Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Biometrický docházkový systém

Jan Rosmarin

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Rosmarin

Informatika

Název práce

Biometrický docházkový systém

Název anglicky

Biometric attendance system

Cíle práce

Cílem práce je vytvoření a zavedení firemního docházkového systému na biometrické bázi. Vytvářený systém bude zahrnovat hardwarovou i softwarovou část.

Metodika

Bude provedena analýza současného stavu a sběr požadavků firmy na docházkový systém ze strany vedení i uživatelů s důrazem na zefektivnění využívání uložených záznamů pro denní kontrolu přístupů a zlepšení využívání docházkového systému jako podkladu pro výpočet mezd. Po proběhlé analýze dojde v rámci hardwarové části projektu k vytvoření funkčního prototypu zařízení pro verifikaci uživatelů a ukládání jejich přístupů. Prototyp bude nasazen paralelně se současným systém pro evidenci docházky. Během zkušebního nasazení dojde za dodržení agilní metodiky vývoje k vytvoření softwaru pro obsluhu celého systému. V následující části práce dojde k popsání kompletního technického řešení systému, finální implementace do provozu a komplexního zhodnocení nasazení s určitým časovým odstupem. Použité technologie budou pro hardwarovou část projektu vývojová platforma Arduino s dotykovým displejem a čtečkou otisku prstu, pro softwarovou část projektu C# a .NET Framework pro vytvoření programu a MS SQL pro uložení dat.

Doporučený rozsah práce

30-40 stran

Klíčová slova

Docházkový systém, biometrie, informační systém, podnik, analýza, implementace

Doporučené zdroje informací

Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. Wiley,2013. ISBN: 978-1-118-54936-0
Lacko L. Mistrovství v Microsoft SQL Server 2012. Praha: Computer press, 2013. ISBN 9788025137734.
MORKES, D. Microsoft SQL Server 2000 : tvorba, úprava a správa databází. Praha: Grada, 2004. ISBN 80-247-0732-2.
Sharp J. Microsoft Visual C# 2010. Praha:Computer press. ISBN 978-80-251-3147-3
TVRDÍKOVÁ, M. -- ČESKÁ SPOLEČNOST PRO SYSTÉMOVOU INTEGRACI. Zavádění a inovace informačních systémů ve firmách. Praha: Grada, 2000. ISBN 80-7169-703-6.

Předběžný termín obhajoby

2017/2018 LS – PEF

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 11.1.2018

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 11.1.2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 15.3.2018

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Biometrický docházkový systém" jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 03. 2018 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu bakalářské práce, kterým byl Ing. Marek Pícka, Ph.D. za odborné vedení, možnost konzultací a poskytování rad během tvorby této práce.

Biometrický docházkový systém

Souhrn

Tato bakalářská práce se zabývá problematikou docházkových systémů. Je v ní představeno chápání pojmu informační systém a uvedeny příklady vývoje informačního systému pro potřeby docházkového systému na biometrické bázi. Dále představí nutné technologie a postupy pro samotný vývoj docházkového informačního systému.

Hlavním cílem práce bude analýza, navržení a vytvoření reálného docházkového informačního systému pro zvolenou firmu. Hardwarové řešení systému bude postaveno na platformě Arduino. Návrh aplikace bude vytvořen s ohledem a postupy specifikace UML. Pro vývoj aplikace je zvoleno prostředí .NET, programovací jazyk C# a knihovna tříd pro tvorbu grafického rozhraní WPF. Jako databázi použijeme MS-SQL konkrétně jeho bezplatnou edici MS-SQL Express. Pro tvorbu uživatelského rozhraní využijeme běžné standardy. Výsledný produkt(hardwarová část) bude umístěn v rámci firmy a softwarová aplikace nahrána na firemní klientské PC.

Vytvořený biometrický docházkový informační systém poskytne firmě zjednodušení a zefektivnění stávajícího stavu kdy je používání stávajícího systému spojeno s různými omezeními.

Klíčová slova: Biometrický docházkový systém, Informační systém, Arduino, UML, .NET, C#, WPF, MS-SQL Express.

Biometric attendance system

Summary

This bachelor thesis deals with the issue of attendance systems. It introduces the understanding of the concept of information system and gives examples of development of the information system for the needs of the attendance system on a biometric basis. It will also introduce the necessary technologies and procedures for the development of the attendance information system itself.

The main aim of the thesis will be the analysis, design and creation of a real attendance information system for the selected company. The hardware solution of the system will be built on the Arduino platform. The application design will be created with the UML specification and procedures. The .NET application, the C # programming language, and the class library for the WPF graphical interface are used to develop the application. As a database, we will use MS-SQL specifically for its free MS-SQL Express edition. We use common standards to create a user interface. The resulting product (hardware part) will be placed within the company and the software application uploaded to the corporate client PC.

The created biometric attendance information system will provide the company with a simplification and streamlining of the current state where the use of the existing system is linked to various constraints.

Keywords: Biometric attendance system, Information system, Arduino, UML, .NET, C#, WPF, MS-SQL Express.

Obsah

Seznam obrázků

1 Úvod

V rámci této bakalářské práce bude vytvořen biometrický docházkový informační systém. Bude provedena analýza současného stavu zaznamenávání časů přístupu zaměstnanců ve sledované firmě, poté návrh nového řešení, tvorba a testování nově vytvořeného systému.

V teoretické části práce budou ukázány důležité teoretické technologické pojmy potřebné k tvorbě projektu, popsáno vývojové prostředí a následně budou definovány teoretické pojmy související s tvorbou informačních systémů. V rámci praktické části práce bude popsán návrh a vytvoření hardwaru a softwaru, implementace a testování nového systému. Hardware a obslužná aplikace bude nakonec nasazena přímo ve zmíněné firmě.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je vytvoření a zavedení firemního docházkového systému na biometrické bázi. Vytvářený systém bude zahrnovat hardwarovou i softwarovou část.

2.2 Metodika

Bude provedena analýza současného stavu a sběr požadavků firmy na docházkový systém ze strany vedení i uživatelů s důrazem na zefektivnění využívání uložených záznamů pro denní kontrolu přístupů a zlepšení využívání docházkového systému jako podkladu pro výpočet mezd. Po proběhlé analýze dojde v rámci hardwarové části projektu k vytvoření funkčního prototypu zařízení pro verifikaci uživatelů a ukládání jejich přístupů. Prototyp bude nasazen paralelně se současným systém pro evidenci docházky. Během zkušebního nasazení dojde za dodržení agilní metodiky vývoje k vytvoření softwaru pro obsluhu celého systému. V následující části práce dojde k popsání kompletního technického řešení systému, finální implementace do provozu a komplexního zhodnocení nasazení s určitým časovým odstupem. Použité technologie budou pro hardwarovou část projektu vývojová platforma Arduino s dotykovým displejem a čtečkou otisku prstu, pro softwarovou část projektu C# a .NET Framework pro vytvoření programu a MS SQL pro uložení dat.

3 Teoretická východiska

V této kapitole se práce bude zabývat vysvětlením všech základních pojmů nutných pro tvorbu tohoto projektu. V první kapitole bude přiblíženo současné pojetí biometrických technologií a v následujících se vysvětlí všechny nutné technologie, které jsou potřebné pro vývoj tohoto projektu. Tyto více praktické poznatky teoretických východisek budou uplatněny při vývoji hardwarové i softwarové části tohoto projektu. Více teoretická část této kapitoly definuje pojmy jako je informační systém, architektura informačního systému a jeho životní cyklus společně s využitím UML diagramů pro návrh systému.

3.1 Biometrické ověřování

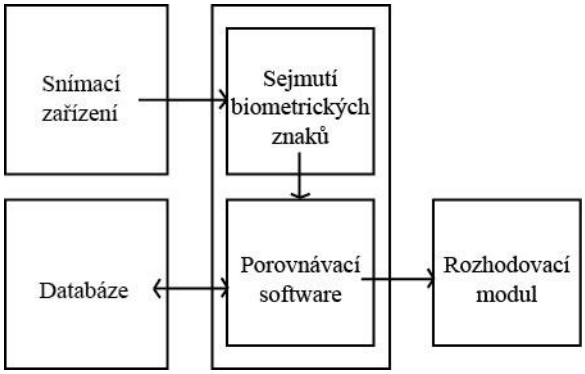
Biometrie je pojem využívaný všude tam, kde je potřeba zajistit korektní identifikaci osoby vyžadující přístup. Funguje na principu měření fyzikálních znaků, které jsou jedinečné pro každého člověka. Při biometrické identifikaci systém sejme biometrické znaky, prohledá svoji databázi a porovná získaný vzorek s uloženými šablonami. Pokud systém nalezne shodu, ukončí proces a kladně verifikuje uživatele.

Biometrický systém pro svoji správnou funkčnost potřebuje hardwarový i softwarový systém. Hardware (kterým může být například čtečka, kamera, mikrofon, optické zařízení) digitalizuje potřebnou biometrickou charakteristiku a software tento digitalizovaný vzorek porovnává se svými záznamy.

Biometrické systémy došly značného rozšíření zejména z důvodu rychlosti ověření, kdy uživateli proces ověřování nepůsobí časový i jiný diskomfort a pro správce systému použití biometrie znamená z důvodu náročnějšího oklamání systému vetší bezpečnost. (Rak, 2003). Samotný proces identifikace osob, který se v dnešní době nejčastěji používá lze rozdělit do 3 oblastí:

- *Identifikace pomocí hesla* – uživatel dostává přístup do systému na základně znalosti hesla. Tento postup je méně náročný na realizaci, ale nepatří mezi nejbezpečnější z důvodu snadného převodu hesla na neoprávněnou osobu.

