Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Problematika internacionalizace softwarových aplikací

Martin Soukup

Bakalářská práce

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Soukup**
Osobní číslo:	**I16142**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Problematika internacionalizace softwarových aplikací**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je přehledné zpracování problematiky týkající se internacionalizace (včetně lokalizace) softwarových řešení. Vybrané aspekty z dané problematiky budou demonstrovány při vývoji vlastní ukázkové aplikace. V teoretické části budou popsány pojmy a standardy, související s problematikou internacionalizace. Budou představeny a porovnány vybrané existující možnosti řešení dané problematiky v souvislosti s vývojem softwaru. Důraz bude věnován především aplikacím desktopovým, ale práce se částečně dotkne i webových aplikací a technologií. V praktické části bude proveden návrh a implementace vlastní aplikace, která bude fungovat jako rezervační systém sportovišť. Tato aplikace bude navržena a vytvářena s důrazem na ukázkové aplikování vybraných postupů a řešení z oblasti internacionalizace. Implementace bude provedena v jazyce C#.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 40 normostran**

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

Seznam odborné literatury:

SAVOUREL, Yves XML internationalization and localization Indianapolis, Ind.: Sams, 2001. ISBN 0672320967
SMITH-FERRIER, Guy .NET internationalization: the developer's guide to building global Windows and Web applications Upper Saddle River, NJ: Addison-Wesley, c2007. ISBN 0321341384
DEARDORFF, Darla K a Harvey CHARLES Leading internationalization: a handbook for international education leaders Stylus Publishing : Co-published with AIEA, Association of International Education Administrators, Leaders in International Higher Education, 2018. ISBN 978-1620367841

Vedoucí bakalářské práce: **Ing. Petr Veselý**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2017**
Termín odevzdání bakalářské práce: **12. května 2018**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 20. března 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury. Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše. Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 8. 2020

Martin Soukup

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat svému vedoucímu práce Ing. Petrovi Veselému za odborné vedení, lidský přístup při konzultacích, pomoc a cenné rady.

ANOTACE

Cílem práce je přehledné zpracování problematiky týkající se internacionalizace (včetně lokalizace) softwarových řešení. Vybrané aspekty z dané problematiky budou demonstrovány při vývoji vlastní ukázkové aplikace.

V teoretické části budou popsány pojmy a standardy, související s problematikou internacionalizace. Budou představeny a porovnány vybrané existující možnosti řešení dané problematiky v souvislosti s vývojem softwaru. Důraz bude věnován především aplikacím desktopovým, ale práce se částečně dotkne i webových aplikací a technologií.

V praktické části bude proveden návrh a implementace vlastní aplikace, která bude fungovat jako rezervační systém sportovišť. Tato aplikace bude navržena a vytvářena s důrazem na ukázkové aplikování vybraných postupů a řešení z oblasti internacionalizace. Implementace bude provedena v jazyce C#.

KLÍČOVÁ SLOVA

Internacionalizace, Aplikace, Vývoj, Technologie

TITLE

Internationalization of software applications

ANNOTATION

The aim of the thesis is to elaborate problems related to the internationalization (including localization) of software solutions. Selected aspects of this issue will be demonstrated when developing a custom sample application.

In the theoretical part, the concepts and standards related to the issue of internationalization will be described. Selected existing solutions to the problem of software development will be introduced and compared. Emphasis will be placed on desk-top applications, but work will also partially affect web applications and technologies.

In the practical part, the design and implementation of your own application will be carried out, which will function as a reservation system for sports grounds. This application will be designed and developed with an emphasis on the exemplary application of selected internationalization practices and solutions. Implementation will be done in C # language.

KEYWORDS

Internationalization, Applications, Development, Technology

OBSAH

SEZNAM ILUSTRACÍ

SEZNAM ZKRATEK

UNESCO – The United Nations Educational, Scientific and Cultural Organisation

URL – Uniform Resource Locator

UTF – Unicode Transformation Format

XML – Extensible Markup Language

ISO – International Organization for Standardization

RFC – Request for Comments

ECMA – European Computer Manufacturers Association

ÚVOD

Internacionalizace je návrh a vývoj obsahu produktu, aplikace či dokumentu, který umožňuje snadnou lokalizaci pro cílovou skupinu, která se liší v kultuře, jazyce nebo regionu. Tento termín se v posledních letech stal velmi běžným kvůli exponenciálnímu nárůstu komunikace. Úspěšná internacionalizace produktů otevírá firmám brány do celosvětového obchodu 21. století.

Bakalářská práce je zaměřena na internacionalizaci softwarových produktů. V první části se čtenář seznámí s pojmy, které s touto problematikou souvisí.

Následuje teoretická část, která nejdříve popisuje základní předpoklady pro internacionalizaci. Dále se práce ubírá stručným náhledem do internacionalizace mobilních a webových aplikací. Zde je popsáno, jak lze začít a docílit úspěchu. Největší část práce je ovšem věnována desktopovým aplikacím, konkrétně programovacímu jazyku C#. Čtenář zde může nastudovat podrobný popis postupů a tříd a knihoven, které jsou základními stavebními kameny internacionalizace.

Poslední kapitolou je část praktická, která obsahuje tři ukázky. Dvě první ukázky jsou krátkého rozsahu a pouze ukazují možná, ovšem v jistých ohledech nedokonalá, řešení, které je ale možné využít v menších projektech. Poslední ukázkový projekt je založen na rezervačním systému sportovišť. Zde jsou ukázány možné postupy internacionalizace v praxi. Těmito postupy je možné se v jistých ohledech řídit, ale každý čtenář by měl mít při čtení práce na paměti, že pro každý projekt je nutné jisté postupy přizpůsobit.

1 ÚVOD DO PROBLEMATIKY INTERNACIONALIZACE

1.1 Internacionalizace

Internacionalizace popisuje proces navrhování produktů tak, aby vyhovovaly potřebám uživatelům mnoha národností a kultur, či byly navrženy tak, aby je bylo možné snadno upravit pro dosažení cíle.

Termín internacionalizace je často zaměňován s lokalizací a globalizací.

[1]

1.2 Lokalizace

Lokalizace je proces přizpůsobení obsahu konkrétnímu místu a hlavní součástí tohoto procesu je překlad textů do různých jazyků. Dává produktu vzhled a dojem, že byl vytvořen speciálně pro cílový trh. [1]

1.3 Globalizace

Globalizace je proces integrace kultury země s kulturami ostatními. Je přirozené, že různé kultury mají různá hlediska a postupy. [1]

1.4 Proč firmy internacionalizují?

Internacionalizace může ušetřit mnoho času pro firmu tím, že bude zajištěna funkčnost finálního produktu na všech cílových trzích.

Mezi hlavní výhody patří:

1. Snadnější přizpůsobení obsahu více místům
2. Ušetřený čas
3. Jednodušší údržba

[1]

1.5 Národy a kultury

1.5.1 Národ

Národ je stabilní komunita lidí, vytvořená na základě společného jazyka, historie a území. Je to kulturně – politické společenství, které si uvědomuje svou autonomii, jednotu a zvláštní zájmy. Benedikt Anderson charakterizoval národ jako „představivou komunitu“ a Paul James za „abstraktní komunitu“.

Národ je představivou komunitou v tom smyslu, že existují materiální podmínky pro představení rozšířených a sdílených spojení. Je to abstraktní komunita v tom smyslu, že je objektivně neosobní. [4]

1.5.2 Kultura

Kultura je charakteristika a znalost určité skupiny lidí, zahrnující jazyk, náboženství, sociální návyky, hudbu a umění. Kultura také může být definována jako sdílené vzory chování a interakce. Slovo „kultura“ pochází z francouzského termínu, který vychází z latinského „colere“, což znamená inklinovat k Zemi a růst. [3]

1.5.3 Západní kultura

Tento termín definuje kulturu evropských zemí a ty, které byly silně ovlivněny evropským přistěhovalstvím (Spojené státy). Západní kultura má své kořeny v období řecko – římské éry ve 14. století. [3]

Pro aktuální téma je důležité, že tyto země mají podobné zvyky formátování s tím, jaké známe z České republiky.

1.5.4 Východní kultura

Východní kultura se týká společných norem zemí na Dálném východě a indického subkontinentu. I tato kultura byla během svého raného vývoje silně ovlivněna náboženstvím. Obecně platí, že ve východní kultuře je menší rozdíl mezi sekulární společností a náboženskou filozofií než na západě. [3]

V této kultuře se mohou objevit jazyky a s tím spojené znaky, které naše známé operační systémy nemusí ve výchozím nastavení podporovat.

1.5.5 Latinská kultura

Mnoho ze španělsky mluvících národů je považováno za součást latinské kulturu, ovšem geografická oblast je více rozšířená. Latinská Amerika je obvykle definována jako části Střední Ameriky, Jižní Ameriky a Mexika, kde jsou dominantní jazyky španělština a portugalština. [3]

1.5.6 Kultura Středního východu

Země Středního východu mají společné jen některé rysy. Oblast se skládá přibližně z 20 zemí, kde nejběžnější společná věc je jazyk, ovšem široká paleta dialektů často ztěžuje komunikaci. [3]

Při přizpůsobování aplikací pro tuto kulturu můžeme narazit na jazyky, které se píší a čtou zprava doleva.

1.5.7 Africká kultura

Africký kontinent je nezbytný pro všechny kultury, vznikl zde život, který se asi před sto tisíci lety začal stěhovat do jiných oblastí. Afrika je domovem řady kmenů, etnických a sociálních skupin. V současné době je Afrika rozdělena do dvou kulturních skupin (severní Afrika a subsaharská Afrika). Severozápadní Afrika má silné vazby na Blízký východ, zatímco subsaharská Afrika sdílí odlišné historické, fyzické a sociální charakteristiky. Drsné prostředí bylo velkým faktorem při rozvoji kultury, díky kterému se mezi velkou populací objevilo množství jazyků, kuchyní, umění i hudebních stylů. [3]

1.5.8 Neustála změna

Bez ohledu na to, jaké kultury je dané obyvatelstvo součástí, je jisté, že kultura se mění. I když je ovšem změna nevyhnutelná, minulost by měla být respektována a zachována. Organizace spojených národů založila skupinu s názvem Organizace spojených národů pro vzdělání, vědu a kulturu (UNESCO), která má identifikovat kulturní a přírodní dědictví a chránit jej. [3]

2 PROCES INTERNACIONALIZACE

2.1 Plánování

I přes to, že lokalizace nemusí být nutně naplánována před vývojem aplikace, internacionalizace aplikace ano, protože ve většině případů je velmi obtížné upravit kód poté, co byl vytvořen.

Proces internacionalizace pomůže upravit kód tak, aby vyhovoval budoucímu přidávání nových národních prostředí. Ovšem je třeba mít na paměti, že pro to neexistují standardy, tudíž jisté změny pro jisté národnosti budou třeba provést i v pozdějších stadiích vývoje.

