Třetí sloupec označuje groupId. Sloupcem je možné nastavit skupiny kategorií, které běhají na stejných tratích. Jsou tedy pak generovány smíšené startovní listiny kategorií. Funkcionalita je vhodná především pro dětské kategorie. Následující sloupec je hodnota zobrazená při uzavíraní kategorií. V posledním sloupci je id typu soutěže kategorie neboli competionId. Id vychází z číselníku CB_CompetionTypes. Obrázek 30 ve svém pravém rohu zobrazuje nastavené skupiny kategorií. V komponentě ComboBox je zobrazen název skupiny kategorií, ale aplikace pracují s její kódovou hodnotou.



Obrázek 30 - Mapování skupin kategorií z číselníku do aplikace

3.6.3 Číselník typu soutěží - CB_CompetionTypes.csv

Číselník typu soutěží je rozšiřujícím číselníkem pro číselník CB_Categories.csv. Je nutný pro správné nahrání výsledků na webové stránky firesport.eu. V prvním sloupci je obsaženo id typu soutěže. Toto id určuje jedinečnost typu soutěže. Hodnota značí variabilitu typu soutěže, které jsou na různých ligách rozdílné. Následující sloupec

zobrazuje význam v textové reprezentaci. Nej častější rozdíl na závodech je použití sklopných nebo nástřikových terčů. Další rozdíl může nastat v typu povolených požárních hadic. Soutěže mohou být pořádány pro takzvanou širokou sadu, úzkou sadu a super úzkou sadu, kde mezi hadicemi je vždy rozdíl jednoho centimetru na průměru hadice. Pro všechny možné kombinace je vytvořen číselník typu soutěží.

3.6.4 Číselník lig - CB_Leagues.csv

Číselník obsahuje seznam lig, pro které je možné nahrávat výsledky na webové stránky firesport.eu. Zpravidla by měla být nastavena právě jedna hodnota. První sloupec povoluje nebo zakazuje nahrání hodnot do komponent aplikace ve stejné logice jako byla popsána v kapitole 4.6.2 Číselník kategorií. Následující sloupec označuje část domény ligy, pro kterou budou nahrávány výsledky na webové stránky firesport.eu.

Zobrazení ligy na webových stránkách firesport.eu je tvořeno pomocí domén třetího řádu, které reprezentuje tento sloupec. Například informace o Polabské hasičské lize jsou dostupné pod url https://plhl.firesport.eu/.

3.6.5 Číselník typů časomír - CB_TypeTimers.csv

Číselník typů časomír obsahuje nastavení pro typy časomír, které byly touto prací analyzovány. První sloupec opět obsahuje hodnotu, zda má být daný typ časomíry zobrazen v komponentách aplikace. Druhý sloupec určuje jedinečný identifikátor časomíry. Následující sloupec reprezentuje název časomíry, který je zobrazen v aplikaci. Poslední sloupec obsahuje hodnotu modulační rychlosti pro nastavení komunikace přes sériový port.

3.7 Datová komunikace přes sériový port

Pro zjištění datové komunikace časomír s počítačem po sériové lince bylo použito reverzního inženýringu pomocí programu Serial Port Monitor. Jedná se o nástroj s funkcemi, které umožňují sledovat a zaznamenávat veškerá data procházejícími sériovými porty počítače s možnostmi nastavení různé modulační rychlosti. Komunikace časomír byla analyzována pro tři různé typy.

Model S-150 byl nejstarší výroby z analyzovaných modelů časomír. Pro komunikaci používá modulační rychlost 9600 Bd a kódování iso-8859-1. Z časomíry je posílán po sériové lince následující řetězec „2:0:2:0:2:0:2:0:2:0:2:0:2:0“.

Druhým analyzovaným modelem byla časomíra DUAL-150Z. Model časomíry využívá stejné kódování iso-8859-1 jako časomíra S-150, ale pro komunikaci používá modulační rychlost 115200 Bd. Z časomíry je posílán po sériové lince následující řetězec „2:0:2:0:2:0:2:0:2:0:2:0:2:0“.

Tabulka 2 zobrazuje význam jednotlivých pozic v posílaném datovém řetězci z časomír S-150 a DUAL-150Z. Elektronika časomír není tvořena pouze pro požární sport, ale je využívána i k dalším sportovním disciplínám, jako je vozatajství nebo sjezdové lyžování. Proto je z časomír přenášeno více informací, než je potřeba pro požární sport. Časomíry neumí paralelně odstartovat dvě trati na jednou, vždy je možné pro požární útok odstartovat pouze jednu trať. Požární útok v časomíře se vyznačuje hodnotou 2 na pozici 1. Hodnoty běžícího času jsou přenášeny na pozicích 2 až 7. V případě, že je časomíra přepnuta do módu pro běh na sto metrů překážek na první pozici je hodnota 4 a jsou přenášeny časy pro čtyři tratě na pozicích 2 až 11.