- *Identifikace s pomocí předmětu* – uživatel dostává od správce systému identifikační předmět (například RFID čip), který potřebuje vždy, kdy potřebuje ověření systémem. Tento způsob ověření patří mezi více bezpečné, ale stále je možné, aby se k ověřovacímu předmětu dostala neoprávněná osoba a získala tak přístup k systému.
- *Identifikace pomocí biometrie* – Výše uvedené nedokonalosti odstraňuje biometrický přístup, kdy je pro případného útočníka průnik do systému (množstvím vynaložených prostředků a potřebného času) výrazně náročnější.



Obrázek 1 : Způsob ověření biometrickým systémem

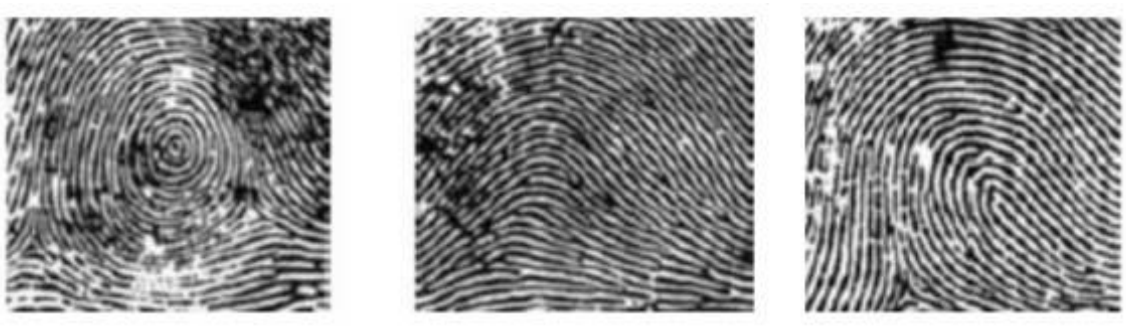
3.1.1 Rozdělení identifikačních biometrických systémů

Protože má každý člověk velké množství jedinečných fyziologických a behaviorálních znaků existuje proto i velké množství druhů systémů, které tyto charakteristiky mohou snímat.

Mezi nejčastěji používané biometrické identifikační metody patří:

- *Snímání otisku prstů* – jedna z nejstarších biometrických metod historicky využívaná v kriminalistice díky rozvoji daktyloskopie což je nauka o obrazech papilárních linií. Z ní vyplývá, že neexistují dvě osoby, které by měli kompletně shodné papilární linie. Biometrické snímače v této oblasti většinou pracují na optoelektronické, teplotní,

elektroluminiscenční, ultrazvukové nebo tlakové bázi. Nevýhody snímání prstu se mohou vyskytnout v situacích, kdy u snímaných osob dochází k častějšímu i drobnému poškození nebo poranění papilárních linií kdy systém snímaný vzor nedokáže identifikovat. Existuje také možnost akceptování snímaného vzoru při použití silikonu nebo potravinářské želatiny pro vytvoření falešného otisku prstu.



Obrázek 2: Hlavní sledované vzory papilárních linií

- *Snímání geometrie ruky* – méně přesná než je metoda snímání otisku prstu je identifikace pomocí geometrie ruky. Principem snímání je zjištění míry délky a šířky jednotlivých prstů, kostí a kloubů. Neměří se znaky, které se mohou v čase měnit jako jsou nehty nebo charakteristiky kůže.
- *Snímání oční duhovky* – Mezi biometrickými charakteristikami patří metody s nejvíce rozlišovacími možnostmi. Díky komplexnosti oční duhovky je pravděpodobnější nalézt identické otisky prstů než nalezení totožných duhovek (Čandík, 2004). V lidské duhovce lze sledovat více viditelných znaků, které se hodí pro biometrické snímání (různé rýhy a skvrny nebo kruhy). Při procesu snímání se sejme obvykle půlkruh ve spodní části duhovky, který je dále transformován na úzký proužek s pravými úhly. Snímek, který je snímán je obvykle černobílý ve vysokém rozlišení a bývá přisvícen infračerveným světlem. Pro uživatele metody je výhoda žádného fyzického kontaktu s kamerou, která duhovku snímá. Proces je zároveň velice rychlý.
- *Snímání oční sítnice* – Podobný proces snímání jako snímání duhovky, velké množství identifikačních znaků sítnice zajišťuje vysokou přesnost snímání.

- *Snímání geometrie obličeje* – Metoda využívá kamery s vysokým rozlišením a hledá hlavně ty znaky, které neovlivňuje mimika obličeje. Měří se očnice, oblast okolo lícních kostí nebo oblasti kolem úst. V zemích, kde existuje velká penetrace kamer na veřejnosti (Velká Británie, Čína) dokáží bezpečností složky díky této metodě najít požadované osoby s velkou mírou úspěšnosti.
-

3.2 Platforma Arduino

V dnešním světe je na trhu obrovské množství různých desek a čipů různé kvality s velmi různorodou mírou uživatelské podpory a přívětivosti zcela závisející na libovůli výrobce a vytvořené komunity. Jednou z vedoucích vývojových platforem se v posledním desetiletí stala platforma Arduino. Je to vývojová platforma na bázi open-source sloužící jak pro výuku tak i pro rychlé prototypování elektroniky a software(Voda,2015). Arduino nabízí velké množství desek od těch nejmenších a méně výkonných po komplexní desky, které obsahují téměř všechna možná rozhraní jako USB, SD karty, WIFI, Ethernet s podporou velkého množství elektronických součástek. Open-source povaha projektu zajistila, že komunita ochotně vytváří nové návody a tutoriály stejně jako uživatelskou podporu díky velkému množství internetových fór.

Oficiální desky Arduino obsahují procesor od firmy Atmel a další elektronické komponenty podle typu dané desky. Většina desek Arduina obsahuje USB převodník sloužící pro komunikaci s PC (Voda, 2015). Jednou z nejčastěji používaných desek je Arduino UNO.

Obrázek 3 : Deska Arduino UNO

3.2.1 Arduino UNO

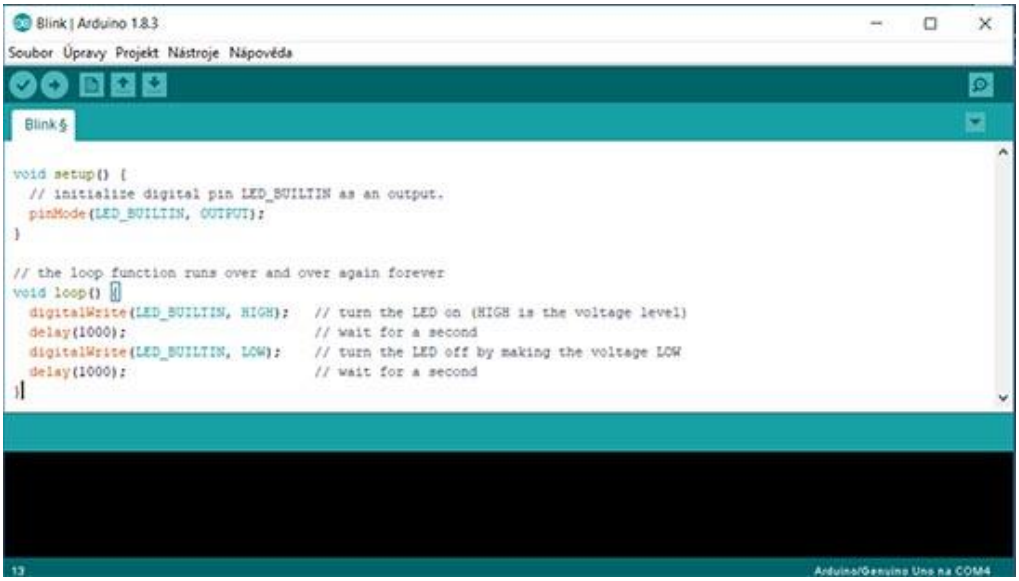
Arduino Uno je v současné době asi nejčastěji používaný typ desky. Je přímým pokračovatelem hlavní vývojové linie, která započala prvním Arduinem se sériovým portem místo USB, pokračujícím přes Arduino Extreme, NG, Diecimila a Duemilanove až k dnešnímu Uno. Na desce najdeme procesor ATmega328 a již klasické USB. Z této hlavní linie se vyvinuly i další dvě speciální desky. První z nich je Arduino Ethernet, která má stejnou výbavu jako Uno. Místo USB portu zde ale najdeme Ethernet port pro připojení k síti. Příjemná je přítomnost slotu pro microSD karty. Druhou deskou je Arduino Bluetooth. Jak už název napovídá, místo USB zde najdeme bluetooth modul pro bezdrátovou komunikaci. Velmi odlehčenou verzi Arduina Uno je Arduino Pro. To postrádá USB port a je tedy nutné ho programovat externím převodníkem. Je určeno spíše k pevnému zabudování do nějakého projektu (Voda, 2015).

3.2.2 Arduino shieldy

Žádná Arduino deska nemůže v základu obsahovat všechny komponenty, které při vývoji projektu bude nutné použít. Proto existují přídavné desky, které lze se základní deskou Arduina jednoduše propojit bez nutnosti pájení, a které významným způsobem rozšiřují funkcionalitu. Mezi důležité shieldy patří desky, které přidávají podporu LAN, SD karet a displejů.

3.2.3 Vývojové prostředí (IDE) Arduina a programovací jazyk

Arduino IDE(aktuální verze 1.8.3) je vývojové prostředí napsané v jazyce Java. Programování je možné v jazyce C/C++ s knihovnou Wiring. Spuštěný program se inicializuje v bloku setup() a pokračuje ve smyčce v bloku loop() až do odpojení napájení.