Dokonce existuje i něco jako nadměrná internacionalizace. Někdo může být přesvědčen o tom, že čím více místům a kulturám svůj kód přizpůsobí, tím lépe. Občas nadměrná internacionalizace bere úsilí i drahocenný čas.

Při uvolňování aplikace na různých trzích je třeba vzít v potaz také místní zákony. Dobrým příkladem mohou být hazardní hry, které jsou dokonce v některých zemích nezákonné. Je třeba si hlídat ukládání dat, sdílení a zabezpečení. [2]

2.2 Internacionalizace pro GUI

Na rozdíl od dřívějších standardů kódování textu má Unicode schopnost reprezentovat všechny různé jazyky. Není úplně nutné jej použít, ovšem obecně je to doporučeno, protože celý proces bude mnohem jednodušší. Musíme se ale ujistit, že vše, co chceme v aplikace použít (vstupy, funkce, knihovny a písma), je s Unicode kompatibilní. Toto se projeví pouze za předpokladu, že bude využit programovací jazyk nízké úrovně, protože vyšší programovací jazyky již standardně Unicode podporují.

V Unicode je definováno přes sto tisíc znaků, ale není tam žádný typ fontu či písmo, které by pokrylo všechny tyto znaky. Společnost Google však zahájila iniciativu na vytvoření rodiny písem s názvem „Noto Fonts“, která by měla zahrnovat více než sto jednotlivých písmen, která jsou navržena tak, aby pokryla všechny skripty kódované v Unicode.

Při přípravě překladu textu do různých jazyků je třeba zvážit, jak bude text uveden a jak to ovlivní celý uživatelský dojem. Pokud je plánována lokalizace pro jazyky, které jsou psány a čteny zprava doleva (arabština, hebrejština a perština), tak bude muset být upraveno uživa-

telské rozhraní. Jedním z doporučených postupů je v tomto případě zrcadlení (posouvání nabídek, tlačítek, karet atd.), aby každý uživatel měl z aplikace přirozený pocit.

I formátování textu, jako je kurzíva, tučné písmo a podržení, se liší napříč jazyky. Proto by se nemělo formátování udávat programově, ale překládat celý text včetně formátu.

Některé jazyky budou mít po překladu textu více znaků, tudíž rozhraní aplikace musí být přizpůsobivé. Jednou z možností je při vyšším počtu znaků zmenšovat jejich velikost, aby nebyl narušen uživatelský dojem různými velikostmi ovládacích prvků.

Formáty čísel a dat jsou další věci, které je nutno přizpůsobit. Pro data některé země používají formát měsíc/den/rok, jiné zase den/měsíc/rok. U čísel se jedná hlavně o desetinné oddělovače, většinou se používají čárky či tečky, ale například Indie a další země používají oddělovače naprosto odlišné (seskupují čísla podle vzorců). [2]

2.3 Postup při internacionalizaci mobilních a webových aplikací

2.3.1 Postup na platformě Android

Internacionalizace na Android platformě začíná u souboru strings.xml, který je hlavním zdrojovým souborem pro data. Jedná se o jakousi interní databázi, ze které je možné vkládat hodnoty do kódu a aplikace bude vědět, jakou kulturu aktuálně využít (strings.xml obsahuje identifikátor a sloupce pro jednotlivé kultury). V projektu je uložena ve res/values/strings.xml.

U dalších zdrojů ve složce res/ je třeba k názvu připojit kód kultury, například pro animace: res/anim-**cs**. Dále také k těmto zdrojům musí existovat jeden zdroj výchozí bez kódu kultury, protože při nenalezení zdroje kultury je automaticky vybírán výchozí zdroj. Pokud neexistuje ani výchozí zdroj, aplikace spadne.

Nejdůležitější postupy při internacionalizaci na platformě Android:

- **Vytvoření flexibilního designu** – navrhnout rozhraní tak, aby vyhovovalo každé předpokládané kultuře
- **Testování na emulátoru** – vše je možné otestovat na jednom stroji přímo při vývoji

[10]

2.3.2 Postup na platformě iOs

Systém iOs také pracuje s řetězci, zde má zdrojový soubor příponu .strings. Výchozí název pro lokalizované textové řetězce je Localizable.strings, tento název by neměl být měněn.

Při vytváření řetězcového souboru je třeba definovat klíč a obsah. Klíčem je název řetězce, který se integruje do kódu, následuje obsah v daném jazyce. Každý řádek končí středníkem.

Při vyvíjení v programu Xcode je třeba přidat složky pro dané kultury. V záložce Info je možnost Localization, zde je možné přidat lokalizační složky pro vybrané kultury. Xcode vytvoří adresáře .proj kombinované s kódy zemí. Sem je ještě třeba nakopírovat a následně upravit zdrojový soubor Localizable.strings. [10]

2.4 Postup při internacionalizaci webových aplikací

Technické aspekty internacionalizace webových aplikací (stránek) mají dopad v každém bodě vývojového cyklu. Při internacionalizaci webů se využívají tři klíčové aspekty:

- Struktura adres URL vhodná pro mezinárodní použití
- Cílení na jazyky
- Kopie a obrázky na míru

Pro snazší procházení stránek vyhledávací nástroje (Google, Bing) sledují řadu různých atributů URL adresy, jako jsou umístění či kódy jazyků.

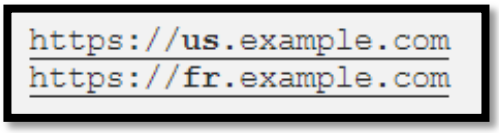
Na úrovni domény je třeba začít oddělením dle geografické oblasti jednou z následujících metod:

- **ccTLDs** (country code top-level domains)

```
https://example.us
https://example.fr
```

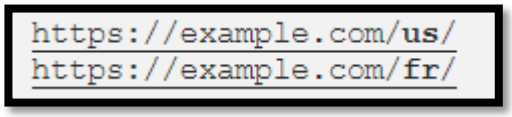
Obrázek 1 – ccTLDs [11]

- **Subdomény**



Obrázek 2 – Subdomény [11]

- **Podadresáře**



Obrázek 3 – Podadresáře [11]

Přidání parametru jazyka (např.: `?lang=cs`) na konec URL adresy je užitečnou alternativou pouze v případě, že jazyk nelze určit žádnou z výše zmíněných metod.

Dále je nutné do hlavičky stránky (`<head>`) přidat atribut `lang="kód jazyka"`.

Pokud není primárně pracováno s asijskými jazyky, je vhodné nastavit kódování webu na UTF8. Je to stejné kódování, jako u URL adres a má širokou podporu prohlížečů pro zobrazování různých znaků v různých jazycích.

Při přizpůsobování a formátování obsahu je vhodné se vyvarovat tagů `<br/>`. Ty se můžou v některých jazycích chovat neočekávaně. Lepší je vždy nastavovat maximální šířku obsahu. U překládaných textů by měly být využity nedělitelné mezery a jiné znaky. [11]

3 INTERNACIONALIZACE DESKTOPOVÝCH APLIKACÍ V .NET C#

3.1 Operační systém

Výběr verze systému Windows, který se má použít pro vývoj aplikace, má významný dopad na funkčnost aplikace. Platí, že nejnovější verze operačního systému poskytují vždy nejlepší podporu. Je minimálně doporučeno používat verze systému Windows, které již mají podporu Unicode. Verze systému může také ovlivnit podporu zrcadlení, která se používá v kulturách zprava doleva (arabština, hebrejština a perština). [5 s. 29]

3.2 Jazyky

Jazyky, které aplikace musí podporovat, diktují mnoho úvah o tom, jak postupovat při internacionalizaci. Latinské jazyky (francouzština, němčina, španělština atd.) jsou pro začátky v internacionalizačních procesech nejlepší volbou, protože nevyžadují tolik složitých postupů, jako například jazyky čtené zprava doleva. Pro podporu těchto jazyků je třeba využít postup zrcadlení, který spočívá v přemisťování ovládacích prvků z jedné strany na druhou a také musí výt vykresleny obráceně (např. rolovací lišty, na kterých se musí otočit hodnoty). Výběr operačního systému zajišťuje také přístupnost znaků různých národů. Novější systému podporují propojování písmen, které umožňuje propojit dvě sady písmen dohromady. Tuto technologii je třeba v situaci, kdy písmo jazyka není přímo podporováno našim operačním systémem (např. glyfy). [5 s. 31]

3.3 Formáty zdrojů

Řetězce, rastry, obrázky, videa, zvuky a další soubory, které jsou snadno přeložitelné, jsou nazývány zdroji. Nejčastěji používaným formátem pro ukládání těchto zdrojů je formát .resx (formát XML). Alternativou je použití databáze k uložení všech zdrojů či XLIFF formátu (XML Localization Interchange File Format). I zde je třeba zvážit klady a zápory ukládání zdrojů. Nejsnadnější je ovšem použít formát .resx, protože je plně podporován .NET Frameworkem a Visual Studiem. Tento formát lze přímo editovat pomocí editoru zdrojů ve Visual Studiu. Ostatní formáty mají značně menší podporu, ovšem někdy nabízí rozsáhlejší in-

dividualizaci, například ukládání některých zdrojů do databáze umožňuje provádět změny pro celý systém za běhu (.resx formát vyžaduje rekompilaci). [5 s. 33]

3.4 Správa zdrojů

Správa zdrojů je využívána proto, aby v systému udržovala kompletní sadu zdrojů pro každý jazyk. Správce zdrojů podporuje následující funkce:

- Udržuje sady zdrojů synchronizované
- Pokud je potřeba překlad zdrojů, provede je automaticky
- Nevyžaduje údržbu
- Dokáže celé zdroje exportovat do jiného projektu či souboru
- Dokáže vytvořit nové zdroje automaticky pro přidání nového jazyka
- Poskytuje správy o chybějících řetězcích napříč zdroji

[5 s. 435]

3.5 Strojový překlad

Strojové překladače jsou programy, které překládají texty. Mohou to být programy desktopové či webové. Strojový překladač může být také službou. Tyto překladače poskytují aplikaci automatický překlad. Je třeba zvážit, zda tento překladač je vůbec třeba (většinou se využívá až ve větších projektech) a do jaké míry budeme jeho služby využívat. Je třeba si uvědomit, že tento překladač nikdy nebude dokonalý a bude dělat chyby. Z tohoto důvodu je nutná jeho údržba, kterou musí zajistit člověk, nejlépe překladatel, který dokáže všechny chyby zachytit a opravit. Ovšem ve velkých projektech je tento postup stále velmi platný, protože většina překladů v dnešní době již bude přesná a může velmi ušetřit náklady a čas. Velmi důležitou úlohu může strojový překlad sehrát v případě, že v aplikaci potřebujeme pseudopřeklad. Tento postup je velmi užitečná strategie při testování internacionalizované aplikace.