Tabulka 2 – Kódových hodnot

Model S-150 a Model DUAL-150Z			
Pozice	Význam	Hodnota při běhu času	Hodnota při zastavení času
1	disciplína	2 nebo 4	2 nebo 4
2	konečný čas	0-n	1-n
3	konečný stav	2	8
4	čas spínače 1	0-n	1-n
5	stav spínače 1	2	8
6	čas spínače 2	0-n	1-n
7	stav spínače 2	2	8
8	čas spínače 3	0-n	1-n
9	stav spínače 3	2	8
10	čas spínače 4	0-n	1-n
11	stav spínače 4	2	8
12	Volná pozice	0	0
13	Volná pozice	0	0
14	konečný stav	2	8

Posledním analyzovaným modelem časomíry byl typ DUAL-150Z – Triple, který je speciálně vytvořen pro potřeby Podlipanské ligy, z.s. Model časomíry vychází

z konstrukce DUAL-150Z. Kódování a modulační rychlost jsou stejné jako u předchozího zkoumaného modelu časomíry. Přenášený datový řetězec je stejný jako u předchozích časomír. Rozdíl v časomírách je přidáním dalšího konektoru pro pistoli a dvou konektorů pro terče.

3.8 HTTP komunikace

Webové stránky firesport.eu byly z puštěny již v roce 2000. Pro jejich vývoj byl použit programovací jazyk PHP. Aktuální používaná verze PHP je 5.3.29. bez použití jakéhokoli frameworku. Webové stránky neobsahují žádné standardizovaný API. Data je možné nahrávat pouze přes vytvořené formuláře na stránce. Veškeré zkoušené interakce byly oznámeny administrátorovi Bc. Janu Adamcovi. Po osobní dohodě byl vytvořen testovací uživatel a testovací závod z roku 2000 řazený do Polabské hasičské ligy tak, aby nedošlo k ovlivnění provozu firesport.eu.

Pomocí prohlížeče Google Chrome a jeho vývojových nástrojů byly odchyceny posílané požadavky (reqesty) z vyplňovaných formulářů. Pomocí aplikace Postman byly sestaveny vzorové požadavky, které byly použity pro testování.

Veškeré požadavky jsou posílány na end point https://plhl.firesport.eu. Doména třetího řádu určuje typ ligy, pro kterou jsou zadané požadavky získány. Změnu ligy je možné konfigurovat viz kapitola 4.6.4 Číselník lig.

3.8.1 Získání závodů daného ročníku

Pro získání závodů daného ročníku je vytvořena metoda GetCompetions v knihovně Interconnection ve třídě FiresportConnector. Metoda zavolá na webovém serveru url, kde je umístěn soubor web_souteze.php. Vstupní parametr je rok, pro který mají být vypsány zadané soutěže na webové stránce firesport.eu. Po zavolání url je vrácena html stránka s vypsanými soutěžemi, a především jejich jedinečnými identifikátory, které jsou nutné ke správnému vložení výsledků popsaných v kapitole 4.8.3 Vkládání výsledků.

Po parsing jsou použity třídy .Net Frameworku. Vrácená data jsou vložena do instance třídy HtmlDocument, do vlastnosti DocumentNode, která je typu HtmlNode. Třída HtmlNode má implementovánu metodu SelectNodes, která má vstupní parametr cestu k hledanému nodu v html stromu. Vrácen je objekt HtmlNodeCollection, který již

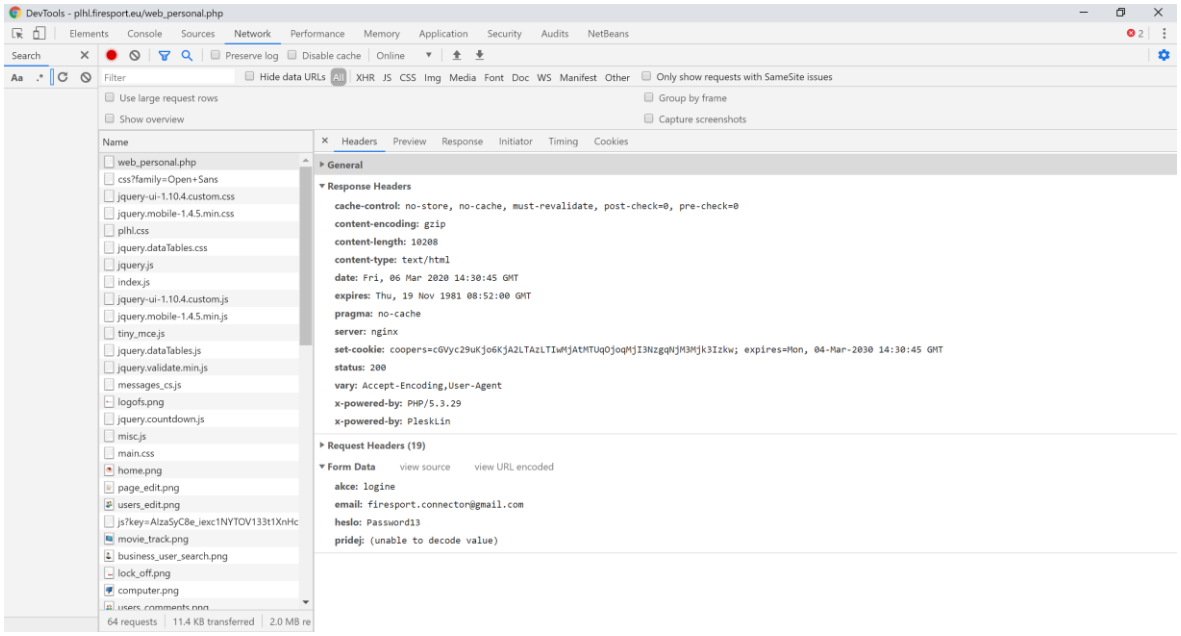
obsahuje kolekci nodu s informacemi o závodech, které je nutné ještě transformovat do kolekce závodů neboli do List<Competion>.