Obrázek 4 : Vývojové prostředí IDE a ukázka jednoduchého programu

3.3 .NET Framework

Framework .NET je aplikační prostředí vytvořené společností Microsoft pro snadný vývoj aplikací. Vývojáři mohou psát kód v mnoha jazycích jako je C#, C++, Visual Basic, F# a další. Naprogramovaný kód pak může být spuštěn všude tam kde je platforma .NET nainstalována. Vzhledem ke svému původu je .NET framework nejvíce používán na platformě Windows, ale díky snaze společnosti Microsoft existují implementace .NET jako je Mono nebo .NET Core, které umožňují kompatibilitu v operačních systémech Linux a Mac OS X.

.NET Framework funguje doslova jako substrát, na kterém lze pěstovat software. Jeho jádro je založené na principech objektově orientovaného programování a všechny základní služby zpřístupňuje široké škále programovacím jazykům. .NET Framework

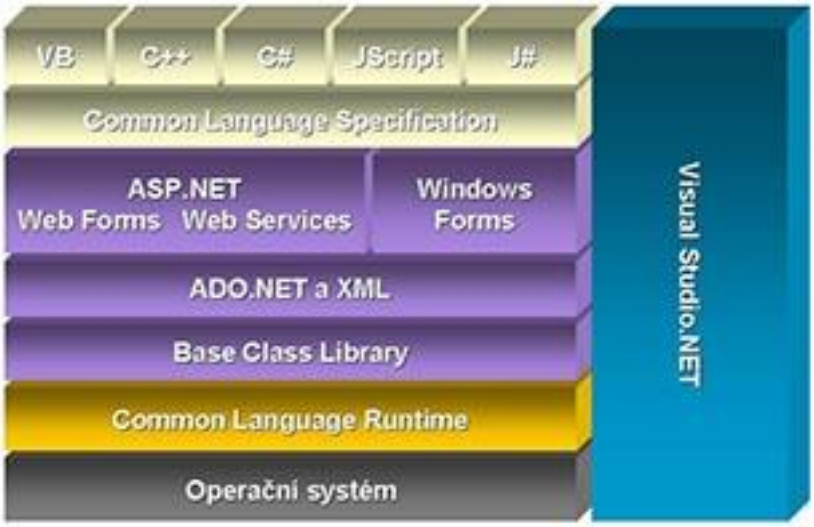
automaticky podporuje třídy, metody, vlastnosti, konstruktory, události, polymorfismus atd. Ve výsledném efektu to znamená, že není podstatné, ve kterém programovacím jazyce komponenty vytváříme případně, jaké komponenty používáme (Běhálek, 2008).

3.3.1 Architektura platformy .NET

Nejnižší úroveň platformy .NET, která zajišťuje překlad do intermediárního kódu MSIL náleží vrstvě CLR – Common Language Runtime. Tuto vrstvu lze zpodobnit s pojmem virtuálního stroje používaném v programovacím jazyce Java. Princip virtuálního stroje je kompilování zdrojového kódu nikoliv přímo do nativního kódu stroje, ale do výše zmíněného intermediárního jazyka (Běhálek, 2008).

Nad prostředím CLR jsou základní knihovny tříd a nad nimi pro přístup k datům a pro práci se soubory xml. Nejvyšší vrstvou jsou potom sady knihoven usnadňující práci s grafickým rozhraním. Vše zastřešují podporované programovací jazyky jejichž vlastnosti jsou definovány pomocí obecného rozhraní CLS.

K důležitému prvku CLR patří podpora společného typového systému CTS. Tento systém nezávisí na jazykové implementaci a dělí se na dvě hlavní kategorie: Typy hodnotové a typy referenční. Základ pak pro každý typ představuje třída System.Object.



Obrázek 5 : Architektura platformy .NET

3.4 Jazyk C#

Programovací jazyk C# je vysokoúrovňový objektově orientovaný jazyk, který vyvinula firma Microsoft a k jehož představení došlo společně s celým vývojovým prostředím .NET Framework. Tento jazyk vychází v mnoha ohledech z programovacího jazyka C/C++, v jiných ohledech je ovšem bližší k programovacímu jazyku Java. Překladač jazyka rozlišuje velká a malá písmena (je case sensitive).

Mezi hlavní vlastnosti jazyka patří následující charakteristiky:

- Jazyk C# je orientovaný čistě objektově
- Obsahuje jednoduchou dědičnost a je možná násobná implementace rozhraní
- K členským datům a metodám přidává vlastnosti a události
- Zajišťuje automatickou správu paměti. Aplikace dokáže korektně uvolňovat zdroje o jejichž uvolnění se stará garbage collector.
- Nativně podporuje komponentové programování
- Dokáže zpracovávat chyby pomocí výjimek

Většina uvedených vlastností vychází přímo z funkcionality vývojového rámce .NET. Jazyk C# je také integrován do vývojového prostředí Visual Studio.NET (Běhálek, 2008)

3.5 Windows Presentation Foundation(WPF)

Framework Windows Presentation Foundation (WPF) je knihovna tříd pro tvorbu grafického rozhraní a formulářových aplikací a je součástí .NET frameworku počínaje verzí 3.0. (Yosifovich, 2012). WPF je považován za nástupce knihovny tříd Windows Forms.

Díky využití velkého množství funkčních ovládacích prvků umožňuje flexibilnější a rychlejší vývoj aplikace. Principem, na kterém framework staví, je vhodnější způsob, který odděluje uživatelské rozhraní od logiky aplikace. Rozhraní zajišťuje pro ty potřeby nově vytvořený jazyk XAML. Vnitřní aplikační logika je programována díky standardním

programovacím jazykům platformy .NET jako je C#, Visual Basic, C++ a další. Ovládací prvky je možné využívat přímým programováním v aplikaci nebo pomocí takzvaného Data Bindingu, kdy dochází k navázání(binding) množiny dat přímo na ovládací prvky. Framework WPF sjednocuje uživatelské rozhraní, 2D a 3D grafiku, vektorovou a rastrovou grafiku, animaci a dokáže provázat data s audiem a videem.

3.5.1 Výhody WPF proti technologii WinForms

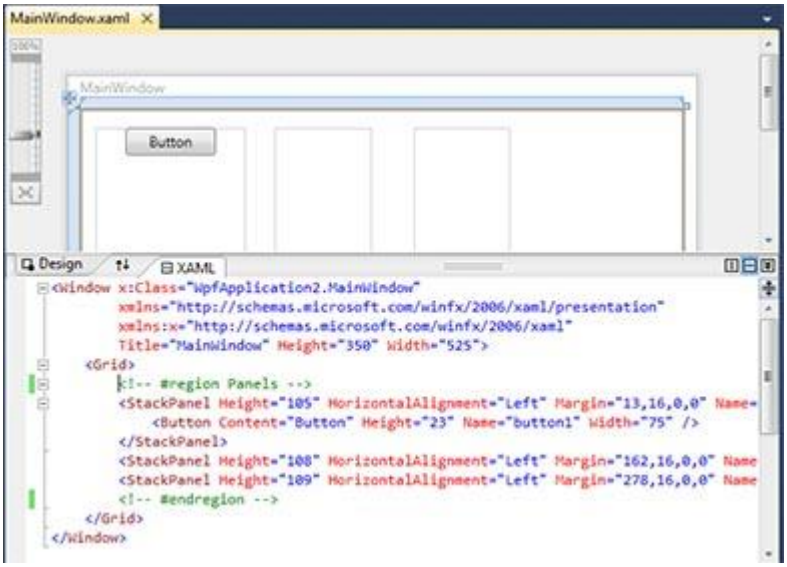
Jak již bylo uvedeno technologie WPF se považuje za nástupce technologie historicky starší technologie WinForms, která se z různých důvodů dodnes stále používá, ale použití WPF zaručuje vývoj s novějšími a vyspělejšími technologiemi.

- Oproti WinForm má programátor při tvorbě rozhraní ve WPF větší kontrolu nad výslednou aplikací. Přístup WinForm spočívající ve velkém množství ovládacích prvků „naházených“ do prostředí způsoboval velkou nepřehlednost při vývoji.
- Důsledně odděluje aplikační logiku od uživatelského rozhraní
- Dokáže využít akcelerovanou grafiku a aplikace má díky tomu vyšší výkon
- Zavedením nových jednotkových délek se WPF aplikace dokáže přizpůsobit různým prostředím (adaptuje se na různě veliké monitory a obrazovky).

3.5.2 Jazyk XAML

Extensible Application Markup Language (XAML) je značkovací jazyk (obdoba XML) sloužící pro návrh a popis grafického uživatelského rozhraní v aplikacích společnosti Microsoft na platformě WPF. Protože vychází z jazyka XML, musí podobně jako on obsahovat právě jeden kořenový element a elementy objektů mohou obsahovat dětské

elementy trojího typu: hodnota vlastnosti obsahu, položka kolekce nebo hodnota, kterou lze přetypovat na element objektu (Yosifovich, 2012). Dětské vnořené elementy dále mohou, ale nemusí obsahovat další vnořené elementy. Dělení samotných elementů probíhá na párové a nepárové. Každý element pak obsahuje atributy nebo vlastnosti jako Background, FontWeight, Text atd. nebo jim lze přiřazovat události(Events) jako je například Click.