[5 s. 42]

3.6 Třída CultureInfo

Tato třída poskytuje informace specifické pro danou kulturu, jako je jazyk, země, kalendář a konvence spojené s danou kulturou. Dále také poskytuje přístup ke konkrétním instancím objektů (např.: DateTimeFormatInfo, NumberFormatInfo, CompareInfo, TextInfo, atd.). Tyto objekty obsahují informace potřebné pro operace specifické pro kulturu, jako je například formátování čísel a dat, porovnávání řetězců a opláštění. Je používána přímo či nepřímo třídami, které formátují, analyzují a manipulují se specifickými údaji kultury. [8]

3.6.1 Názvy a identifikátory kultury

Třída CultureInfo specifikuje jedinečný název pro každou kulturu, který je založen na RFC 4646. Tento název je kombinací dvoupísmenového kódu kultury podle ISO 639 spojeného s jazykem a dvoupísmenového kódu subkultury ISO 3166 spojeného se zemí. Formát názvu kultury založený na RFC 4646 je `languagecode2>-country/regioncode2`, kde `languagecode2` je dvoupísmenový kód jazyka a `country/regioncode2` je dvoupísmenový kód subkultury (příklad: en-US pro Spojené státy americké).

Některé názvy kultur také určují skript ISO 15924. Například Cyrl určuje cyrilický skript a Latn latinský skript. Název kultury, které obsahuje skript, používá formát `languagecode2-scripttag-country/regioncode2` (příklad: `uz-Cyrl-UZ` pro Uzbekistán).

Neutrální kultura je definována pouze dvoumístným malým kódem jazyka (příklad: `fr` pro francouzštinu).

Identifikátor kultury je standardní mezinárodní numerická zkratka a má komponenty nezbytné k jedinečné identifikaci jedné z nainstalovaných kultur. Aplikace může používat vlastní či předdefinované identifikátory. [8]

Tabulka 1 – Příklady

Kultura	Specifická kultura	Anglický název
cs	cs-CZ	Czech
de	de-DE	German
en	en-US	English
fr	fr-FR	French
pl	pl-PL	Polish
ru	ru-RU	Russian

3.6.2 Invariantní, neutrální a specifické kultury

Kultury jsou obecně seskupeny do tří sad: invariantní kultury, neutrální kultury a specifické kultury.

Invariantní je necitlivá na kulturu. Aplikace určuje invariantní kulturu podle názvu, a to pomocí prázdného řetězce (popř. dle jeho identifikátoru). Vlastnost InvariantCulture definuje instanci invariantní kultury. Je spojena s anglickým jazykem, ale neváže se na žádnou zem či region.

Neutrální kultura je kultura, která je spojena s jazykem, ale také se neváže na zem či region. (např: `fr` je neutrální název pro francouzskou kulturu a `fr-FR` je název specifické francouzské kultury). Pro neutrální kulturu se nedoporučuje vytvoření instance třídy CompareInfo, protože obsahuje libovolná data.

Definované kultury mají vlastní hierarchii, kde rodičem specifické kultury je neutrální kultura a té je zase rodičem kultura invariantní. Vlastní kultury by měly definovat vlastnost Parent v souladu s definovanou hierarchií.

Instance tříd DateTimeFormatInfo a NumberFormatInfo nemohou být vytvořeny pro neutrální kultury. [8]

3.6.3 Vlastní kultury

Kromě předdefinovaných kultur podporovaných operačním systémem Windows a .NET Framework podporuje rozhraní .NET Framework tři typy vlastních kultur:

- Nové kultury, které doplňují kultury již dostupné.
- Nahrazující kultury, které mají odlišné vlastnosti od standardních kultur.
- Standardní kultury se změnami uživatelů.

Protože rozhraní .NET Framework podporuje vlastní kultury, práci s daty specifickými pro kulturu by mělo být zváženo:

- Vlastní kultury mohou mít hodnoty, které přesahují rozsah předdefinovaných kultur. Například některé kultury mají neobvykle dlouhé názvy měsíců, neočekávané formáty dat nebo času nebo jiná neobvyklá data.
- Při zobrazení dat specifických pro kulturu v uživatelském rozhraní by měli být respektovány uživatelské úpravy. Například uživatel může chtít 24hodinový formát hodin nebo `rrrrMMdd` formát data.
- Vlastní kultury přepíší výchozí hodnoty, proto nelze považovat data kultur za stabilní.

[8]

3.6.4 Serializace objektů třídy CultureInfo

Při serializaci objektů jsou uloženy pouze dvě hodnoty (Name, UseUserOverride). Objekt je deserializován pouze v případě, když hodnota Name má stejný význam. Tomu tak nemusí být vždy, protože:

- Kultura představená v konkrétní verzi operačního systému nemůže být deserializována ve verzi starší.
- Kultura musí být v systému nainstalována.

[8]

3.6.5 Nastavení aktuální kultury

Pro změnu kultury aktuálního vlákna je třeba provést:

- Konstruktorem CultureInfo(String) vytvořit novou instanci CultureInfo. Konstruktor CultureInfo(String, Boolean) je možné využít v případě přepisování aktuální kultury kulturou stejnou s jinými parametry (např: přidávání skriptu).
- Přiřazení objektu CultureInfo k vlastnosti CultureInfo.CurrentCulture či CultureInfo.CurrentUICulture. V rozhraních .NET Framework 4.5.2 a nižších je možné kulturu přiřazovat přímo vláknům (Thread.CurrentCulture).

[8]

```
using System;
using System.Globalization;

public class Example
{
   public static void Main()
   {
      CultureInfo current = CultureInfo.CurrentCulture;
      Console.WriteLine("The current culture is {0}", current.Name);
      CultureInfo newCulture;
      if (current.Name.Equals("fr-FR"))
         newCulture = new CultureInfo("fr-LU");
      else
         newCulture = new CultureInfo("fr-FR");

      CultureInfo.CurrentCulture = newCulture;
      Console.WriteLine("The current culture is now {0}",
                        CultureInfo.CurrentCulture.Name);
   }
}
// The example displays output like the following:
//     The current culture is en-US
//     The current culture is now fr-FR
```

Obrázek 3 – Nastavení aktuální kultury - příklad [8]

3.6.6 Kultura a vlákna

Když je spuštěno nové vlákno aplikace, jeho aktuální kultura je nastavena na současnou systémovou kulturu, nikoli na aktuální kulturu vlákna. Ve verzích nižších než .NET Framework 4.5 je nejběžnějším řešením předání celého objektu CultureInfo, reprezentující aktuálně nastavenou kulturu aplikace pro delegáta System.Threading.ParameterizedThreadStart. Počínaje .NET Framework 4.5 je možné nastavit kulturu všem vláknům v aplikační doméně tím, že přiřadíme objekt CultureInfo vlastnostem DefaultThreadCurrentCulture a DefaultThreadCurrentUICulture (UI pro uživatelské rozhraní). [8]

3.6.7 Dynamická data

Všechna data kultur, vyjma dat kultur invariantních, jsou dynamická. Země či regiony mohou provést jisté změny a definice dané kultury se změní také. Při ukládání dat by měla aplikace využívat invariantní kulturu, binární či jiný specifický formát na kultuře nezávislý, protože data uložená podle aktuálních hodnot konkrétní kultury se mohou stát nečitelnými či zastaralými pokud se tato kultura změní. [8]

3.6.8 Konstruktory

- **CultureInfo(Int32)** – Inicializuje novou instanci třídy dle identifikátoru kultury
- **CultureInfo(String)** – Inicializuje novou instanci třídy dle jména kultury
- **CultureInfo(Int32/String,Boolean)** – Inicializuje novou instanci třídy na základě identifikátoru či jména. Hodnota Boolean určí, zda se použijí nastavení kultury předdefinované uživatelem.

```
CultureInfo en = new CultureInfo("en");
```

Obrázek 4 – Konstruktor CultureInfo(String) - příklad [9]

[8]

3.6.9 Nejdůležitější vlastnosti třídy

- **Calendar** – Výchozí kalendář aktuální kultury
- **CurrentCulture** – Kultura aktuálního vlákna
- **DateTimeFormat** – Definuje vhodný formát zobrazení dat a časů
- **DisplayName** – Úplný název kultury
- **NumberFormat** – Definuje vhodný formát pro zobrazení čísel

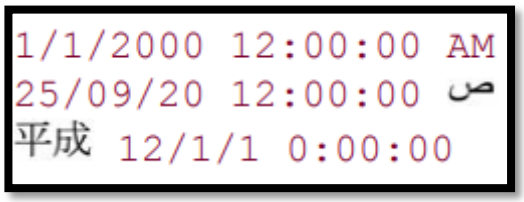
[8]

3.6.10 Vlastnosti Calendar, DateTime a DateTimeFormat

Bez ohledu na to, jak je vytvořena struktura DateTime, časový bod není závislý na žádné dané kultuře, takže například datum 1. 1. 2020 odkazuje na pevný bod v čase, a to bez ohledu na použitý kalendářní systém k jeho vytvoření. Třída DateTimeFormat obsahuje informace o formátování data a času. Ze své vlastnosti Calendar čerpá informace o tom, jak je daný časový bod v aktuálním kalendáři reprezentován. Následující kód poslouží jako ukázka k tomu, jak je možné jednotlivým kulturám přiřadit kalendáře a otestovat formátování.

```
private void Generate()
{
    CultureInfo enCInfo = new CultureInfo("en-US");

    CultureInfo arCInfo = new CultureInfo("ar-SA");
    arCInfo.DateTimeFormat.Calendar = new HijriCalendar();

    CultureInfo jpCInfo = new CultureInfo("ja-JP");
    jpCInfo.DateTimeFormat.Calendar = new JapaneseCalendar();

    DateTime firstJan2000 = new DateTime(2000, 1, 1, new GregorianCalendar());

    Console.WriteLine(firstJan2000.ToString(enCInfo));
    Console.WriteLine(firstJan2000.ToString(arCInfo));
    Console.WriteLine(firstJan2000.ToString(jpCInfo));
}
```

Obrázek 5 – Ukázkový kód kalendářů pro 3 různé kultury



Obrázek 6 – Výsledek kódu pro každou kulturu [5 s. 274]

Přestože jednotlivá vyobrazení 1. 1. 2000 ukazují různé dny, měsíce i roky (a také různé anotace AM / PM), je to stále stejný bod v čase a proměnná `firstJan2000` nemění svou hodnotu. [5 s. 273 – 274]

3.6.11 Vlastnosti Number, Currency a NumberFormatInfo

Čísla a měny se podobají datům a časům i co se týče formátování, ale odpadají složitosti kalendářů:

- Čísla a měny jsou formátovány pomocí třídy NumberFormatInfo.
- Třída CultureInfo má vlastnost NumberFormatInfo pod názvem NumberFormat.
- Je možné si navrhnout vlastní formátovací řetězce nezávislé na zadané kultuře.

[5 s. 282]

3.6.12 Nejdůležitější metody třídy

- **GetCultureInfo(String)** – Vrací instanci kultury dle zadaného názvu kultury. Ještě může mít parametr Int32 či String, String. Vrácená hodnota je jen pro čtení.
- **Clone()** – Vytvoří kopii aktuální kultury. Vhodné hlavně při manuálním vytváření dalších vláken programu.
- **GetCultures(CultureTypes)** – Vrací seznam podporovaných kultur dle parametru.