3.8.2 Přihlášení

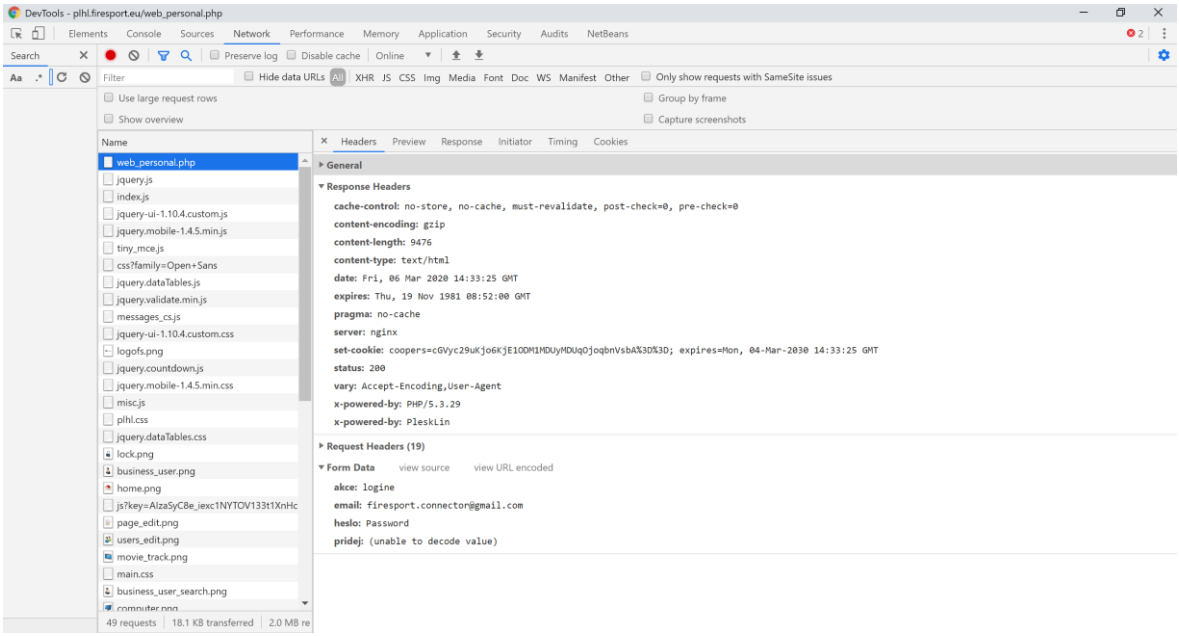
Pro nahrávání výsledků je potřeba, aby uživatel byl na stránce firesport.eu přihlášený. Přihlašovací formulář se skládá z pole pro zadání emailu a pole pro zadání hesla. Pro poslání požadavku na server je použita metoda POST. Zpracování na serveru probíhá souborem web_personal.php.

Před odesláním požadavku je nevhodně pracováno se zadaným heslem, které v požadavku je posíláno ve formátu prostého text. Pro heslo není použit žádný typ šifrování jako je například MD5 či jiné. Na tuto skutečnost byl upozorněn administrátor webových stránek jako na bezpečnostní chybu. Zmíněná chyba je zobrazena na obrázku 31.

Při úspěšném i neúspěšném přihlášení je v set-cookie vrácen coopers, který obsahuje hexadecimální řetězec. Tento řetězec je nutné odesílat pro všechny požadavky odesílané z formulářů jako přihlášený uživatel. V případě, že coopers končí řetězcem „%3D%3D”, nepodařilo se uživatele přihlásit. Byly zadány nesprávné přihlašovací údaje viz obrázek 32.