Obrázek 6 : Ukázka použití jazyka XAML

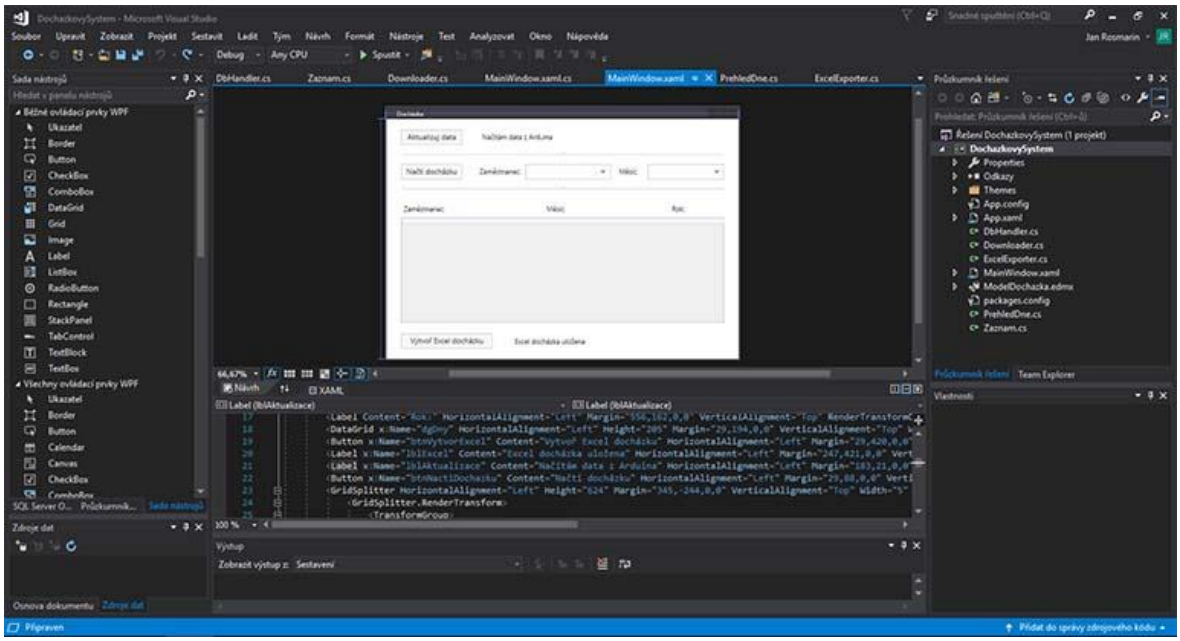
3.6 Entity framework

Je technologie společnosti Microsoft umožňující takzvané objektově relační mapování (ORM). Entity Framework dokáže mapovat databázové tabulky přímo na C# třídy a díky tomu je možné v kódu pracovat pouze s objekty a nestarat se o logiku dat(Entity framework dokáže sám na pozadí generovat SQL dotazy). Díky tomu programátor nemusí vůbec přijít do styku s jazykem SQL a výsledná aplikace tak zůstane 100% objektová.

Obrázek 7 : Princip fungování Entity frameworku

3.7 Microsoft Visual Studio 17

Visual Studio je vývojové prostředí (IDE) vyvinuté firmou Microsoft. Je možné ho využít pro vývoj různých aplikací jako jsou konzolové aplikace. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s webovými stránkami, webovými aplikacemi a webovými službami. (Běhálek, 2008).



Obrázek 8 : Prostředí Visual Studia IDE

Jako většina vývojových prostředí obsahuje i Visual Studio editor kódu, který podporuje IntelliSense a refaktoring. Nástroj Intellisence dokáže dokončit navrhnout a dokončit kód při psaní proměnných, funkcí metod i cyklů a dotazů. Do prostředí integrovaný debugger dokáže pracovat na úrovni kódu i na úrovni stroje. Díky refaktoringu dokáže vývojář provádět změny v kódu tak, aby nedošlo k omylům při jeho změnách, nástroj umožňuje zlepšovat kvalitu, čitelnost i čistotu psaného kódu.

Mezi další užitečné nástroje patří designer formulářů pro tvorbu aplikací s GUI, designerů podporuje Visual Studio několik (například WPF designer, WinForms designer, Designer mapování entit, designer tříd a databázových schémat).

Díky správci balíčků NuGet což je správce rozšíření, je možné jednoduše instalovat různé knihovny a rozšíření, díky čemuž se dá vylepšovat funkčnost Visual Studia na mnoha úrovních. Jazyky podporované Visual studiem jsou díky jazykovým službám. Ty umožňují, aby editor kódu a debugger mohl podporovat téměř jakýkoliv programovací jazyk. Mezi vestavěné jazyky patří C/C++ (použitím Visual C++), VB.NET (použitím Visual Basic .NET) a C# (použitím Visual C#).

3.8 Databázový system

Pro každý projekt, ve kterém je nutné pracovat s větším množstvím dat je potřeba zajistit vhodný databázový systém. „Velmi jednoduše řečeno, databáze je kolekce strukturovaných informací. Databáze jsou navrženy specificky ke správě velkého počtu informací a uchovávají data organizovaným a strukturovaným způsobem, který umožňuje uživatelům navrhnout a získat tyto data, když je potřebují“ (Agarwal, Huddleston a Raghuram, 2008). Databázových modelů existuje velké množství, mezi ty nejvíce využívané patří objektové databáze, hierarchické databáze a zejména databáze relační.

Ty fungují na relačním principu kde je každý záznam považován za řádek a každý sloupec za atribut. Kolekce záznamů se nachází v tabulkách a ty jsou mezi sebou propojeny díky relacím. Záznamy v databázích musí splňovat více požadavků, mezi hlavní patří požadavek na primární klíč. Díky němu je každý záznam v dané tabulce jedinečný. V tomto projektu využívaný databázový systém Microsoft SQL Server Express obsahuje dostatek

vhodných funkcí jako jsou pohledy, uložené procedury, triggery a integritní omezení. Pro práci s daty využívá jazyk SQL.

3.9 Informační systém

Definic informačního systému existuje větší množství, nejčastěji je Informační systém definován jako množina lidí, dat a postupů , která působí společně pro získání užitečných informací. (Basl, Bažíček, 2012).

- Systémem rozumíme množinu prvků systémů a jejich vazeb.
- Informacemi rozumíme data, která uživateli systémů poskytují nějakou znalost na jejichž základě můžeme plánovat, rozhodovat a řídit.

Informační systém můžeme také definovat podle zákona [Zákon č. 365/2000 Sb.]: „Informačním systémem funkční celek nebo jeho část zabezpečující cílevědomou a systematickou informační činnost. Každý informační systém zahrnuje data, která jsou uspořádána tak, aby bylo možné jejich zpracování a zpřístupnění, provozní údaje a dále nástroje umožňující výkon informačních činností.“

3.10 Architektura informačních systémů

Pomocí architektury informačních systémů definujeme rámec nutných řešení a definuje nám určitý směr vývoje informačního systémů. Musí být přehledná, všem účastníkům procesu srozumitelná a v rámci možností jednoduchá. Pro budování informačního systému je architektura IS podobně důležitá jako jsou výkresy pro technické nebo stavební obory.
Na základě vytvořené globální architektury navrhujeme dílčí architektury a jejich propojení. Tyto dílčí architektury dělíme na architektury

1. Funkční – Navržení funkcí IS hierarchicky. Nejnižší úroveň funkční hierarchie je viditelná uživatelům
2. Procesní – Navržení budoucího stavu podnikových procesů.

3. Datová – Navržení datové základny IS.
4. Softwarová – Určuje z jakých softwarových částí bude IS postaven a jaké mezi nimi budou vazby
5. Hardwarová – Určuje počty, typy a vazby komponent hardwaru
6. Technologická – Technologické řešení aplikace, propojuje softwarovou a hardwarovou část, definuje způsob zpracování aplikace IS a uživatelského rozhraní.
7. Procentuální – určuje nám kolik procent je nutných pro zpracování dané architektury.
8. Neurčitá – Pro neurčitost dané architektury slouží pro rozšíření dané oblasti

Při navržení kvalitní systémové architektury (minimalizace duplicit, kompatibilita systému a jeho integrita) minimalizujeme ze systémového hlediska náklady na případnou rekonstrukci systému (Basl, Bažíček, 2012).

3.11 Životní cyklus vývoje informačního systému

Životní cyklus je charakterizován jako časový úsek, začínající potřebou vytvořit informační systém a končí jeho nasazením. Dělí se na několik navazujících fází.

3.11.1 Specifikace cílů

V této fázi dochází ke shromáždění požadavků na systém, stanovení alespoň přibližné doby realizace a velikost předpokládaných nákladů a jednotlivých zdrojů (zdroje lidské, hardware, software atd.). Cíle se následně stanoví podle celkové analýzy aktuálního systému, známých problémů a požadavku koncových uživatelů (Tvrdíková, 2000). Po specifikaci cílů dojde k vytvoření další analýzy.

3.11.2 Systémová analýza

Tato důležitá vývojová fáze se charakterizuje analýzou odhalených poznatků z první fáze. Fáze je důležitá nutností odhalit nedokonalosti a chyby struktury systému a dat, protože při neodhalení těchto chyb může být v pozdější fázi vývoje komplikované tyto chyby odstranit.

3.11.3 Navržení systému

Navazuje na systémovou analýzu. Většinou jde o dokument obsahující časovou posloupnost vývoje, cenu, informace o implementaci a dodatečné informace jako je záruční/pozáruční servis a podmínky předání systému.

3.11.4 Implementace systému

Představuje naprogramování systému. Podklady pro vývojáře vycházejí z výstupu všech předcházejících fází. Na naprogramování jednotlivých funkcí naváže jejich propojení a ověření funkčnosti. Následně se připraví datové podklady pro testování (která by měla vycházet z dat reálných.

3.11.5 Systémové testy

Testy jsou důležitou součástí životního cyklu a jejich provádění na hotovém systému nám ukazuje chyby, které musí být po zjištění opraveny. Testovat bychom neměli v produkčním prostředí, ale pouze ve speciálně vytvořeném pro účely testování.
Charakterizujeme ho instalací, zaškolením jednotlivých uživatelů a poskytnutím potřebných manuálů. Zavedení je provedeno na základě vypracovaného harmonogramu a to tak, aby mohlo proběhnout v co možná nejkratší době.