[8]

3.7 Lokalizovatelné řetězce

Pro ilustraci, jak funguje základní proces internacionalizace, využijeme jeden krátký řetězec `"Ahoj"`, který bude zobrazován prostřednictvím komponenty MessageBox. Tento řádek kódu je nyní pouze v českém jazyce a bude tomu tak vždy. K internacionalizaci tohoto kódu je třeba projít dvěma fázemi. V první řadě je třeba tuto část kódu učinit lokalizovatelnou a poté je třeba ji lokalizovat. Aby tato část kódu byla lokalizovatelná, musíme odebrat textový řetězec z kódu a přesunout ho na nějaký jiný kontejner, který lze za běhu měnit. Zde nám poslouží zdrojové soubory .resx.

Název souboru .resx je důležitý, protože se v kódu používá pro identifikaci zdroje. V menších projektech nám stačí jeden zdrojový soubor, který se váže na celou aplikaci, protože bude obsahovat pouze pár desítek informací, ale pro větší projekty je třeba pro každý soubor (formulář) mít vlastní zdrojový soubor, který bude patřičně pojmenovaný.

Pro příklad je výchozí jazyk čeština a projekt je celkem malý, tudíž je využit pouze jeden zdrojový soubor pro celý projekt. Zdrojový soubor pro výchozí jazyk je pojmenován Zdroje.resx a pro další jazyk, angličtinu například, je zdrojový soubor pojmenován Zdroje.en-US.resx. Výchozí zdrojový soubor také znamená, že bude využit v případě, že pro daný jazyk daný zdroj neexistuje. [5 s. 78]

3.8 Editor zdrojů

Editor zdrojů umožňuje do jednotlivých zdrojů přidávat řetězce, obrázky, ikony, zvuky, soubory a ostatní zdroje. Do pole název vložíme identifikátor prostředku, dle kterého bude

prostředek při lokalizaci identifikován. Do pole hodnota se vkládá daná hodnota prostředku v daném jazyce. [5 s. 80]

Název	Hodnota	Komentář
CenaZnak	Kč	
ChybaDB	Chyba databáze	
NaTentoCasNelzeRezervovat	Na tento čas nelze rezervovat	
PoleRegNevyplnena	Pole nebyla vyplněna správně	
RegUspesna	Registrace úspěšná	
RezervaceVytvorena	Rezervace vytvořena	
RezervaceZrusena	Rezervace zrušena	
UzivatelExistuje	Uživatel již existuje	
UzivatelJmenoHesloNenalezeno	Jméno či heslo nenalezeno	

Obrázek 7 – Editor zdrojů

3.9 Lokalizovaný řetězec

Nyní je možné se vrátit k výše zmíněnému kódu v kapitole 5.6. Přidáme další zdrojový soubor pro jiný jazyk. Pro příklad opět angličtinu. Tudíž nový zdrojový soubor pojmenuje stejně jako výchozí, ale s příponou en-US.resx. Do pole název opět vložíme stejný identifikátor, ovšem hodnota bude v jiném jazyce. Výše zmíněný řetězec již nebude vkládán do komponenty MessageBox, sloužící pro zobrazení, jako text. Místo pevného řetězce bude v kódu následující: `Zdroje.IdentifikatorHodnoty`. [5 s. 88]

3.10 Lokalizace formulářů

V předchozí kapitole byla ukázka překladů řetězců. Stejný postup lze použít i při lokalizaci formulářů, ale tato metoda je velmi zdlouhavá. Každá vlastnost na formuláři by musela být nastavena ručně. Pro lokalizaci formulářů nabízí Visual Studio jednodušší způsob, který bude opět představen na jednoduchém příkladu. [5 s. 113]

Vytvoříme jednoduchý formulář a umístíme na něj jedno tlačítko s nějakým textem (např: Zavřít). Nyní v designeru formuláře (Form1.Designer.cs) je tlačítko vytvořeno kódem, ovšem text tlačítka je psán přímo v kódu. Nejprve je třeba ve vlastnostech formuláře nastavit vlastnost Localizable na hodnotu true. Nyní v designeru formuláře vidíme kód podstatně delší. Nejdůležitější rozdíl je využití nového objektu ComponentResourceManager. Tento správce zdrojů je využíván k načtení všech hodnot a vlastností pro daný formulář a jeho komponenty. Tyto hodnoty a vlastnosti jsou uloženy v novém souboru pro daný formulář (Form1.resx). [5 s. 114]

```
private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(325, 421);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 0;
    this.button1.Text = "Zavřít";
    this.button1.UseVisualStyleBackColor = true;
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(412, 456);
    this.Controls.Add(this.button1);
    this.Name = "Form1";
    this.Text = "Form1";
    this.ResumeLayout(false);

}
```

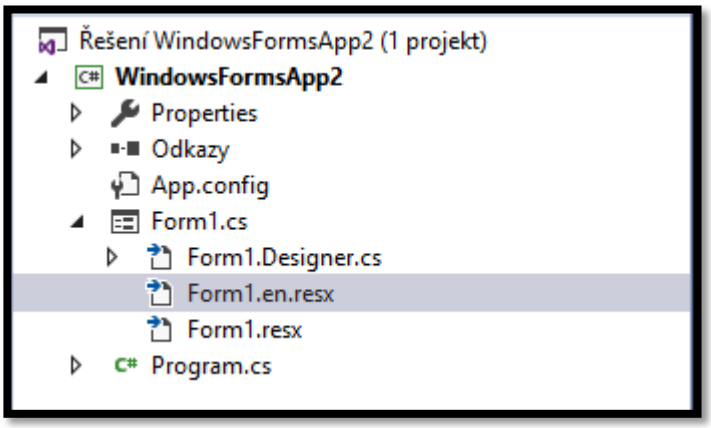
Obrázek 8 – Form1.designer.cs před lokalizací

```
private void InitializeComponent()
{
    System.ComponentModel.ComponentResourceManager resources =
        new System.ComponentModel.ComponentResourceManager(typeof(Form1));
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // button1
    //
    resources.ApplyResources(this.button1, "button1");
    this.button1.Name = "button1";
    this.button1.UseVisualStyleBackColor = true;
    //
    // Form1
    //
    resources.ApplyResources(this, "$this");
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.Controls.Add(this.button1);
    this.Name = "Form1";
    this.ResumeLayout(false);

}
```

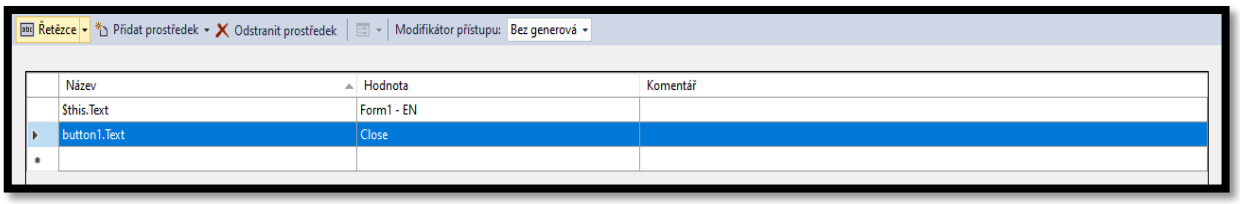
Obrázek 9 – Form1.designer.cs s hodnotou Localizable = True

Nyní můžeme tento vzorový formulář lokalizovat. Ve vlastnostech formuláře je seznam jazyků (vlastnost Language), kde se zobrazí jazyky aktuálně podporované systémem. Opět jako vzor využijeme angličtinu. Po přepnutí formuláře do angličtiny změníme text tlačítka na „Close“. Po uložení změn se vytvoří nový zdrojový soubor pro formulář Form1.en.resx (kód kultury se může lišit dle volby programátora). Výše zmíněný zdrojový soubor bude obsahovat

hodnotu vlastnosti textu tlačítka, ale jiné informaci ne, protože zde se ukládají pouze změny oproti výchozímu nastavení. [5 s. 122]



Obrázek 10 – Vytvořené zdrojové soubory



Obrázek 11 – Obsah Form1.en.resx

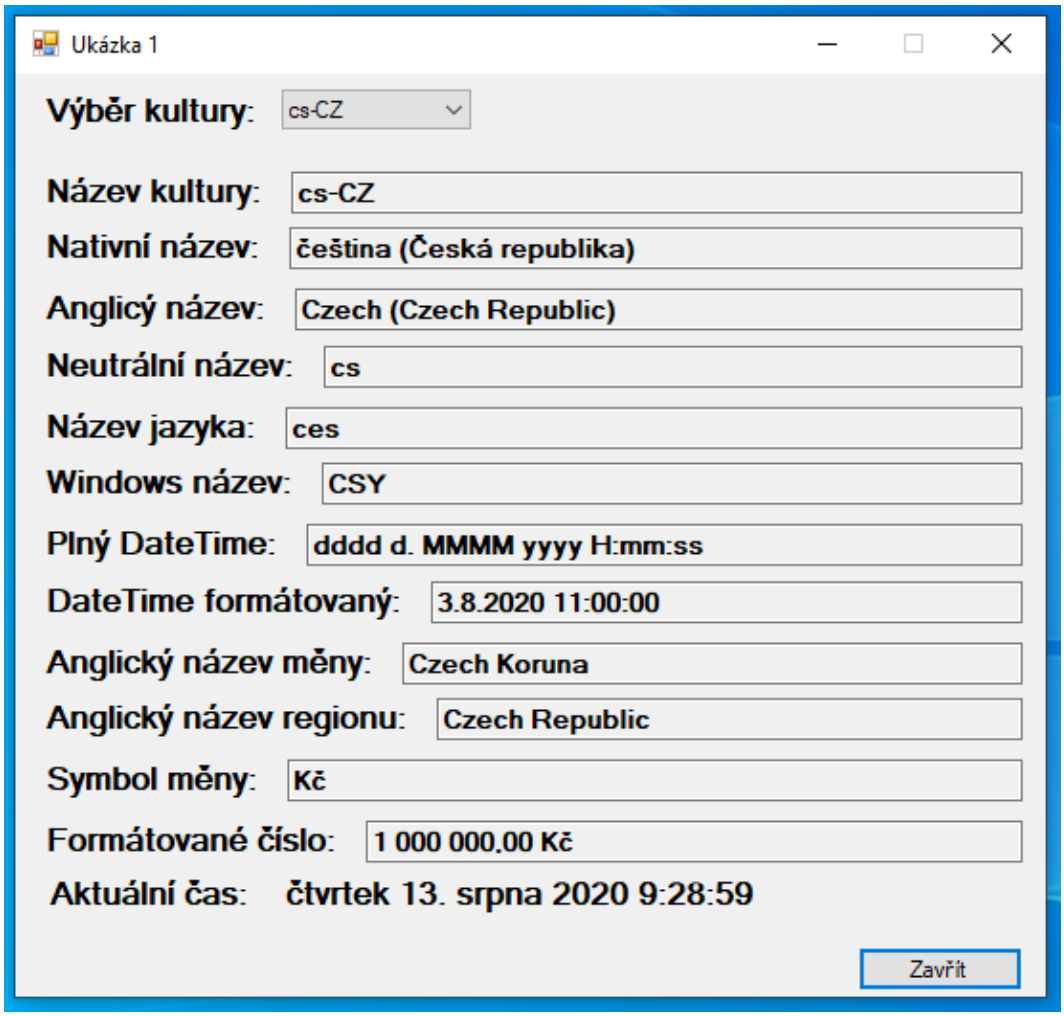
Změny ve výchozím nastavení se promítají do konkrétních jazyků. Pokud však bude změna provedena na anglickém formuláři, tak se vazba na výchozí formulář na danou vlastnost ztratí. Tudíž pokud nyní ve výchozím formuláři přepíšeme text tlačítka, tak se již nezmění na formuláři anglickém, pokud ovšem tlačítko posuneme a na anglickém formuláři s ním ještě tímto způsobem manipulováno nebylo, tak se tato změna projeví i tam. V případě, že by na anglickém formuláři došlo k nechtěnému zásahu a ztrátě vazby na výchozí vlastnost, je třeba ve zdrojovém souboru daného jazyka pro daný formulář vymazat danou vlastnost.