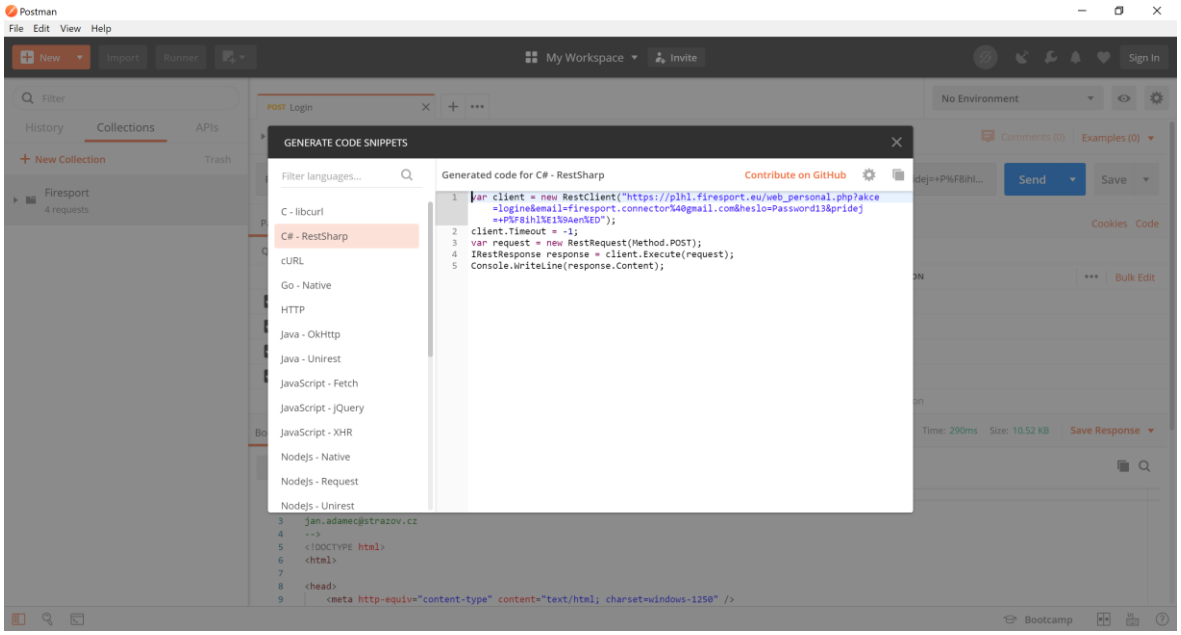
Obrázek 31 - Požadavek úspěšného přihlášení



Obrázek 32 - Požadavek neúspěšného přihlášení

V aplikaci Postman byly vytvořeny vzorové požadavky, ze kterých lze vygenerovat kód do jazyka C# viz Obrázek 33. Vygenerovaný kód, ale očekává standardizované REST API. Kód tedy nebylo možné použít.

V knihovně Interconnection ve třídě FiresportConnector pro implementaci metody Login bylo vycházeno z posílaného požadavku v následujícím formátu „https://plhl.firesport.eu/web_personal.php?akce=logine&email=firesport.connector%40gmail.com&heslo=Password13&pridej=+P%F8ihl%E1%9Aen%ED“ metodou POST.