3.11.6 Zavedení systému

Jednotlivých strategií pro zavádění systémů rozlišujeme více:

- Strategie pilotní – Systém zavádíme jen pouze pro jednu organizační jednotku naráz. Jako referenční jednotka je vybrána jednotka, na které je možné ověřit problémové oblasti. Pro naše potřeby není tato strategie nutná
- Strategie postupná – Využíváme ji většinou u rozsáhlejších informačních systémů. Zavedeny jsou na začátek všechny důležité části, na kterých závisí systém a následně jsou zavedeny části méně důležité. Tato strategie bývá poměrně časově náročná.
- Strategie nárazová – V jeden moment je ukončen provoz stávajícího systému a okamžitě je zaveden systém nový. Strategie je náročná na správnou přípravu a může být riziková pokud jsou v novém systému neodhalené chyby.
- Strategie souběžná – Nový systém je zaveden a paralelně s ním běží systém současný. Pokud systém pracuje spolehlivě a uživatelé ho používají bez problémů dojde postupně k odstavení původního systému. Tato strategie dokáže být náročnější na zaměstnance, kteří musí v průběhu zavádění pracovat s oběma systémy zároveň. Tato strategie je vhodná pro potřeby našeho projektu.

3.11.7 Provoz a údržba systému

Poslední fáze vývojového životního cyklu, ve které byl systém nasazen a je aktivně používán. Důležitou součástí je údržba systému, která zajišťuje informační systém ve správném chodu a implementuje případné změny, které byly vyvolány novými požadavky uživatelů.

3.12 Modely životního cyklu

Model životního cyklu software popisuje vzájemné vztahy mezi fázemi životního cyklu softwaru. Každý model obsahuje vlastní metodiku jak zajistit dostatečnou kvalitu produktu. V tomto kontextu často bývá pojem model a metodika zaměňován.

3.12.1 Vodopádový model

Ve vodopádovém modelu jsou jednotlivé fáze životního cyklu prováděny postupně a navazují na sebe. Tento model je vhodné zapojit v případech, kdy jsou všechny požadavky definovány ve fázi specifikace požadavků. Hlavním problémem je pozdní integrace. Pokud se při integraci projeví problémy, mohou vyžadovat změnu návrhu nebo přeprogramování a dochází ke zpoždění.

3.12.2 Prototypový model

Prototypový model předpokládá změnu požadavků zákazníka v průběhu životního cyklu a umožňuje reagovat na tyto změny. Hlavním cílem je seznámení zákazníka s první verzí systému v co nejkratším čase prostřednictvím prototypu. Prototyp je zjednodušená implementace systému, která je upravována na základě připomínek zákazníka.

Výhodou oproti vodopádovému modelu je výše zmíněná možnost reagovat na změny. Nevýhodou je náročnost metody u rozsáhlých systémů.

3.13 UML a modelování aplikací

Rozšířením objektově orientovaného programování se hledal způsob, jakým způsobem a z jakých pohledů se dívat na informační systémy. Začalo vznikat velké množství specifikací a standardů jakým způsobem objekty a jejich další aplikační zakreslovat V průběhu 90. let se díky firmě Rational Software podařilo za sjednocení různých druhů metodik vytvořit standard UML, který se úspěšně používá dodnes (Arlow, Neustadt, 2012). V následujích bodech budou předvedeny různé pohledy na UML diagramy.

3.13.1 UML jako náčrt(sketch)

UML diagramy je možné používat ve velmi jednoduché podobě jako náčrt. Obvykle jde o diagramy kreslené ručně například na tabuli nebo papír. Takto se používají například na jednání s klientem, kde lze podstatu problém lépe pochopit, když je vyjádřena graficky, protože je lidem grafická podoba bližší než text. Ovšem načrtávat lze stejně tak i při průběžném navrhování systému, kdy je potřeba diskutovat uvnitř vlastního týmu.

Použití diagramů je velmi důležité kvůli důležité vlastnosti, kterou je abstrakce. Každý diagram ukazuje vlastně jiný úhel pohledu na určitý problém. V dané chvíli prezentace není nutné zobrazit zbytek systému, a proto se zobrazí pouze to co je důležité v danou chvíli. Díky tomu UML diagramy pomáhají zlepšovat komunikaci a omezovat riziko špatného porozumění námi nebo klientem, díky čemuž by mohlo dojít ke špatnému návrhu systému.

3.13.2 UML jako plán(blueprint)

Lépe čitelnější než náčrt je UML diagram jako plán, který je více detailnější. Diagramy se vytváří v CAD nástrojích a slouží jako plán implementace, který využijí programátoři. Usnadňuje komunikaci v rámci týmu a ulehčuje implementaci systému, protože se pomocí těchto diagramů dokáží programátoři v systému lépe orientovat. Zároveň až dojde k dokončení systému, budou diagramy dále sloužit jako systémová dokumentace. Díky tomu, že jsou UML diagramy standardem i nezasvěcení programátoři jsou schopni se v systému dobře orientovat.

3.13.3 UML jako programovací jazyk

Posledním význam má UML jako programovací jazyk. Z detailního UML diagramu je možné generovat šablonu kódu sloužící jako základ pro implementaci. Je běžné, že databáze tyto modely používají na generování základních skriptů. UML diagramy dokáží

sloužit jako předpokládané rozšíření k programovacím jazykům orientovaných objektově. Nejrozšířenější variantu pro naše využití pak představuje vícenásobné UML.

3.13.4 Dělení UML diagramů

UML diagramy se v použití dále dělí na tyto základní skupiny:

- Diagramy struktury (Structure Diagrams) – Je popisována struktura systému, tedy z čeho se systém skládá.
- Diagramy chování (Behaviour Diagram) – Je popisováno chování systému, tedy jak se systém chová. V diagramech chování se ještě nachází samostatná skupina diagramy interakce (interaction diagrams), která popisuje interakce, které mezi sebou mají jednotlivé části systému.

Vytváření UML diagramů musí být účelové. Diagramy by se měly vytvářen, protože jsou potřeba mají pro svého tvůrce určitou přidanou hodnotu. Pokud je tvorba diagramů vynucená pouze tím, aby existovaly, nemá obvykle taková tvorba velký smysl. (Arlow, Neustadt, 2012)

3.14 Použité externí knihovny v projektu

V této části budou uvedeny externí knihovny, které budou využity pří vývoji softwaru pro platformu Arduino i pro klientskou aplikaci vytvořenou ve WPF.

3.14.1 Externí knihovna pro Arduino – TinyWebServer

Knihovna je určena pro vytvoření minimalistického „webserveru“ na platformě Arduino. Díky této knihovně bude moci klientský software přistupovat k uloženému csv souboru na SD kartě v Arduinu.

3.14.2 Externí knihovna pro WPF aplikaci – EPPlus

Tato knihovna slouží pro čtení i tvorbu Excel souborů s použitím Open Office XML formátu(xlsx). Tato knihovna je licencována pod GNU LGPL licencí a je tedy možné ji využít zdarma i pro komerční účely. Instalace do projektu probíhá přes Nuget správce balíčků.

4 Vlastní práce

V této kapitole je popisová postup vývoje hardwarové a softwarové části projektu podle výše uvedeného životního cyklu. Práce začíná rozvržením činností, které je třeba v rámci vývoje provést, navazuje analýza současného stavu docházkového informačního systému ve sledované firmě a seznam požadavků funkčnosti v novém systému. Potom je navržena systémová architektura, základní systémové funkce a vývoj vzhledu uživatelského rozhraní. Závěrem bude popsána vlastní implementace systému, testování a nasazení.

4.1 Projektový plán

V rámci plánu dojde k popisu nejdůležitějších činností, které bude třeba provést při vývoji informačního systému.

4.1.1 Činnosti

- Sběr a analýza požadavků uživatelů docházkového systému
- Návrh systému
  - Návrh architektury
  - Návrh UI hardwarové a softwarové části
- Vývoj
  - Vývoj UI pro platformu Arduino
  - Vývoj obsluhujícího softwaru platformy Arduino
  - Vývoj UI a softwaru klientské části
- Testy
  - Testovací scénáře
- Tvorba dokumentace
- Instalace hardwaru a instalace obsluhujícího klientského softwaru

4.2 Analýza

Firmu je možné charakterizovat jako firmu malou(do deseti zaměstnanců) a působí na trhu výroby a prodeje sportovních potřeb. Firma provozuje svoji činnost v objektu rozděleném na dvě části. Vedení a administrativní pracovníci mají kanceláře v patře, zaměstnanci působící ve výrobě pracují v provozu v přízemí budovy kde dochází k zaznamenávání přístupu.

4.2.1 Současný stav sledování přístupů

Dle požadavků firmy se docházka zaměstnanců sleduje pouze v provozu firmy, tedy v přízemí budovy. Historicky byl využíván písemný systém evidence docházky, kde si zaměstnanci zapisovali příchody, odchody a přestávky sami. Výhodou takového systému byla flexibilita a rychlost záznamu, nevýhodou naopak nutná poctivost zaznamenávajícího a obtížná digitalizace dat pro vytváření přehledů. Ze získaných zkušeností s takto vedeným systémem docházky vyplynulo, že takové řešení v případě nepoctivosti zaměstnanců není vhodné a proto se hledalo řešení spolehlivější.

Z důvodu omezeného rozpočtu byl firmou pořízen jednoúčelový docházkový systém vyrobený v Číně, který umožňoval evidovat docházku elektronicky a přístup byl na bázi biometrie – otisku prstu. U tohoto systému patřilo k výhodám nemožnost falšování záznamů a rychlost verifikace. K nevýhodám firma přiřadila pouze anglický jazyk obslužného zařízení, neintuitivnost hardware, pouze anglický jazyk v obslužném software jehož UI nebylo řešeno dobrým způsobem a i při snadné digitalizaci dat nebyli uživatelé spokojeni s výstupy z programu, které bylo pro mzdové a docházkové potřeby nutno složitě upravovat. Toto řešení bylo v provozu okolo 3 let, po kterých došlo k vznesení požadavků na nový systém. Stávající systém má nevýhody v nemožnosti úpravy obslužného software, obslužný hardware je pouze v anglickém jazyce a jeho ovládání při různých nestandardních situacích není předvídatelné.