[5 s. 124]

4 NEÚPLNÉ UKÁZKY ŘEŠENÍ A DŮVODY

4.1 Řešení bez zdrojových souborů

Toto řešení zastupuje pouze tu nejjednodušší myšlenku internacionalizace aplikací. Navrhnutý formulář je třeba přizpůsobit i jiné kultuře (pro ukázku byla vybrána kultura `en-US`).



Obrázek 12 – Navrhnutý formulář

Ukázkový formulář má za úkol po spuštění vypsat hlášku o generování a následně vygenerovat informace pro zadanou kulturu.

Ovšem tento příklad se řídí pouze základními programovacími technikami, tudíž nejsou přítomny zdrojové soubory .resx a není využit objekt třídy CultureInfo. Tato volba zanechává pouze jednu možnost, přizpůsobení různým kulturám řešit pomocí rozhodovacích bloků, které musí být všude, kde se data po zadání kultury mění.

```
public frmPR01()
{
    InitializeComponent();
    if(jazyk == "cs-CZ")
    {
        MessageBox.Show("Bude vygenerována ukázka pro jazyk cs-CZ");
        lblKulturaNazev.Text += "cs-CZ";
        lblKulturaNativNazev.Text += "čeština (Česká republika)";
        lblKulturaEngNazev.Text += "Czech (Czech Republic)";
        lblKulturaNeutNazev.Text += "cs";
        lblKulturaJazykNazev.Text += "ces";
        lblKulturaWindNazev.Text += "CSY";
        lblDateTime.Text += "dddd d. MMMM yyyy H:mm:ss";
        lblDateTimeFormat.Text += "3.8.2020 11:00:00";
        lblMenaEngNazev.Text += "Czech Koruna";
        lblRegionEngNazev.Text += "Czech Republic";
        lblMenaSymbol.Text += "Kč";
        lblFormatCisla.Text += "1 000 000,00 Kč";
        btnZavrit.Text = "Zavřít";
        this.Text = "Ukázka 1";
    }
    else if(jazyk == "en-US")
    {
        MessageBox.Show("A sample for the en-US language will be generated");
        lblKulturaNazev.Text += "en-US";
        lblKulturaNativNazev.Text += "English (United States)";
        lblKulturaEngNazev.Text += "English (United States)";
        lblKulturaNeutNazev.Text += "en";
        lblKulturaJazykNazev.Text += "eng";
        lblKulturaWindNazev.Text += "ENU";
        lblDateTime.Text += "dddd, MMMM dd, yyyy h:mm:ss tt";
        lblDateTimeFormat.Text += "8/3/2020 11:00:00 AM";
        lblMenaEngNazev.Text += "US Dollar";
        lblRegionEngNazev.Text += "United States";
        lblMenaSymbol.Text += "$";
        lblFormatCisla.Text += "$1,000,000.00";
        btnZavrit.Text = "Close";
        this.Text = "Expample 1";
    }
}
```

Obrázek 13 – Rozhodovací blok pro dvě kultury

Jak je vidět z výše uvedeného obrázku kódu, toto řešení je naprosto nereálné pro většinu projektů. Není ovšem reálné z důvodu nefunkčnosti, i větší projekty by mohly na této myšlence fungovat, ovšem už jen kód této krátké ukázky je dlouhý a navíc se stále opakuje. Přidávání podpory pro další kultury je do tohoto řešení velice složité. Z toho důvodu není toto řešení doporučováno.

4.2 Řešení pouze dle klasických postupů internacionalizace

Na tuto ukázku je využit formulář z předchozí kapitoly, ovšem nyní s využitím již výše pospaných postupů. Což obnáší objekty tříd CultureInfo a RegionInfo a zdrojové soubory. Nyní má tento formulář již nastavenou vlastnost Localizable na hodnotu True. Jsou přizpůsobeny ovládací prvky a jejich texty. Tyto změny jsou uloženy ve zdrojových souborech formuláře pro jednotlivé kultury.

```
private void VygenerujData()
{
    txtKulturaNazev.Text = cInfo.Name;
    txtKulturaNativNazev.Text = cInfo.NativeName;
    txtKulturaEngNazev.Text = cInfo.EnglishName;
    txtKulturaNeutNazev.Text = cInfo.TwoLetterISOLanguageName;
    txtKulturaJazykNazev.Text = cInfo.ThreeLetterISOLanguageName;
    txtKulturaWindNazev.Text = cInfo.ThreeLetterWindowsLanguageName;
    txtDateTime.Text = cInfo.DateTimeFormat.FullDateTimePattern;
    txtDateTimeFormat.Text = DateTime.Now.ToString(cInfo.DateTimeFormat);
    txtMenaEngNazev.Text = rInfo.CurrencySymbol;
    txtRegionEngNazev.Text = rInfo.EnglishName;
    txtMenaSymbol.Text = rInfo.CurrencyEnglishName;
    txtFormatCisla.Text = 1000000.ToString("C", cInfo.NumberFormat);
    lblAktCasText.Text = DateTime.Now.ToString("F", cInfo.DateTimeFormat);
}
```

Obrázek 14 – Generování dat ukázky

Na výše uvedeném obrázku je vidět kód, který generuje stejné informace, jako ten z předešlé kapitoly. Ovšem zde se již informace načítají přímo, a to dle zadané kultury.

Také zde byla možnost otestovat kulturu arabskou, která je jedna z kultur RTL (Right to left). Zde bylo využito zrcadlení celého formuláře, které by v předchozí kapitole, kde se celý proces psal ručně, bylo až nemyslitelně složité.

Pro vypsání hlášky je využit zdrojový soubor projektu a další informace se již načítají přímo z vytvořené kultury. Toto řešení má velkou řadu výhod oproti řešení v předchozí kapitole, ovšem je zde i výhoda, která není na první pohled úplně viditelná. Jelikož se data kultury načítají přímo z vytvořeného objektu, jsou vždy aktuální. Data kultur se mohou v .NET aktualizovat a poté by řešení v předchozí kapitole ukazovalo zastaralé údaje.

Řešení je plně funkční a může být využito i pro velké projekty, z výše uvedených tříd se dají načíst veškeré informace potřebné k formátování aplikace pro danou kulturu (kalendáře, čísla, atd.). Ovšem většina projektů pracuje i s dalšími systémy, jako jsou například databáze. V ten moment přichází do úvahy, co a jak se bude řešit, protože externí data aplikace (z databáze) jsou dynamická a jejich obsah nelze předvídat. Tento problém bude popsán v následující kapitole.

5 NÁVRH A ŘEŠENÍ INTERNACIONALIZACE V .NET C#

5.1 Analýza

5.1.1 Popis problému

Bezejmenná společnost plánuje na svém pozemku vystavět komplex sportovišť, které bude hodinově pronajímán veřejnosti. Z důvodu velké poptávky v dané lokalitě by bylo velmi náročné vyřizovat každého přímo pracovníkem na recepci, tudíž tato společnost potřebuje vytvořit program, který bude veškerou rezervaci řešit. Tento systém poběží na terminálech u vstupu do tohoto komplexu. Jelikož je tento komplex v turistické destinaci, kam se sjíždějí turisté zejména z Evropy, musí tento systém umožňovat pohodlné používání různým národům.

5.1.2 Budoucí stav

Rezervační systém poběží na předem připravených terminálech majitel komplexu. Veřejnost bude mít možnost se do systému zaregistrovat. Po registraci bude možné rezervovat volná sportoviště na daný čas. Rezervace bude možné měnit či rušit v reálném čase. Předpokládají se tři typy uživatelů. Klasický uživatel uvidí všechna sportoviště a naplánované rezervace. Do volných časů může přidávat rezervace vlastní. Správce bude mít možnost filtrovat všechny rezervace, popřípadě je také přidávat po dohodě s uživateli (např. po telefonické dohodě). Administrátor jako správce celého systému bude mít možnost přidávat do systému nová sportoviště, protože majitel komplexu plánuje jeho budoucí rozšíření. Systém bude přizpůsobitelný národům z Evropy, počítá se s přizpůsobením České republice, Anglii a Rusku.

5.2 Použité technologie

5.2.1 Visual Studio

Visual studio je vývojové prostředí od společnosti Microsoft, které je hojně využíváno pro vývoj aplikací konzolových, aplikací s grafickým rozhraním (Windows Forms) a dále také webových stránek či služeb, a to jak ve strojovém kódu, tak i v řízeném kódu na platfor-

mách od společnosti Microsoft (Microsoft Windows, Windows Mobile a dalších). Nejnovější verze Visual Studia je z roku 2019.

Hlavním požadavkem na prostředí Visual Studio je snadný návrh formulářů při internacionalizaci.

Nejdůležitějšími prvky Visual Studia jsou:

- **Editor kódu**, který podporuje zvýraznění syntaxe a automatické dokončení za použití IntelliSense**.**
- **Debugger** může být přiřazen běžícím procesům a debugovat je. Pracuje jak se strojovým, tak se spravovaným kódem.
- **Designery**, které umožňují snadné navrhování aplikací
- **Rozšiřitelnost**

Visual Studio podporuje následující produkty:

- **Microsoft Visual C++**
- **Microsoft Visual C#**
- **Microsoft Visual Basic .NET**
- **Microsoft Visual F#**
- **Microsoft Visual Web Developer a Team Foundation Server**

[6]

5.2.2 C#

C# je vysokoúrovňový objektově orientovaný jazyk programovací jazyk vyvinutý společností Microsoft. Společnost tento jazyk založila na jazycích Java a C++. Tento jazyk lze využít k vyvíjení formulářových aplikací pro Windows, aplikací pro mobilní zařízení a aplikací webových. Standard ECMA definuje současný C# jako jednoduchý, moderní a mnohoúčelový jazyk.