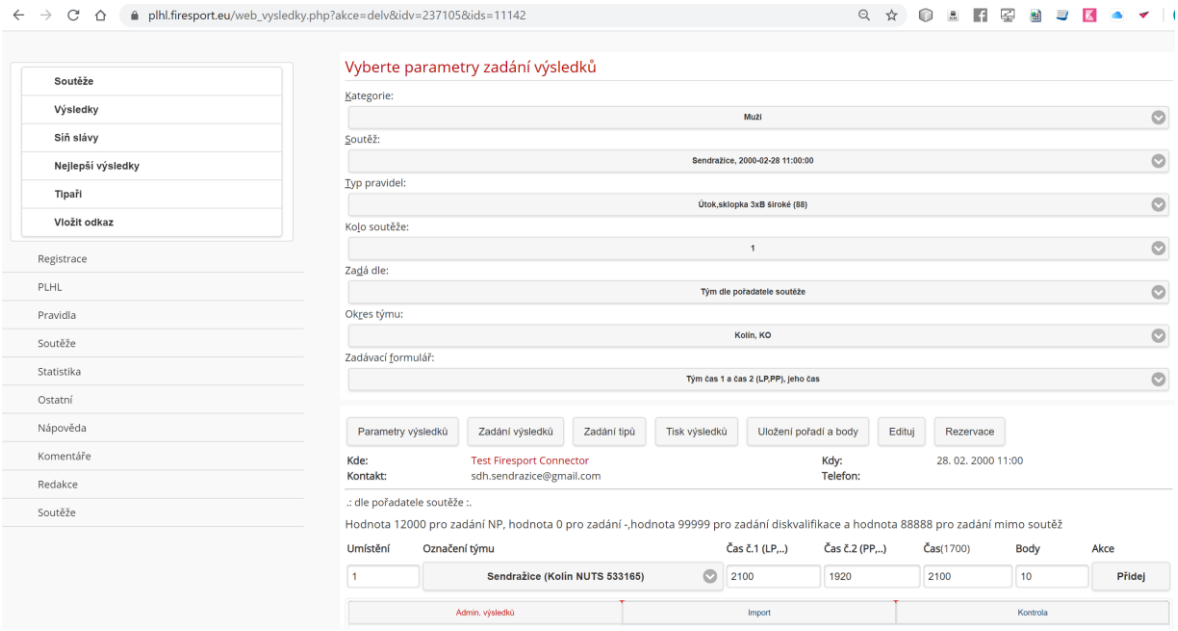


Obrázek 33 - Postman generování kódu

3.8.3 Vkládání výsledků

Vkládání výsledků na webové stránky firesport.eu, je velmi specifické. Pro zadání výsledků je nutné zadat kategorii, soutěž, typ pravidel neboli typ soutěže, čas levého terče, čas pravého terče, výsledný čas, bodový zisk, pořadí a další parametry. Řada parametrů má svá kódová zastoupení, která jsou předpřipravena v číselnících, které jsou popsány v kapitole 4.6 a jejich podkapitolách.

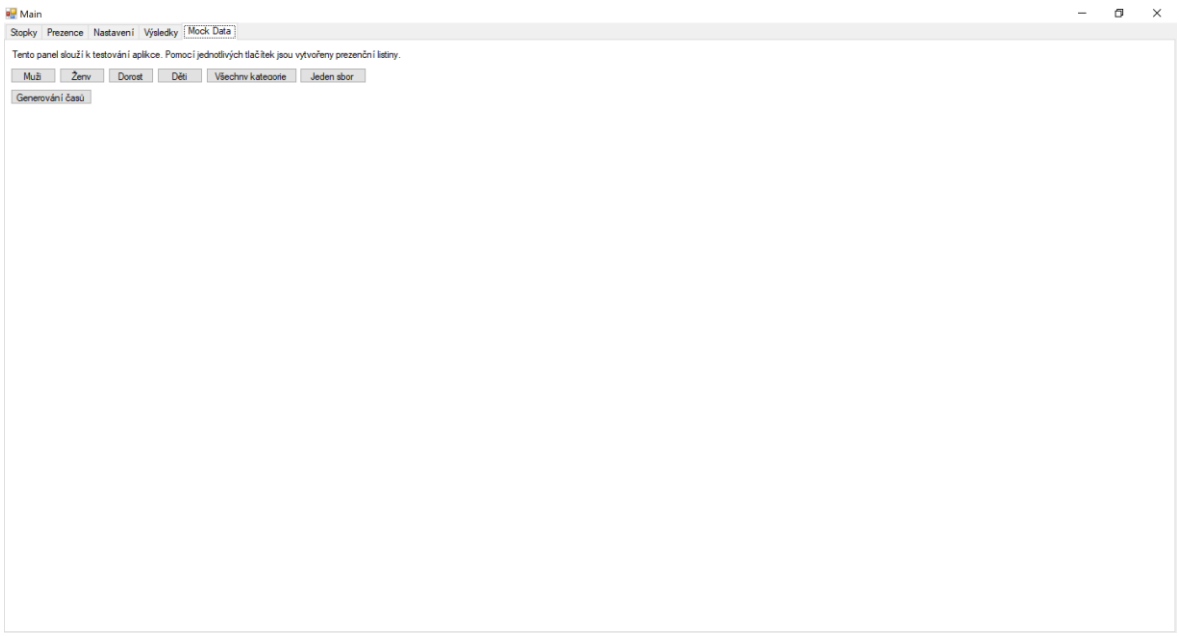
Nevhodnost vytvořeného formuláře je patrná pro parametry pořadí, bodového zisku nebo výsledného času. Všechny zmíněné parametry je možné vypočítat na základě vyplnění dosaženého času levého a pravého terče. Autor formuláře pro vkládání dat si chtěl ulehčit výpočet těchto parametrů, tak odpovědnost za vyplnění dat přenesl na uživatele viz obrázek 34. V knihovně Interconnection ve třídě firesportConnector vytváří metoda AddResult požadovaný reqvest včetně duplicitních parametrů.



Obrázek 34 - Formulář pro vkládání výsledků

3.9 Testování aplikace

Pro testování aplikace byla vytvořena záložka Mock Data, zobrazená na obrázku 35, kde jsou předpřipraveny tlačítka pro různé případy vytvoření prezenčních listin. Dále se zde nachází tlačítko pro generování časů z důvodu, aby bylo možné aplikaci testovat i bez připojené časomíry. Vygenerované časy jsou totožné pro každý běh generování. Pro testování nahrání výsledků na webové stránky firesport.eu byl vytvořen uživatel „firesport.connector@gmail.com“ s heslem „password13“. K testování byl vytvořen i závod pro Polabskou hasičskou ligu. Pro nalezení závodu je potřeba vybrat ligu „plhl“ a rok 2000. V případě, že počítač není připojen k internetu, je v ListBoxu, který zobrazuje závody pro daný ročník, je vypsána hláška „Nejste připojeni k internetu. Zkontrolujte své připojení“.