4.2.2 Současné technologické vybavení.

V rámci objektu firmy jsou rozvedeny Ethernet rozvody a u současného systému i předpokládaného umístění hardwarové části nového projektu je síťová i LAN zásuvka. Pro klientskou část bude využito 1 standardní PC na MS Windows, na kterém běží stávající docházkový obslužný software. Technickou podporu a případnou úpravu systému zajišťuje odpovědný zaměstnanec firmy.

4.2.3 Funkční požadavky na nový systém

Ze setkání se stávajícími uživateli hardwaru docházkového systému vyplynuly následující požadavky na nový systém:

- Systém zvolí typ přístupu pokud možno automaticky
- Uživatel na displeji hardwaru uvidí aktuálně zadávaný přístup i všechny toho dne zadávané přístupy
- Uživatel má při zadání přístupu krátký časový úsek na jeho opravu.

Ze setkání se stávajícími uživateli klientské části docházkového systému vyplynuly následující požadavky na nový systém:

- Klientský software nemusí být na PC neustále spuštěn a ke stáhnutí nových přístupů ze zařízení může dojít po spuštění programu manuálně nebo automaticky.
- Při generování docházky za sledované období, musí uživatel mít možnost úpravy jednotlivých údajů a případných nepřesností
- V systému jsou automaticky uvedeny státní svátky a výpočet času je proveden automaticky

4.2.4 Požadavky na technickou podporu

Odpovědný pracovník provede instalaci nového systému, který poběží paralelně se stávajícím a bude zajišťovat ukládání otisků prstů stávajících i nových

pracovníků(enrollment) do systému. Dokáže také nainstalovat klientskou část na požadované PC společně s instalací MSSQL Express.

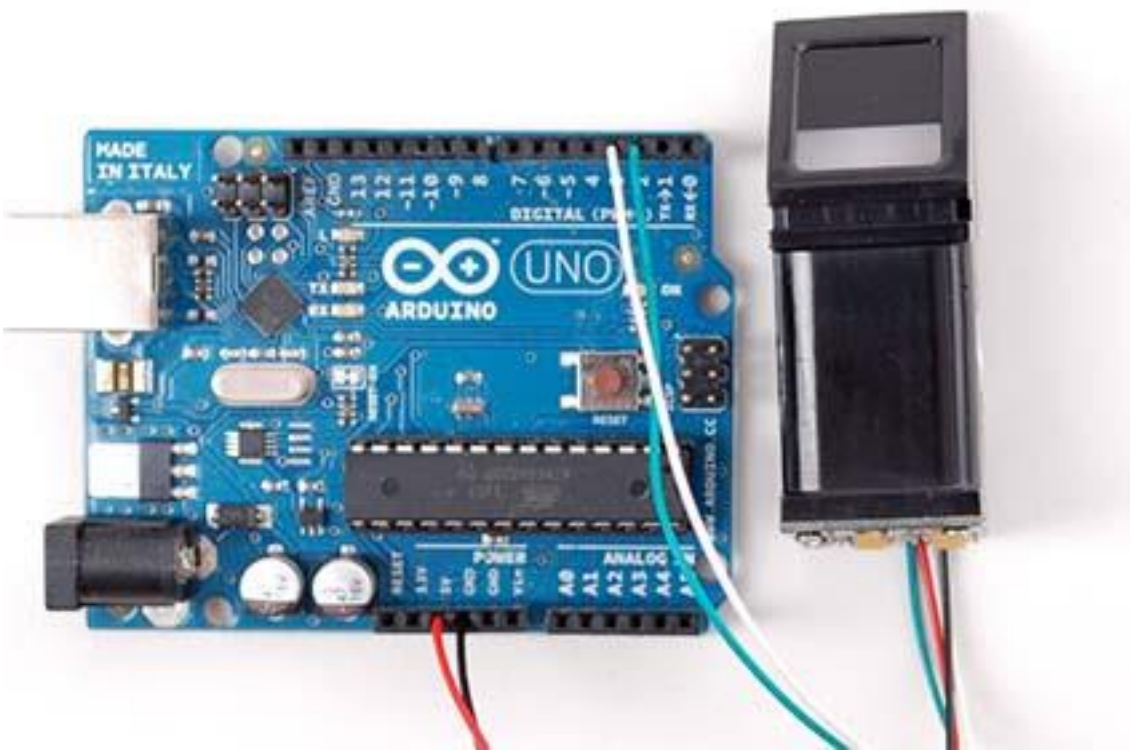
4.3 Návrh a vývoj hardwaru

4.3.1 Návrh a vývoj platformy Arduino

Hardwarová část nového systému bude postavena na platformě Arduino. Z množství různých Arduino desek a jejich klonů na trhu byla vybrána platforma Arduino UNO. Pro zajištění požadované funkcionality jsou nutné tyto přídavné komponenty:

- Arduino LAN Shield – Umožní komunikaci modulu přes Ethernet.
- Slot na SD kartu – Součástí LAN shieldu. Na SD kartu budou ukládány záznamy.
- Čtečka otisku prstu – Pro identifikaci zaměstnanců a přiřazení záznamů.
- 2.8 TFT dotykový kapacitní displej – Pro zadávání typu přístupu a potvrzování voleb.

Díky modulárnosti Arduino shieldů je možná jednoduchá kompletace Arduina UNA, Lan Shieldu i TFT displeje bez nutnosti pájení. To bude nutné pouze pro finální připojení čtečky otisku prstů. Zapojení čtečky otisku prstů je zobrazeno na následujícím obrázku.



Obrázek 9 : Zapojení čtečky otisku prstu

Prvním kontaktem uživatele při příchodu k přístroji bude čtečka otisku prstu. Uživatel přiloží prst a čtečka porovná zadaný otisk s uloženým vzorem. Při nenalezené shodě displej zobrazí chybovou zprávu „Neregistrovaný otisk prstu, opakujte zadání“. Při nalezené shodě s uloženým vzorem se zobrazí UI displeje.

Uživatelské rozhraní Arduina musí být vytvořeno s ohledem na povahu displeje(dotykový). Podle požadavku uživatelů bylo UI displeje navrženo následovně:

- V horní části displeje bude viditelné datum a čas.
- Pro zadání typu přístupu budou sloužit 4 tlačítka (Příchod, Odchod, Pauza Start, Pauza konec).
- Po probuzení displeje přiložením prstu ke čtečce otisků displej zobrazí pouze tlačítka, která dávají v kontextu akce smysl. Při prvním přístupu daného dne systém zobrazí pouze tlačítko Příchod, stejně tak v průběhu dne pokud uživatelem poslední zadané tlačítko bylo začátek Pauzy nabídce systém uživateli pouze tlačítko Pauza konec, protože jiná tlačítka nedávají v situaci smysl.

- Pokud dojde k zadání přístupu, bude každé tlačítko při příštím probuzení pod sebou zobrazovat čas kdy bylo zadáno.
- Při zmáčknutí tlačítka typu přístupu dojde k zobrazení dvou tlačítek OK a X a zároveň mezi nimi bude displej odpočítávat od 9 do 1. Pokud nedojde ke zmáčknutí žádného tlačítka po uplynutí odpočítávání, systém záznam automaticky potvrdí a uloží. Při zmáčknutí tlačítka OK systém záznam uloží a odpočítávání přeruší předčasně, při zmáčknutí tlačítka X systém odpočítávání přeruší a k uložení záznamu nedojde.
- Displej bude aktivní maximálně 30 sekund, po uplynutí doby se displej zneaktivní a na displeji bude pouze výzva k zadání otisku prstu.

Po skončení úspěšného potvrzení typu přístupu(tlačítkem OK nebo uplynutím dané doby) se záznam uloží na SD kartu do csv souboru. Každý nový záznam bude uložen na samostatný řádek.

Záznam ukládaný do csv souboru má formu 0;1;14.02.2018 17:34:14 a jeho pole jsou odděleny středníkem. Jednotlivá pole znamenají – Id uživatele(0-*) ;Typ záznamu (0-7, Příchod, Odchod a 3 pauzy, každá se začátkem a koncem) ; Datum a čas záznamu.

Protože je nutné umožnit klientské stanici přístup k csv souboru bude využita externí Arduino knihovna TinyWebServer. Díky tomu dokáže přistoupit klientská stanice k Arduinu k jako malému webovému serveru a přes http protokol stanice přenese na SD kartě uložený csv soubor do paměti k dalšímu zpracování.

Obrázek 10 : Uživatelské rozhraní na dotykovém displeji Arduina

4.4 Návrh a vývoj software

Softwarová aplikace bude vytvořena v prostředí Visual Studio na platformě WPF v programovacím jazyku C#. Pro uložení dat se využije nainstalovaný MSSQL Server Express.

4.4.1 Návrh databáze

Vytvořená databáze dbo.Dochazka bude obsahovat pouze 2 potřebné tabulky(Record a User), které budou využívány pro evidenci zaměstnanců a pro ukládání záznamů přístupu stažených z Arduina. Tabulka *Record* bude určena pro uložení záznamů a bude obsahovat 3 pole odpovídající polím záznamu uložených v csv souboru tedy pole IdUzivatel(id uživatele 0-*) typu int, TypZaznamu(id zaznamu 0-7) typu int a CasZaznamu(datum a cas) typu datetime. Tabulka bude obsahovat cizí klíč tabulky User, kdy Id uživatele tabulky Record bude odpovídat Id uživatele tabulky User. Následuje skript pro vytvoření tabulky Record.

```
CREATE TABLE [dbo].[Record] (
    [Id]         INT      IDENTITY (1, 1) NOT NULL,
    [IdUzivatel] INT      NOT NULL,
    [TypZaznamu] INT      NOT NULL,
    [CasZaznamu] DATETIME NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_Record_ToTable] FOREIGN KEY ([IdUzivatel]) REFERENCES [dbo].[User] ([Id])
);
```

Obrázek 11 : Skript tabulky dbo.Record

K naplnění tabulky daty dojde automaticky programem při stažení csv souboru z Arduina.