Podpora vysoké úrovně internacionalizace je zde hlavním požadavkem na programovací jazyk.

Hlavní vlastnosti programovacího jazyka C#:

- **Vícenásobná dědičnost v C# neexistuje**
- **Neexistují globální proměnné a metody**
- **Je typově bezpečnější než C++**
- **Není důležité pořadí deklarace metod**
- **Rozlišuje malá a velká písmena**

[7]

5.2.3 MySQL

MySQL je open – source relační databáze. Relační databáze organizuje data do jedné či více tabulek, ve kterých mohou typy dat souviset (tyto vztahy strukturují data). SQL je programovací jazyk, který se používá k vytváření, úpravám a dalším operacím s daty v relační databázi.

Hlavní funkce dostupné v MySQL:

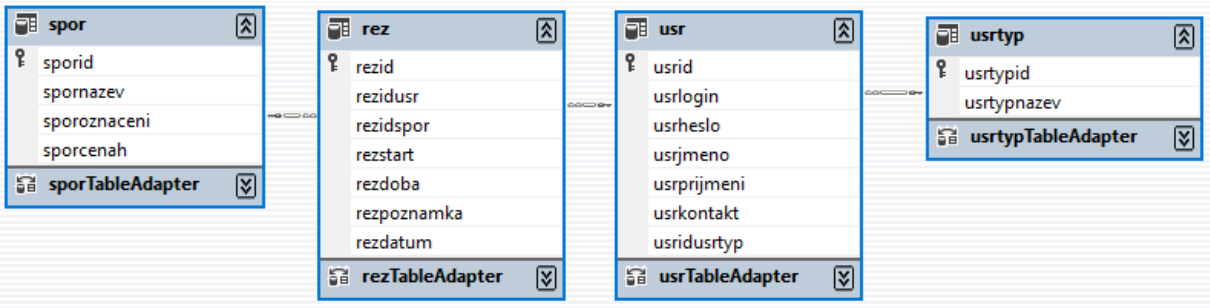
- **Podpora napříč platformami**
- **Informační schéma**
- **Transakce se záchytnými body**
- **Ukládaní dotazů do mezipaměti**

[12]

5.3 Implementace

5.3.1 Databáze

Lokální databáze obsahuje data aplikace v tabulkách. Ukládají se informace o uživatelích, jako například jméno, přihlašovací údaje a kontakt. Tabulka výčtů ukládá typy uživatelů. Do tabulky sportovišť jsou ukládány informace o jednotlivých sportovištích a v tabulce rezervací jsou uloženy identifikátory uživatelů a sportovišť, datum, čas a doba trvání rezervace.



Obrázek 15 – Databáze a vztahy tabulek

Na výše uvedeném obrázku jsou vidět sloupce a vztahy mezi jednotlivými tabulkami. Tabulka uživatelů se váže na tabulku typů uživatelů, která určuje tři druhy uživatelů (klasický uživatel, správce a administrátor). Nejdůležitější tabulkou je tabulka rezervací, která se váže na identifikátor uživatele a sportoviště.

Každá tabulka má vlastní primární klíč a s databází pracuje aplikace pomocí klasických SQL příkazů.

5.3.2 Práce s databází

Rezervační systém komunikuje s databází prostřednictvím třídy DB. Ta obsahuje jednu privátní proměnou, která udržuje spojení s databází. Dále také obsahuje metody pro připojení a odpojení od databáze.

```
private static SqlConnection con;

Počet odkazů: 1
public static bool Pripojit()
{
    if (con == null)
    {
        string conStr = "Data Source=(LocalDB)\\MSSQLLocalDB;" +
            "AttachDbFilename=C:\\Users\\SOUKUP\\Desktop\\SOUKUP_RS_BP\\SOUKUP_RS_BP\\DB.mdf;" +
            "Integrated Security=True";
        con = new SqlConnection(conStr);
        try
        {
            con.Open();
            return true;
        }
        catch
        {
            return false;
        }
    }
    return false;
}

Počet odkazů: 1
public static void Odpojit()
{
    if(con != null)
    {
        con.Close();
        con = null;
    }
}
```

Obrázek 16 – Třída DB

Poté třída DB definuje další podtřídy, které slouží pro práci s jednotlivými tabulkami (DB.USR, DB.SPOR, DB.REZ). Tyto třídy zajišťují přidávání, odebírání, popřípadě editaci a výběr dat z tabulek pro zobrazení v aplikaci (např. v DataGridView).

Třída DB.SPOR obsahuje vlastní třídu ComboItem, která slouží pro zobrazování dat v ovládacím prvku ComboBox.

```
public class ComboItem
{
    private int value;
    private string text;

    Počet odkazů: 14
    public int Value
    {
        get { return value; }
    }

    Počet odkazů: 0
    public string Text
    {
        get { return text; }
    }

    Počet odkazů: 1
    public ComboItem(int Value, string Text)
    {
        this.value = Value;
        this.text = Text;
    }

    Počet odkazů: 0
    public override string ToString()
    {
        return text;
    }
}
```

Obrázek 17 – Třída ComboItem

5.3.3 Postup internacionalizace

Prvním krokem ještě před samotnou internacionalizací je nutnost podrobně rozplánovat celý projekt. Tento rezervační systém byl rozplánován do několika formulářů, které budou popsány níže. Každý formulář obsahuje několik ovládacích prvků, jejichž uspořádání musí vyhovovat všem zemím či kulturám, pro které je internacionalizace plánována.

Vstupem je volba jazyka či země při spouštění, tím celé aplikaci určíme, v jakém režimu má běžet. Výběr je řešen ovládacím prvkem ComboBox, který pak vrací kód jazyka. Tento kód použijeme pro vytvoření nového objektu CultureInfo, jehož vstupním parametrem řetězec s kódem a právě do tohoto objektu se po jeho vytvoření uloží veškeré užitečné informace. Nyní již můžeme běžící aplikaci říci, v kterém režimu (národnosti) má běžet. Tato změna probíhá na úrovni vláken aplikace. Pro tuto akci je také potřeba připojit jmenný prostor System.Globalization.

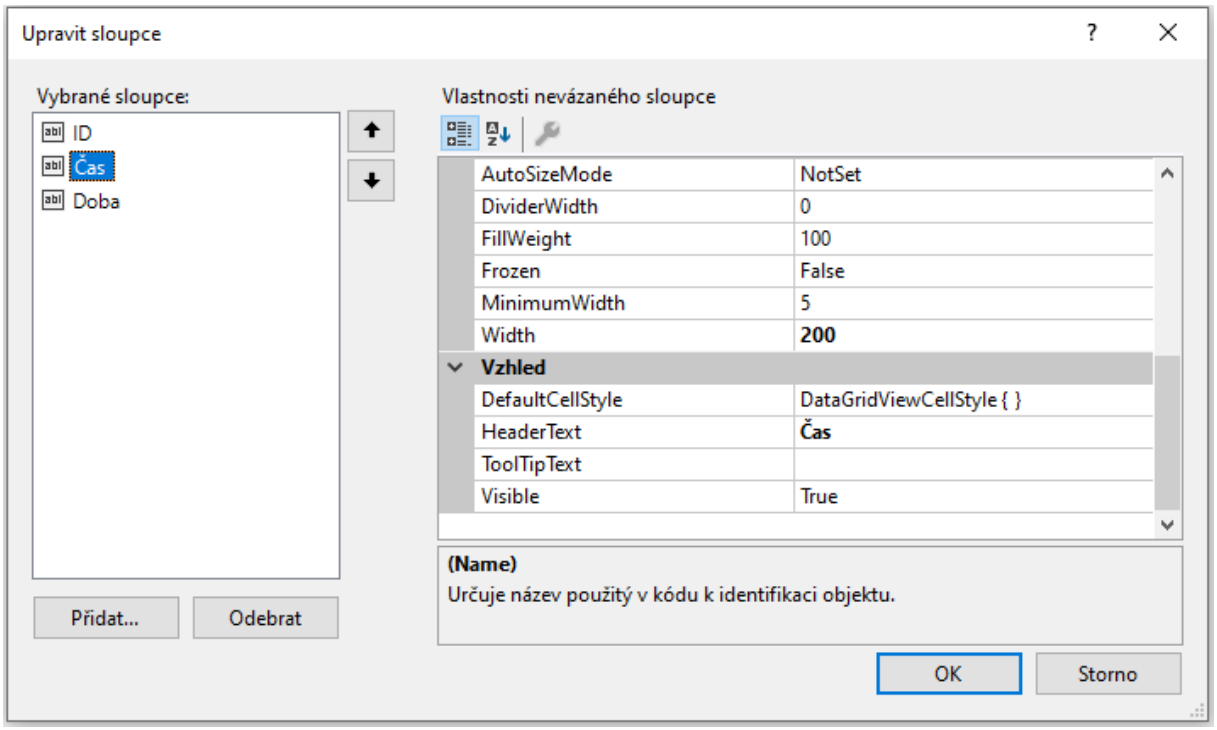
```
Data.cInfo = new CultureInfo(Data.Jazyk);
CultureInfo.DefaultThreadCurrentCulture = Data.cInfo;
CultureInfo.DefaultThreadCurrentUICulture = Data.cInfo;
```

Obrázek 18 – Vytvoření objektu třídy CultureInfo uloženého v třídě Data a nastavení kultury vláknu

Dalším krokem je přizpůsobování formulářů a ovládacích prvků. První akcí v návrháři je třeba každému formuláři nastavit vlastnost Localizable na hodnotu `true`, aby pro každý jazyk byl generován zvlášť zdrojový soubor formuláře. Ten se vygeneruje po změně hodnoty vlastnosti Language. Tato vlastnost má také hodnotu default (výchozí). Ve výchozím nastavení je nejprve celý formulář navržen pro pozdější usnadnění práce, protože po uložení všech změn je další zdrojový soubor kopií té výchozí a stačí pouze upravit ty věci, které jsou potřeba.

Změny textů ovládacích prvků na formuláři jsou prováděny změnou vlastnosti Text jednotlivých ovládacích prvků. Poté můžeme upravit velikost a uspořádání, aby vizáž aplikace byla jednotná. Některé jazyky mohou mít texty delší, tudíž je nutné formulář i ovládací prvky zvětšit a přerovnat. Tyto akce lze jednoduše provést posunováním jednotlivých prvků v návrháři.

Úprava sloupců zobrazených z databáze v DataGridView provedeme pravým kliknutím na tento ovládací prvek a vybereme možnost upravit sloupce.



Obrázek 19 – Úprava sloupců v DGV

Ostatní kroky internacionalizace se musí řešit přímo v kódu a liší se dle potřeb a určení dané aplikace. V tomto případě pracujeme s daty, časy a měnami.