Obrázek 35 - Obrazovka s připravenými testovacími daty

Aplikace byla testována pomocí následujících testovacích scénářů, které mohou nastat při závodech v požárním útoku.

3.9.1 Vytvoření prezence týmů všech kategorií u sboru

Vstup

Uživatel aplikace se v horním menu přepne do sekce „Prezence“. Vybere sbor Sendražic z okresu Kolín. Ve stromu kategorií vybere všechny kategorie pro prezenci. Po vybrání klikne na tlačítko „Přidat“. Alternativou testu může být použití panelu s Mock Daty, kde je k testu připraveno tlačítko „Jeden Sbor“.

Výstup

Aplikace zobrazí v pravé části okna sbor se všemi vybranými kategoriemi.

3.9.2 Postupné přidávání týmů kategorií u sboru

Tento testovací scénář je velmi častý při pořádání závodů, kdy dochází k postupnému ukončování prezence jednotlivých kategorii. Velké sbory zvlášť prezentují kategorie dětských týmů a týmy dospělých kategorií.

Vstup

Uživatel v sekci „Prezence“ vždy přidá pouze jednu kategorii od daného sboru. A tento postup opakuje pro všechny kategorie.

Výstup

Aplikace postupně zobrazuje v pravé části okna prezentované kategorie pro daný tým.

3.9.3 Odebrání prezentovaného týmu

V rámci prezence může dojít k chybě zadávajícího uživatele, který pro prezenci vybere nesprávný sbor nebo tým.

Vstup

Test je nutné provést až po scénáři 4.9.1., když je v aplikaci prezentován nejméně jeden tým. Uživatel v pravé části okna zaškrtne tým, který chce odstranit z prezence a klikne na tlačítko „Odebrat“.

Výstup

V pravé části okna je odstraněn vybraný tým uživatelem.

3.9.4 Uzavření skupiny kategorií

Uzavření skupiny kategorií je závislé na testovacím scénáři 4.9.1. nebo 4.9.2. Skupinu kategorií nelze uzavřít bez nejméně jednoho vyplněného týmu, patřícího do dané skupiny. Skupiny kategorií se řídí dle číselníku CB_Categories.csv popsaného v kapitole 4.6.2 Číselník kategorií.

Vstup

Uživatel na záložce „Prezence“ vybere v pravém horním ComboBoxu skupinu kategorií pro uzavření a klikne na tlačítko uzavřít. Před uzavřením je zobrazena hláška, kterou je nutné potvrdit pro uzavření.

Výstup

Na záložce prezence jsou překresleny všechny TreeView, kde nelze již pracovat s týmy řazenými do kategorie, která je součástí uzavřené skupiny kategorií. Na záložce „Stopky“ jsou pod jednotlivými tratěmi vygenerovány startovní listy z uzavřené skupiny kategorií.

3.9.5 Zkušební start

Obvyklým případem zkoušky všech připojených zařízení je před zahájením end-to-end test, který má za úkol vyzkoušet správné zapojení všech zařízení do časomíry, jako je startovací pistole, spínače terčů a vyzkoušení sériové komunikace s PC.

Vstup

V případě, že už jsou vytvořeny startovací listiny, je nutné, aby se uživatel přepnul do sekce „Nastavení“ a kliknul na tlačítko „Vytvoření týmu“. Tlačítko na první pozici všech startovacích listin vloží testovací tým, který nebude aplikací zpracováván do výsledků. Startér vystřelí z pistole, která je připojena k časomíře. Rozhodčí u terčů manuálně sepne spínač levého terče a pak spínač pravého terče.

Výstup

Po výstřelu z pistole je rozeběhnut čas pro danou trať. Při manuálním sepnutí levého terče je zastaven čas pro terč na časomíře i v aplikaci v počítači. Při manuálním sepnutí pravého terče je zastaven čas pravého terče. V aplikaci je zobrazen i konečný čas, který se rovná hodnotě pravého terče.

3.9.6 Dokončený pokus

Nejčastější scénář, který nastává při provádění pokusu hasičského útoku, je dokončený pokus, který není rozhodčím vyhodnocen jako neplatný, a je zapsán výsledek. Pro test je nutné dokončení testovacího scénáře 4.9.4., na který tento test navazuje. V aplikaci jsou vytvořeny startovní listiny.

Vstup

Startér vystřelí z pistole, která je připojena k časomíře. Rozhodčí u terčů manuálně sepne postupně spínače na terčích. Nejdříve na levém a pak na pravém terči

Výstup

Aplikací je zapsán do výsledků dosažený čas na levém a pravém terči pro tým, který byl na řadě ve startovním listu. Tým je ze startovního listu odstraněn.

3.9.7 Ulitý start

Scénář, který je možné za závodech v požárním sportu ojediněle vidět. Nejméně jeden ze členů týmu vyběhne ze startovní čáry dříve než na povel startéra, který následně dvakrát vystřelí z pistole, aby signalizoval ulitý start. Rozhodčím je start vyhodnocen jako neplatný a je opakován. Pro test je nutné dokončení testovacího scénáře 4.9.4, na který tento test navazuje. V aplikaci jsou vytvořeny startovní listiny.