Tabulka *User* bude sloužit k evidenci zaměstnanců. A obsahuje pouze 2 pole Id(Id uživatele odpovídá Id otisku prstu) a Name pro uložení zobrazovaného jména uživatele. Následuje skript pro vytvoření tabulky User.

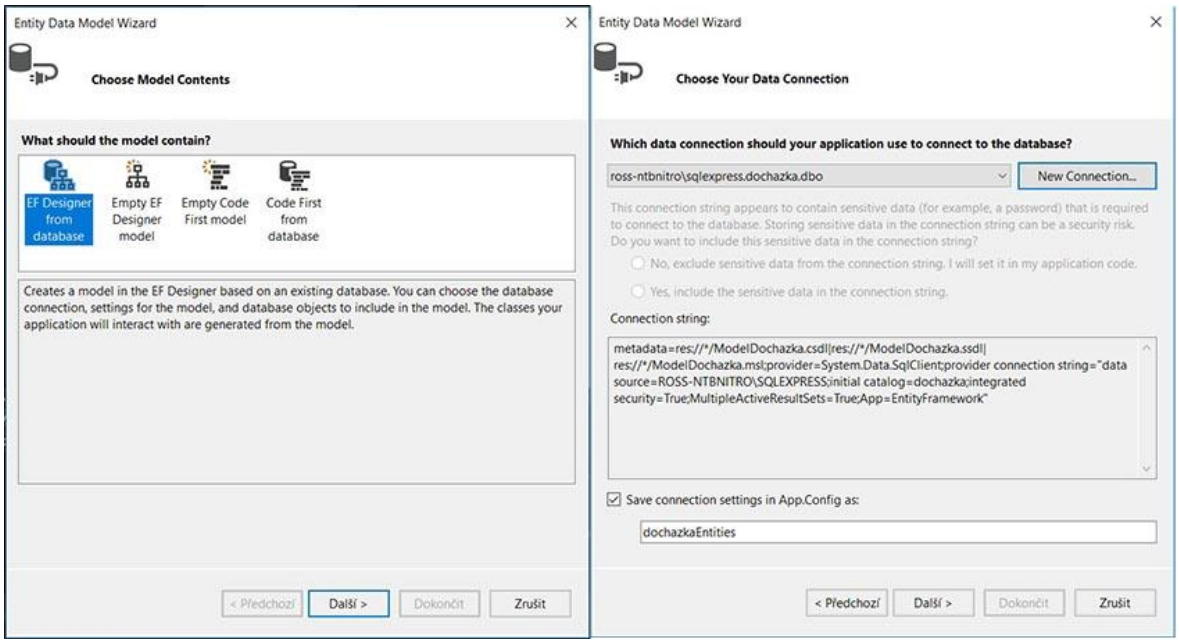
```
CREATE TABLE [dbo].[User] (
    [Id]   INT          NOT NULL,
    [Name] VARCHAR (30) NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Obrázek 12 : Skript tabulky dbo.User

K naplnění tabulky daty dojde manuálně.

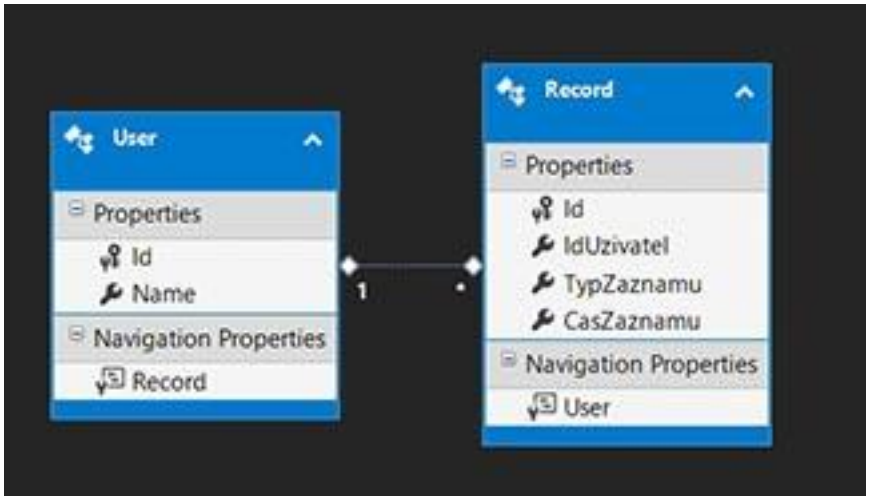
4.4.2 Entity Framework

Pro komunikaci programu s databází slouží v projektu Entity Framework. Díky němu jsou namapovány tabulky databáze na datový model, se kterým WPF aplikace bude pracovat jako s objekty. V rámci Visual Studia zvolíme vytvoření nového modelu a jeho vytvoření z již existující databáze, kterou vybereme v dialogovém okně. Po úspěšném otestování spojení s databází se připojení uloží do konfiguračního nastavení programu.



Obrázek 13 : Proces vytvoření modelu databáze

Po namapování modelu Designerem je možné použít ve zdrojovém kódu třídy User a Record s vlastnostmi odpovídající tabulkám v databázi.



Obrázek 14 : Namapované třídy Entity Frameworkem

4.5 Implementace aplikace

Obslužná aplikace má mít podle zadaných požadavků uživatelů tyto vlastnosti:

- Umět stáhnout z webového server arduina csv soubor, soubor parsovat na List záznamů.
- Záznamy, které již budou obsaženy v databázi nevyužít. Nové záznamy uložit do databáze.
- Po zvolení zaměstnance a požadovaného měsíce vytvoří v aplikaci docházkový přehled pro kontrolu uživatelem.
- Vyexportovat docházkový přehled do samostatného souboru Excel.

4.5.1 Potřebné třídy

Pro požadovanou funkcionalitu programu je tedy potřeba vytvořit třídy pro komunikaci s platformou Arduino, pro práci se záznamy předaných z Arduina, pro komunikaci a práci s databází a pro finální export docházky do požadovaného Excel formátu.

- MainWindow – Hlavní třída projektu, slouží k obsluze událostí kliknutí na tlačítka, k inicializaci formuláře a načtení seznamu zaměstnanců a přehledu měsíců do ovládacích prvků ComboBox.

- Downloader – Třída, jejíž úkol a jediná metoda stáhnutí csv souboru z Arduina. Načtený csv soubor třída parsuje na jednotlivé záznamy, jejichž kolekci metoda vrací jako návratovou hodnotu

```
class Downloader
{
    // Třída stahuje csv soubor z arduina a parsuje jednotlive zaznamy
    Počet odkazů: 0
    public static string IpAdresa { get; set; }

    Počet odkazů: 1
    public static List<Zaznam> StahniCsv()
    {
        List<Zaznam> zaznamy = new List<Zaznam>();

        TextFieldParser parser = new TextFieldParser(@"http://192.168.123.200/dochazka.csv");
        parser.TextFieldType = FieldType.Delimited;
        parser.SetDelimiters(";");

        while (!parser.EndOfData)
        {
            //cteni radku, prvni bunka id uzivatele, druha bunka typ zaznamu, treti cas zaznamu
            string[] fields = parser.ReadFields();
            int uzivatel = Convert.ToInt32(fields[0]);
            int typ = Convert.ToInt32(fields[1]);
            DateTime cas = Convert.ToDateTime(fields[2]);

            Zaznam zaznam = new Zaznam(uzivatel, typ, cas);

            zaznamy.Add(zaznam);
        }
        parser.Close();

        return zaznamy;
    }
```

Obrázek 15 : Ukázka třídy Downloader

- DBHandler – Třída má za úkol předávání a získávání záznamů do/z databáze. Metoda UlozDoDatabaze() za parametr přijímá kolekci záznamů z nichž uloží do databáze pouze ty, které jsou novější než v databázi existující. Metoda NactiZDatabaze() má za parametr id uživatele a zvolený měsíc. Z databáze načte pouze ty záznamy, které odpovídají parametrům. Její návratová hodnota je kolekce třídy Zaznam.

```
57      public List<Zaznam> NactiZDatabaze(int idUzivatele, Mesic mesic)
58      {
59          // všechny záznamy daného uživatele ve zvoleném měsíci
60          var results = db.Record.Where(a => a.User.Id == idUzivatele && a.CasZaznamu.Month == (int)mesic);
61
62          List<Zaznam> zaznamy = new List<Zaznam>();
63
64          // převedení údajů z databáze do objektů Zaznam aby s nimi mohl pracovat PrehledDne
65          foreach (var record in results)
66          {
67              int uzivatel = record.IdUzivatel;
68              int typ = record.TypZaznamu;
69              DateTime cas = record.CasZaznamu;
70
71              Zaznam zaznam = new Zaznam(uzivatel, typ, cas);
72
73              zaznamy.Add(zaznam);
74          }
75
76          return zaznamy;
77
78      }
79
        Počet odkazů: 1
80      public DbHandler()
81      {
82          db = new dochazkaEntities();
83      }
```

Obrázek 16 : Ukázka třídy DBHandler a její metody NactiZDatabaze

- Zaznam – Třída jejíž hlavní vlastnosti odpovídají uloženým datům v Arduinu.