Datum se zapisuje odlišně v různých koutech světa, tudíž ovládacím prvkům, kde vybíráme například datum rezervace, musíme určit, v jakém formátu se datum zobrazí. To zajistíme tím, že ovládacímu prvku DateTimePicker nastavíme vlastní formát, který udržuje již výše zmíněný objekt třídy CultureInfo.

```
dtmDatum.Format = DateTimePickerFormat.Custom;
dtmDatum.CustomFormat = Data.cInfo.DateTimeFormat.ShortDatePattern;
```

Obrázek 20 – Nastavení formátu data

ShortDatePattern odstraní z data čas a zobrazí ho v klasické krátké podobě pro daný region. Tomuto ovládacímu prvku bylo také nastaveno, aby rezervace byly možné pouze na další den a 7 dní následujících.

Čas se vybírá samostatně, ovšem ne vždy stejně. Vybíráme celé hodiny, ale někde se používá 24 hodinový formát, zatímco například anglické národy používají formát 12 hodinový. Tento problém vyřešíme tím, že ovládacímu prvku pro výběr hodiny řekneme minimální a maximální hodnotu. Nyní je ještě třeba přidat ovládací prvek, podle kterého je určeno, zda je dopoledne či odpoledne. Číslo 13 nebude v tomto formátu nikdy dosaženo, jde pouze o číslo pro algoritmické účely.

```
if (Data.Jazyk == "en-001")
{
    numZacatek.Maximum = 13;
    rdbAM.Visible = true;
    rdbPM.Visible = true;
}
```

Obrázek 21 – Nastavení na 12 hodinový formát

Práce s měnami je v každém rezervačním systému naprostá samozřejmost. Tento rezervační systém ovšem nemá jistotu stálého připojení k internetu, tudíž byly využity statické kurzy ke dni 21. 4. 2020. O převod se stará dělící a násobící algoritmus.

V případě potřeby je možné využít například finance.yahoo.com. Po předání parametrů (měny pro převod) využijeme třídu WebClient a její metodu DownloadString, která nám vrátí aktuální kurz.

Řešit musíme také řetězce, které generuje přímo v kódu. Pro překlad těchto řetězců jsou využity zdrojové soubory, do kterých ukládáme název a hodnotu. Soubor pro český jazyk má název Zdroje.resx, pro další jazyky bude mít vždy název Zdroje.KULTURA.resx (např. Zdroje.ru.resx) a podle nastaveného regionu daného vlákna bude vybrán příslušný zdrojový soubor. Název musí být u všech hodnot ve zdrojových souborech stejný, aby byla vybrána správná hodnota. V jiném případě bude použit výchozí zdrojový soubor.

```
MessageBox.Show(Zdroje.RezervaceVytvorena);
```

Obrázek 22 – Použití zdrojového souboru

Název	Hodnota	Komentář
CenaZnak	Kč	
ChybaDB	Chyba databáze	
NaTentoCasNelzeRezervovat	Na tento čas nelze rezervovat	
PoleRegNevyplnena	Pole nebyla vyplněna správně	
RegUspesna	Registrace úspěšná	
RezervaceVytvorena	Rezervace vytvořena	
RezervaceZrusena	Rezervace zrušena	
UzivatelExistuje	Uživatel již existuje	
UzivatelJmenoHesloNenalezeno	Jméno či heslo nenalezeno	

Obrázek 23 – Ukázka zdrojového souboru

Dynamická data z databáze (např. názvy sportovišť) je třeba přeložit také. Toto je řešeno pomocí další databázové tabulky. Ta obsahuje tři sloupce:

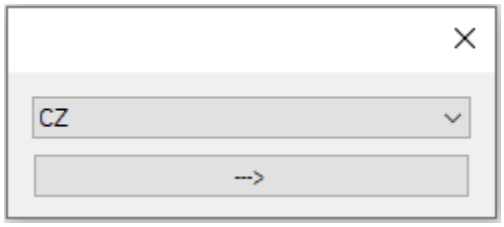
- **Název**, který zůstává stejný pro všechny jazyky
- **Hodnota**, která udává hodnotu textu pro daný jazyk
- **Jazyk**, který pro danou hodnotu definuje jazyk

Při vypisování těchto dat je třeba v kódu doplit překlad. Ten zajišťuje metoda v třídě DB, která načte aktuální kulturu aplikace a název překládaného prvku. Na základě těchto dat vybere z databáze vhodný překlad. Formulář, určený pro správu těchto překladů, je popsán v kapitole 5.4.8.

5.4 Formuláře

5.4.1 Výběr jazyka

Jako první se spouští formulář pro výběr jazyka. Obsahu pouze dva ovládací prvky, ComboBox a Button. V ComboBoxu vybereme jazyk, který se stiskem tlačítka nastaví, proběhne kontrola a připojení k databázi. Tento formulář se skryje, ale jeho vlákno zůstává běžet, a objeví se formulář pro přihlášení.

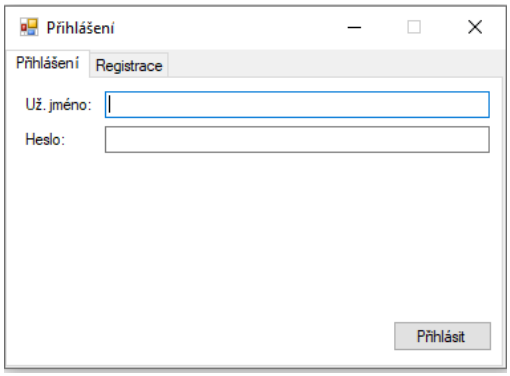


Obrázek 24 – Výběr jazyka

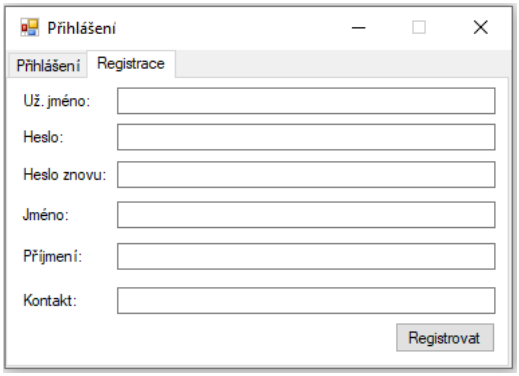
5.4.2 Přihlášení

Formulář pro přihlášení obsahuje dva tabulátory (TabPage). Na prvním jsou dvě pole s popisky sloužící pro zadání přihlašovacích údajů do systému. Stiskem tlačítka se ověří platnost uživatele a uživatel samotný je přihlášen. Po této akci se dostaneme k hlavnímu oknu systému.

Druhou záložkou je registrace, kde uživatel vyplní své údaje a po stisknutí tlačítka dojde k ověření platnosti údajů a následné registraci do systému.



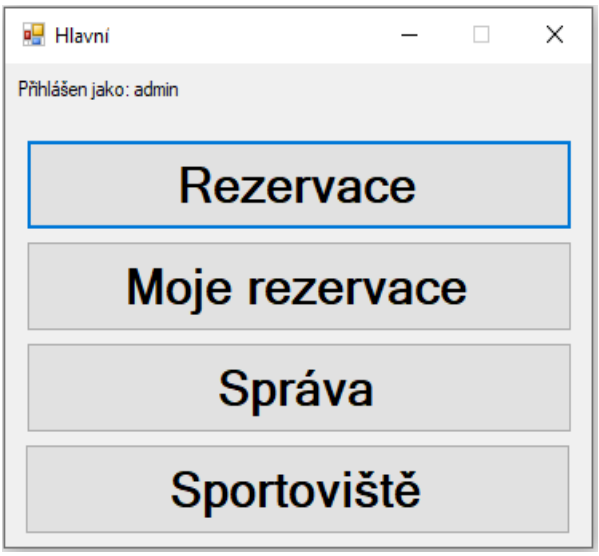
Obrázek 25 – Přihlášení



Obrázek 26 – Registrace

5.4.3 Hlavní okno

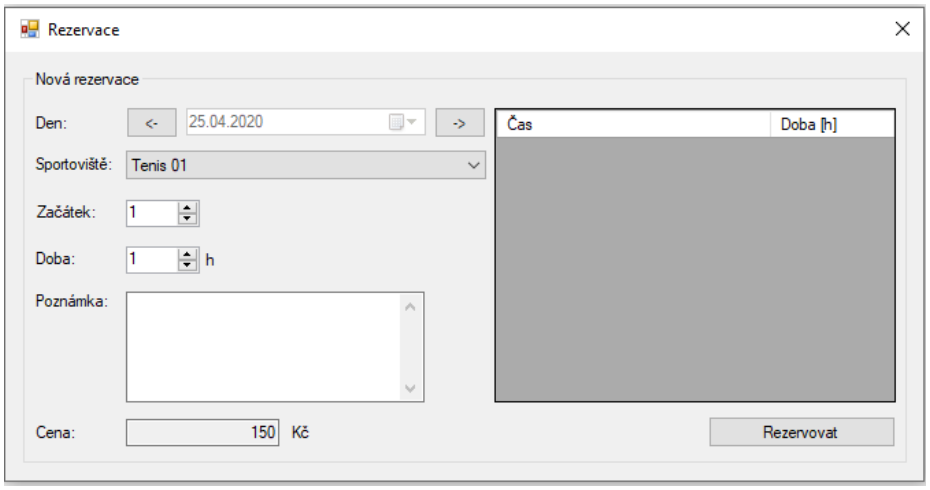
Na horní straně formuláře uživatel vidí své přihlašovací jméno a dle svých práv v aplikaci také další možnosti. Klasičtí uživatelé vidí pouze přehled svých rezervací a mají možnost vytvořit novou rezervaci. Správci (obsluha) má navíc možnost vidět rezervace všech uživatelů a administrátoři mohou spravovat i samotná sportoviště.



Obrázek 27 – Hlavní okno

5.4.4 Rezervace

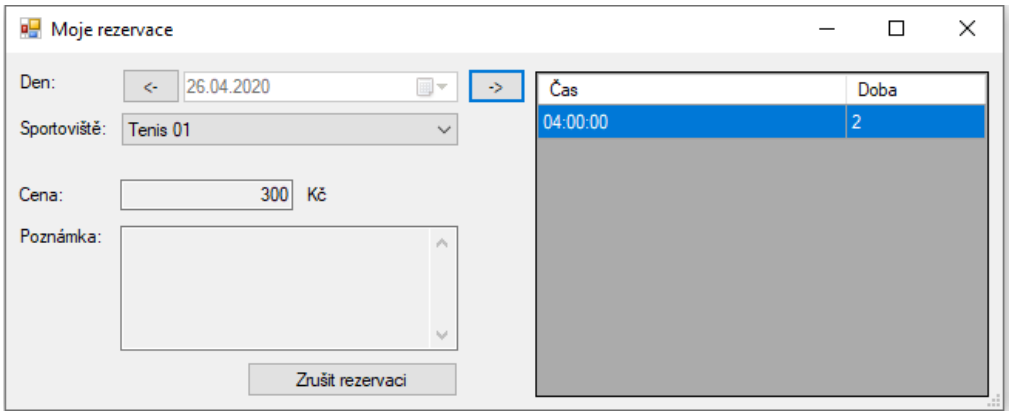
Zde je možnost vytvořit novou rezervaci. Uživatel také vidí již vytvořené rezervace na daný čas, datum a sportoviště (nevidí však jména uživatelů) a v případě kolize času a data ho systém upozorní. Je možnost vyplnit poznámku a také se zobrazuje cena za danou rezervaci.