Vstup

Startér dvakrát za sebou vystřelí z pistole.

Výstup

Na časomíře a v aplikaci se rozběhne čas. Není neprovedeno odstranění týmu ze startovní listiny a není zapsán čas do výsledků. Čas na časomíře i v aplikaci běží, dokud není resetován obsluhou aplikace.

3.9.8 Zapsání neplatného pokusu při nesepnutí terčů

Testovací scénář odpovídá situaci, která se velmi často na závodech vyskytuje. Při provádění pokusu nedojde k sepnutí spínačů z důvodu nedokončení pokusu. Takovýto pokus dle pravidel požárního útoku je vyhodnocen jako neplatný.

Vstup

Startér vystřelí z pistole, která je připojena k časomíře. Týmu, který provádí pokus, se v průběhu dvou minut nepodaří sepnout spínač na obou terčích a tím nezastaví čas běžící v časomíře. Pokus je rozhodčím vyhodnocen jako neplatný. Uživatel aplikace na záložce „Stopky“ klikne na tlačítko „N“ pro trať, na které byl pokus vyhodnocen jako neplatný.

Výstup

Po kliknutí na tlačítko je aplikací zapsan pro daný tým neplatný pokus, který je zobrazen v záložce „Výsledky“. V případě, že dojde k sepnutí spínače na jednom terči, čas je zapsán, ale pokus je vyhodnocen jako neplatný.

3.9.9 Zapsání neplatného pokusu po sepnutí terčů

Situace, kterou má simulovat tento scénář je následující. Týmu se podaří sepnou oba terče a tím zastavit čas na časomíře. Pokus je ale rozhodčím vyhodnocen jako neplatný z důvodu porušení pravidel, jako může být dotknutí nástřikové čáry proudařem či jiným členem družstva, zkřížení proudů proudařů, nesešroubování přívodního vedení do stroje po ukončení pokusu a další alternativy, které vycházejí z pravidel požárního útoku. Test nelze provést bez ukončení testu 4.9.4.

Vstup

Do výsledků je k týmu, který byl právě na řadě, zapsán dosažený čas, který je nutné manuálně uživatelem aplikace změnit. Uživatel v záložce „Výsledky“ vyhledá daný tým, dvakrát na něj klikne kurzorem a manuálně nastaví dosažené časy v daném pokusu jako neplatné.

Výstup

V seznamů výsledků je přepočítáno pořadí pro všechny dosažené výsledky.

4 Závěr

Cílem diplomové práce bylo vytvořit aplikaci pro komunikaci s časomírami TVR elektronik. Aplikace zefektivňuje organizaci závodů v požárním útoku vytvářením prezenčních listin, startovacích listin a zápis výsledků na webové stránky firesport.eu.

Dosavadní zkoušky potvrdily správnost návrhů jednotlivých částí i jejich spojení do funkčního celku. V současné době byla aplikace předána Sboru dobrovolných hasičů Sendražice a Sboru dobrovolných hasičů Kostelec nad Černými Lesy k vyzkoušení jednotlivých funkcionalit. První ostré použití aplikace je v plánu 1.8.2020 na závodech Polabské hasičské ligy pořádané sborem dobrovolných hasičů Sendražice.

Při testech aplikace bylo zjištěno, že by vytvoření již předdefinovaných prezenčních listin týmů pro soutěž mohlo vést k dalšímu zjednodušení organizace závodů, podobně jako jsou na záložce „Moc Data“ tlačítka pro prezenci týmů. Bylo by vhodné vytvořit konfigurační soubor, který by nastavoval prezenční listinu. Předdefinovaný soubor by se skládal vždy z týmů, které jsou registrovány do daného ročníku ligy. Uživatel aplikace by následně pouze odstranil týmy, které na soutěž nepřijely, a přidal týmy, které se chtějí soutěže účastnit, ale nejsou v lize registrovány.

Pro rozšíření aplikace by bylo vhodné i zvážit funkcionalitu manuální změny pořadí vygenerované startovní listiny. V rámci prezence může dojít k žádosti prezentujícího týmu, aby tým šel na start jako první nebo poslední z blíže nespecifikovaného důvodu. V aktuální chvíli by uživatel aplikace nedokázal vyjít vstříc prezentujícímu týmu.

Další možné úpravy aplikace mohou vyplynout z jejího užívání při pořádání závodů v požárním útoku.