```
7   namespace DochazkovySystem
8   {
        Počet odkazů: 14
9       class Zaznam
10      {
11          // Trida pro jednotlivý záznam z csv souboru v Arduinu
12
            Počet odkazů: 3
13          public int Uzivatel { get; set; }
14
15          //Typ zaznamu pristupu - 1 = Prichod, 2 = Odchod, 3 = Pauza start, 4 = Pauza konec
            Počet odkazů: 19
16          public int Typ { get; set; }
17
            Počet odkazů: 15
18          public DateTime Cas { get; set; }
19
20          // Konstruktor vytvoří instanci pouze pokud budou existovat všechny parametry
            Počet odkazů: 2
21          public Zaznam(int uzivatel, int typ, DateTime cas)
22          {
23              Uzivatel = uzivatel;
24              Typ = typ;
25              Cas = cas;
26          }
```

- PrehledDne – Třída, která jako parametr přijímá kolekci záznamů. Tato kolekce musí obsahovat záznamy jednoho dne. Třída převede záznamy do lidsky srozumitelné podoby viditelných začátečních a koncových časů přístupu.
- ExcelExporter – Třída vytvořená s pomocí externí třídy EPPlus pro tvorbu a čtení Excel dokumentů. Jako parametr použije třída kolekci denních přehledů v rámci jednoho měsíce. Vytvoří nový excel dokument, který naformátuje a doplní daty

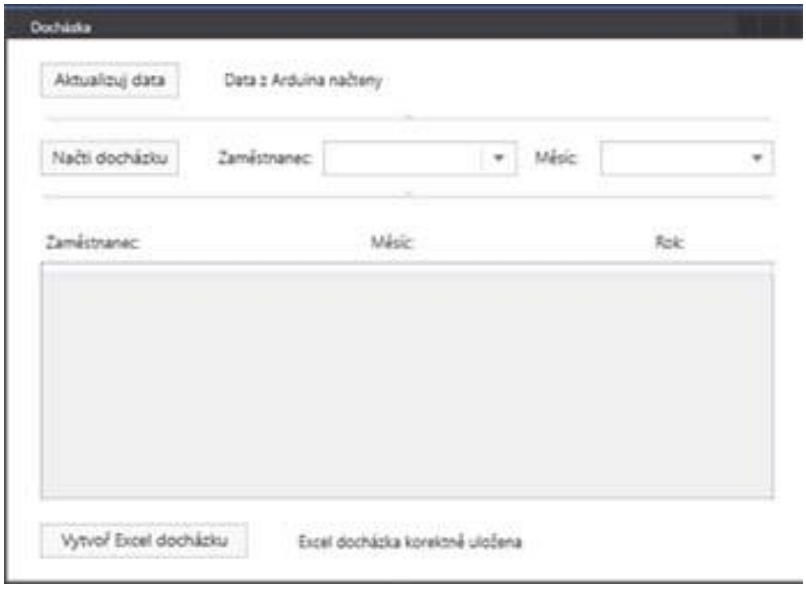
převzatých z parametru. Nakonec uloží vytvořený Excel dokument do počítače pod jménem uživatele a sledovaného měsíce docházky.

4.5.2 Hlavní WPF ovládací prvky aplikace

V rámci uživatelského rozhraní bude aplikace využívat několik prvků pro podporu své funkcionality. Tyto ovládací prvky jsou uvedeny zde:

- Window – Hlavní okno programu
- Grid – Ovládací prvek pro umístění všech ostatních ovládacích prvků
- Button – Tlačítko pro vyvolání události.
- Label – Textový obsah
- Gridsplitter – Ovládací prvek pro oddělení.
- ComboBox – Ovládací prvek pro výběr položky z kolekce
- DataGrid – Komponenta zobrazující větší množství dat

Tyto ovládací prvky budou použity následovně: Aplikace bude obsahovat celkem 3 buttony. Prvním tlačítkem bude tlačítko btnAktualizuj pro aktualizaci dat v databázi. Po korektně provedené aktualizaci label lblAktualizuj zobrazí potvrzení správně provedené aktualizace. ComboBoxy cboxZamestnanec a cboxMesic načtou při inicializaci programu z databáze všechny zaměstnance respektive seznam měsíců. Tlačítko btnNactiDochazku na základě hodnot v ComboBoxech načte po stisknutí odpovídající denní přehledy do ovládací prvku DataGrid. Uživatel bude v případě možnosti moci hodnoty v DataGridu upravit a nakonec finálně použitím tlačítka btnVytvorExcel vytvořit excelový soubor. Úspěšné i neúspěšné vytvoření a uložení souboru oznámí textový ovládací prvek lblExcel.



Obrázek 17 : Uživatelské prostředí aplikace

4.6 Nasazení a testování

Hardwarová část je umístěna vedle stávajícího systému zaznamenávání přístupu. Po zkušební dobu fungují stávající i nový systém paralelně vedle sebe a uživatelé se při zaznamenávání přístupu přihlašují do obou systémů. Softwarová aplikace byla nainstalována na požadované PC a stejně jako hardwarová část běží paralelně v testovacím režimu. Uživatel(ka) při vytváření měsíčních docházkových přehledů využívá oba programy a oba druhy výstupů jsou porovnávány díky čemuž lze průběžně odstraňovat nalezené chyby.

4.6.1 Testovací scénáře

Po nasazení systému došlo k otestování systému na základě 2 testovacích scénářů:

Scénář 1

Předpoklad: Uživatel se pokusí načíst docházku konkrétního uživatele s id 2 v měsíci únoru.

Krok testu: Zvolení korektního uživatele a měsíce

Očekávaný výsledek: Uživatelský prvek DataGrid zobrazí denní přehled daného měsíce.

Výsledek: Korektní zobrazení.

Scénář 2

Předpoklad: Uživatel se pokusí exportovat upravenou docházku.

Krok testu: Uživatel upraví časové údaje v DataGridu a následně se pokusí docházku exportovat.

Očekávaný výsledek: Program exportuje docházku do souboru Excel.

Výsledek: Korektní uložení souboru na disk.

5 Zhodnocení výsledků

Nový biometrický docházkový systém poskytuje jako nový informační systém řešení sledování docházky zaměstnanců pro výše uvedenou firmu jako logické vyústění snah o zlepšení sledování přístupu. Jako přidaná hodnota je vidět výrazně zpřehledněný systém zadávání informací, který v porovnání s předcházejícím poskytuje zaměstnancům zlepšený komfort při jeho využívání.

Nový docházkový systém poskytuje zvýšenou funkcionalitu a nabízí zrychlení zpracování informací pro mzdové a jiné vnitropodnikové účely. Díky korektně navrženému systému nepředstavuje ani pro nové uživatele překážku pro pochopení jako je systém původní.

6 Závěr

V rámci teoretické části byly definovány jednotlivé pojmy potřebné pro tento projekt. Tyto pojmy se týkaly přehledu biometrických technologií, vývoje informačních systémů, významu systémové architektury a životního vývojového cyklu. Dále byly ukázány technologie pro návrh, implementaci a testování systému. Technologie, které byly pro projekt použity jsou pro hardwarovou část platforma Arduino a pro softwarovou část prostředí .NET s programovacím jazykem C# a databází MS-SQL Express.

V praktické části bylo analyzováno stávající řešení docházkového systému a na základě výsledků navržen nový docházkový informační systém. Tento byl následně paralelně se stávajícím implementován a otestován pomocí připravených scénářů. Finální produkt, jímž je biometrický docházkový systém na principu snímání otisku prstu běží aktuálně v produkčním prostředí ve výše zmíněné firmě v paralelním testovacím provozu se systémem stávajícím

7 Seznam použitých zdrojů

1. ČANDÍK, M. Objektová bezpečnost II. Univerzita Tomáše Bati ve Zlíně 2004, str.39-57. ISBN 80-7318-217-3.

2. RAK, R. Biometrická identifikace a verifikace. In: Security Magazín, Roč. X., vyd. 53, 3/2003. Family media, spol. s. r. o., Praha, 2003, str. 56-59. ISSN 1210-8723.

3. VODA, Z. Průvodce světem Arduina. Bučovice: HW Kitchen, 2015. ISBN: 978-80-87106-90-7.

4. BLUM J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. Wiley,2013. ISBN: 978-1-11854936-0

5. BĚHÁLEK, M. Programovací jazyk C# [online]. VŠB-TU Ostrava, 2008. Dostupné na: www.cs.vsb.cz/behalek/vyuka/pcsharp/text.pdf

6. AGARWAL, V.V -- HUDDLESTON, J. -- RAGHURAM, R. Beginning C# 2008 Databases From Novice to Professional. New York, 2008. ISBN13: 978-1-59059-900-6

7. LACKO L. Mistrovství v Microsoft SQL Server 2012. Praha: Computer press, 2013. ISBN 9788025137734.

8. MORKES, D. Microsoft SQL Server 2000 : tvorba, úprava a správa databází. Praha: Grada, 2004. ISBN 80-2470732-2.

9. SHARP J. Microsoft Visual C# 2010. Praha:Computer press, 2010: ISBN 978-80-251-3147-3

10. TVRDÍKOVÁ, M. -- ČESKÁ SPOLEČNOST PRO SYSTÉMOVOU INTEGRACI. Zavádění a inovace informačních systémů ve firmách. Praha: Grada, 2000. ISBN 80-7169-703-6

11. BASL, J. -- BLAŽÍČEK, R. Podnikové informační systémy. Praha: Grada, 2012. ISBN 978-80-247-4307-3

12. NAKOV, S -- KOLEV, V. Fundamentals of Computer Programming with C#. Sofia, 2013. ISBN: 978-954-400-773-7

13. YOSIFOVICH, P. Windows Presentation Foundation 4.5 Cookbook 2012. Birmingham, UK, 2012. ISBN: 978-1849686228

14. ARLOW, J. -- Neustadt, I. UML a unifikovaný proces vývoje aplikací. Praha: Computer Press, 2003. ISBN 80-7226-947-X