Obrázek 28 – Rezervace

5.4.5 Moje rezervace

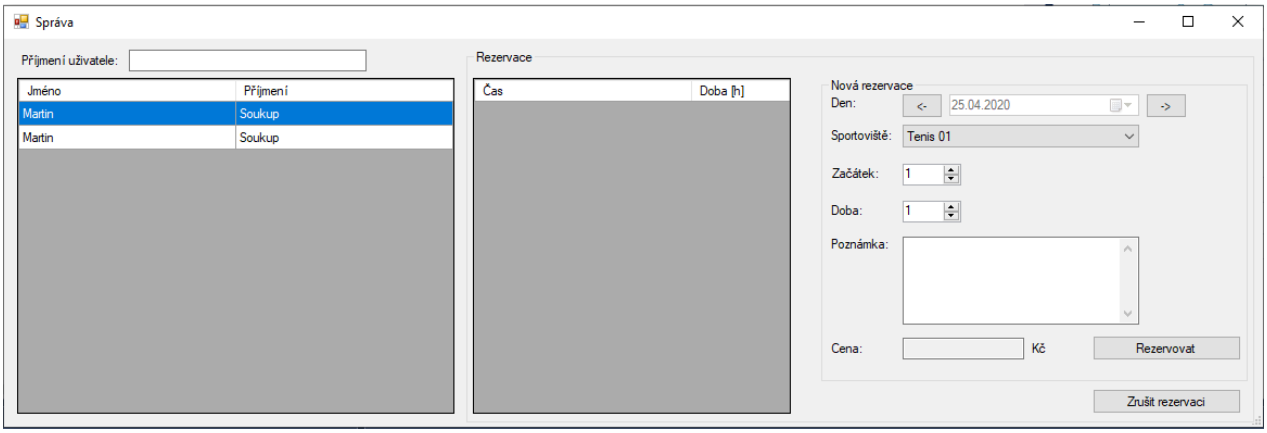
Přehled vlastních rezervací obsahuje ovládací prvky, jako formulář rezervací. Výběr data a sportoviště zůstává, odpadá možnost vytvoření nové rezervace. Rezervaci lze pouze zrušit, ale pouze v případě, že ještě nenabyla platnosti.



Obrázek 29 – Přehled rezervací

5.4.6 Správa rezervací

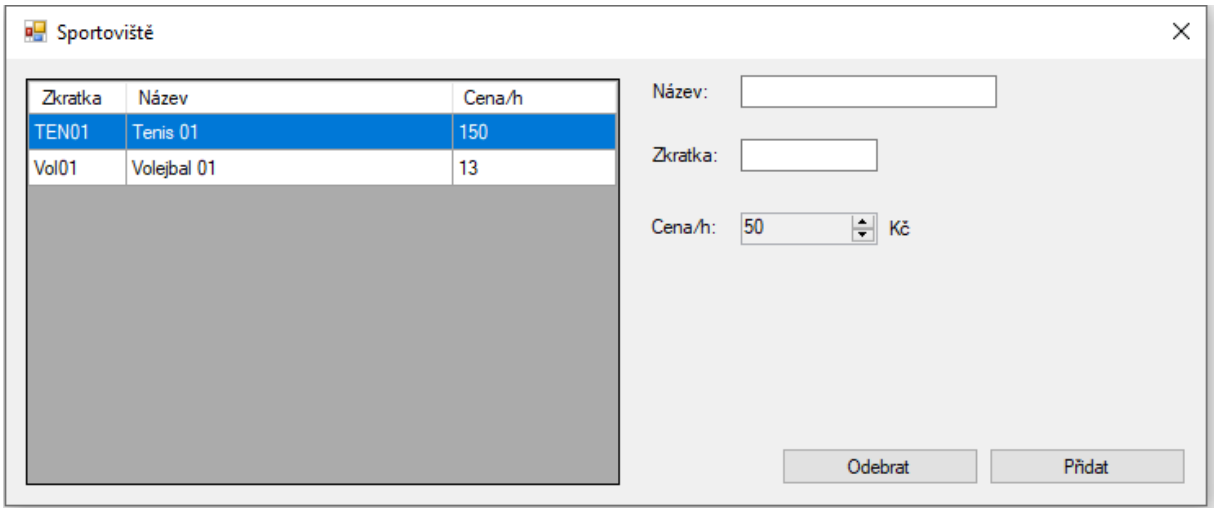
Zde mají správci možnost vytvářet a rušit registrace pro všechny uživatele. Formulář obsahuje dvě tabulky DataGridView, přičemž uživatelé se dají filtrovat dle příjmení a rezervace uživatelů se filtrují dle data a sportoviště. Dále formulář obsahuje nutné ovládací prvky pro zadání hodnot rezervace.



Obrázek 30 – Správa rezervací

5.4.7 Správa sportovišť

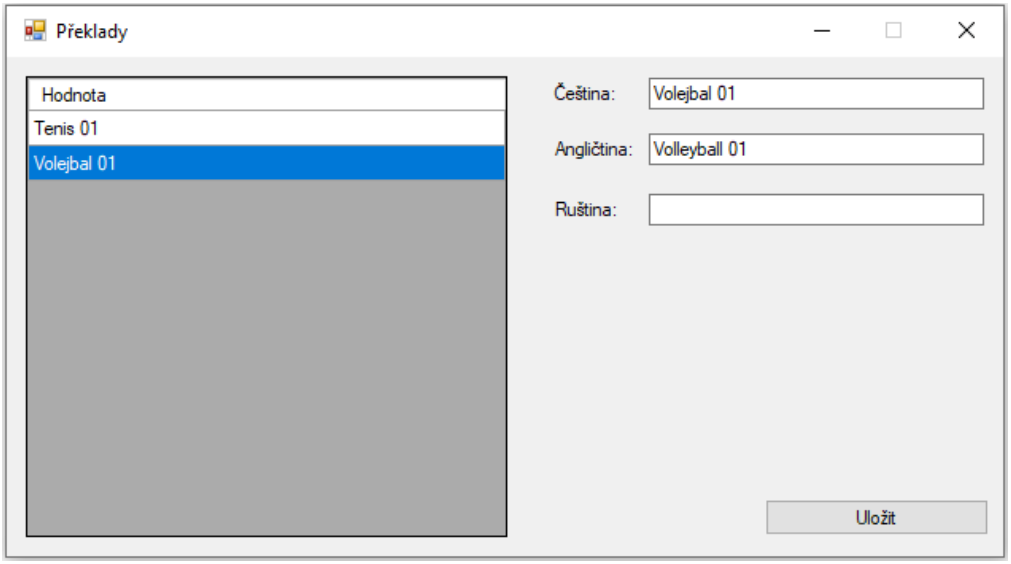
Na tomto formuláři je možnost pro administrátory přidávat či odebírat jednotlivá sportoviště. Opět se zde nachází ovládací prvky nutné k zadání potřebných dat a tabulka s přehledem sportovišť.



Obrázek 31 – Správa sportovišť

5.4.8 Správa překladů sportovišť

Tento formulář nabízí možnost spravovat jednotlivé překlady sportovišť. V tabulce hodnot jsou zobrazeny názvy sportovišť tak, jak byly prvně zapsány do systému. Následně pro každé sportoviště je možné vytvořit překlad. Pokud bude některé pole prázdné, daný překlad nebude zapsán.



Obrázek 32 – Správa překladů sportovišť

ZÁVĚR

Jak bylo zmíněno v úvodu, výsledkem bakalářské práce je poskytnutí základních znalostí o dané problematice a vysvětlení postupů při internacionalizaci, a to hlavně pro desktopové aplikace psané v programovacím jazyce C#.

Ukázky byly psány a testovány na operačním systému Microsoft Windows 10 a na .NET Frameworku verze 4.8. Z tohoto důvodu nemusí na starších verzích plně fungovat.

Potenciálním rozšířením bakalářské práce by bylo prohloubení znalostní v oblasti mobilních a webových aplikací. S tím samozřejmě by byly ukázány i nové postupy na těchto platformách.

Toto téma jsem si vybral, protože s touto problematikou jsem se ve své relativně krátké praxi ještě nesetkal. Tudíž i tvorba práce samotná mě naučila mnoho nového a obohatila mé znalosti. Po dokončení jsem si jistý, že jsem schopen vytvořit aplikaci, která je určená pro více kultur.

POUŽITÁ LITERATURA

[1] *What is internationalization?* [online]. 2016 [cit. 2020-02-25]. Dostupné z: https://www.technitrad.com/what-is-internationalization/

[2] ELHADY, Hady. *The Ultimate Guide To Mobile App Internationalization* [online]. [cit. 2020-03-04]. Dostupné z: https://instabug.com/blog/mobile-app-internationalization/

[3] ZIMMERMANN, Kim. *What Is Culture?* [online]. 2017 [cit. 2020-03-23]. Dostupné z: https://www.livescience.com/21478-what-is-culture-definition-of-culture.html

[4] *Nation* [online]. 16. 3. 2020 [cit. 2020-03-23]. Dostupné z: https://en.wikipedia.org/wiki/Nation

[5] FERRIER, Smith. .NET Internationalization: The Developer's Guide to Building Global Windows and Web Applications. August 07, 2006. United States: Addison Wesley Professional, 2006. ISBN 978-0-321-34138-9.

[6] Microsoft Visual Studio. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 3. 7. 2020 [cit. 2020-07-24]. Dostupné z: https://cs.wikipedia.org/wiki/Microsoft_Visual_Studio

[7] C Sharp. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 13. 4. 2020 [cit. 2020-07-24]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp

[8] CultureInfo Class [online]. United States [cit. 2020-07-24]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.globalization.cultureinfo?view=netcore-3.1

[9] The CultureInfo class. C# Tutorial [online]. 2007-2020 [cit. 2020-07-29]. Dostupné z: https://csharp.net-tutorials.com/working-with-culture-and-regions/the-cultureinfo-class/

[10] ELDAN, Lior. App Localization and Internationalization – Tips & Guide [online]. 2.9.19 [cit. 2020-08-05]. Dostupné z: https://www.moburst.com/app-store-optimization/localization/

[11] DEENANAUTH, Sara. Localization: Building Websites for a Global Audience [online]. Mar 21, 2018 [cit. 2020-08-08]. Dostupné z: https://medium.com/devtopia/localization-building-websites-for-a-global-audience-6f53d31ce737

[12] MySQL. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 6 August 2020 [cit. 2020-08-10]. Dostupné z: https://en.wikipedia.org/wiki/MySQL#Features