5 Seznam použitých zdrojů

1. *Hasičský záchranný sbor České republiky: Disciplíny požárního sportu* [online]. [cit. 2020-03-22]. Dostupné z: https://www.hzscr.cz/clanek/discipliny-pozarniho-sportu.aspx?q=Y2hudW09MQ%3d%3d
2. *Hasičský záchranný sbor České republiky: Pravidla požárního sportu* [online]. [cit. 2020-03-22]. Dostupné z: https://www.hzscr.cz/clanek/pravidla-pozarniho-sportu.aspx
3. *TVR elektronik VÝROBA SPORTOVNÍCH ČASOMÍR* [online]. [cit. 2020-03-22]. Dostupné z: https://www.trv-kocab.cz/cs/casomiry/casomira-pozarni-sport
4. *Docs .NET: Průvodce technologií .NET* [online]. [cit. 2020-03-22]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/standard/
5. PECINOVSKÝ, Rudolf. *Návrhové vzory: [33 vzorových postupů pro objektové programování]*. Brno: Computer Press, 2007. ISBN 978-802-5115-824.
6. GAMMA, Erich. *Návrh programů pomocí vzorů: stavební kameny objektově orientovaných programů*. Praha: Grada, 2003. Moderní programování. ISBN 80-247-0302-5.
7. *C# in Depth: Implementing the Singleton Pattern in C#* [online]. [cit. 2020-03-22]. Dostupné z: https://csharpindepth.com/articles/singleton
8. *Průvodce technologií .NET: Návrhový vzor Pozorovatel* [online]. [cit. 2020-03-22]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/standard/events/observer-design-pattern
9. *Davelop PAPER: Design pattern singleton pattern* [online]. [cit. 2020-03-22]. Dostupné z: https://developpaper.com/design-pattern-singleton-pattern/
10. *C# Corner: Observer Design Pattern Using C#* [online]. [cit. 2020-03-22]. Dostupné z: https://www.c-sharpcorner.com/article/observer-design-pattern-using-c-sharp/
11. *ICPSR Find & Analyze Data: What is a Codebook?* [online]. [cit. 2020-03-22]. Dostupné z: https://www.icpsr.umich.edu/icpsrweb/content/shared/ICPSR/faqs/what-is-a-codebook.html
12. Použití vláken a dělení na vlákna. *Microsoft Docs* [online]. [cit. 2020-04-03]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/standard/threading/using-threads-and-threading
13. Použití vláken a dělení na vlákna. *Microsoft Docs* [online]. [cit. 2020-04-03]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/framework/winforms/controls/how-to-make-thread-safe-calls-to-windows-forms-controls

14. Evidence družstev. *Podlipanska liga, z.s.* [online]. [cit. 2020-04-03]. Dostupné z: http://www.podlipanskaliga.cz/node/191
15. MACDONALD, Matthew. *Pro .Net 2.0 Windows Forms and Custom Controls in C#*. 31. prosinec 2004. COMPUTER BOOKSHOPS. ISBN 1590594398.
16. SELLS, Chris a Michael WEINHARDT. *Windows Forms 2.0 programming*. Upper Saddle River, NJ: Addison-Wesley, c2006. ISBN 0321267966.
17. GRIFFITHS, Ian a Matthew ADAMS. *.NET Windows forms in a nutshell*. Cambridge: O'Reilly, c2003. ISBN 9780596003388.
18. HALL, Gary McLean. *Pro WPF and Silverlight MVVM: effective application development with Model-View-View Model*. New York, NY: Apress, [2010]. ISBN 1430231629.
19. ČÁPKA, David. Úvod do C# a .NET frameworku. *IT Network* [online]. [cit. 2020-04-03]. Dostupné z: https://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-uvod-do-jazyka-a-dot-net-framework
20. Úvod do C# a .NET frameworku. *Pragmateek* [online]. 2014-12-12 [cit. 2020-04-03]. Dostupné z: http://pragmateek.com/is-wpf-dead-the-present-and-future-of-wpf/
21. MACDONALD a MATTHEW. *Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5*. Germany: Springer-Verlag Berlin and Heidelberg GmbH & Co. ISBN 9781430243656.
22. ALBAHARI, Joseph. *C# 7.0 in a Nutshell*. O'Reilly Media, Inc, USA, 2017. ISBN 1491987650.
23. BORY, Pavel. *C# bez předchozích znalostí*. Brno: Computer Press, 2016. ISBN 978-80-251-4686-6.
24. LINQ to XML. *Microsoft Docs* [online]. [cit. 2020-04-03]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/bb387098(v=vs.140)?redirectedfrom=MSDN
25. OrderBy: Sorts a collection in ascending order. *LINQ Samples* [online]. [cit. 2020-04-03]. Dostupné z: https://linqsamples.com/linq-to-objects/ordering/OrderBy-numbers-lambda-csharp
26. Language Integrated Query (LINQ). *Microsoft Docs* [online]. [cit. 2020-04-03]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/
27. Sdružení hasičů Čech, Moravy a Slezska. *Sdružení hasičů Čech, Moravy a Slezska* [online]. [cit. 2020-04-03]. Dostupné z: https://www.dh.cz/

6 Přílohy

Zdrojový kód aplikace Timer

Zdrojový kód knihovny Interconnection

Projekt pro aplikaci Postman s vytvořenými voláními firesport.eu

Vyexportované diagramy z aplikace drawio