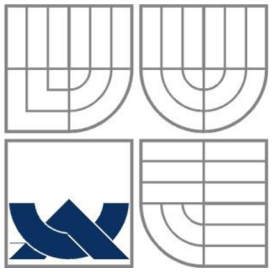
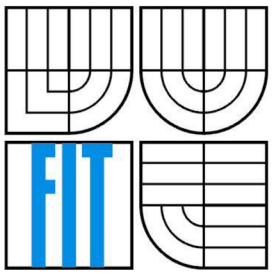


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO INTEGRACI PLATEBNÍCH TERMINÁLŮ

SYSTÉM FOR INTEGRATION OF PAYMENT TERMINALS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jakub Randa

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Pavel Očenášek, Ph.D.

BRNO 2014

Abstrakt

Tato diplomová práce se zabývá problematikou integrace elektronických platebních terminálů do jiných zařízení a umožnění platebních transakcí z těchto zařízení. Analyzuje aktuální situaci na trhu a možnosti vlastní implementace takového zařízení. Diskutuje bezpečnost platebních operací a platebních terminálů a legislativní situaci okolo elektronických plateb v České republice. Popisuje postup implementace aplikace pro integraci platebních terminálů a analyzuje výsledky jejího testování.

Abstract

This master's thesis describes the process of integration of electronic payment terminals into another technological solutions and eventually enabling the opportunity to make electronic payments using these solutions. It analyses the current situation on the field of electronic payments and the possibilities of own solution for these payments. It discusses the security aspects of payment operations and terminals and it also describes current legal situation involving the electronic payments in Czech Republic. It describes the process of implementation of the application for payment terminal integration and analyses the results of testing.

Klíčová slova

Elektronické platby, platební terminál, platební karta, bezpečnost

Keywords

Electronic payments, payment terminal, payment card, security

Citace

Randa Jakub: Systém pro integraci platebních terminálů, diplomová práce, Brno, FIT VUT v Brně, 2014

Systém pro integraci platebních terminálů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jakub Randa
27.5.2014

Poděkování

Děkuji panu Ing. Pavlovi Očenáškovi, Ph.D. za odborné vedení, projevenou ochotu a trpělivost a poskytnutí cenných rad a připomínek v průběhu tvorby této práce.

Obsah

1 Úvod

Elektronická platba prostřednictvím platebních karet a vlastních zákaznických karet získává v posledních letech prudce na popularitě. Ať už kvůli komfortu takové platby v porovnání s platbou v hotovosti a s ní spojenou nepříjemností mít u sebe značné množství různých finančních prostředků, větší ochraně proti odcizení nebo třeba možnosti platit v internetových obchodech z pohodlí svého domova, čím dál tím větší množství lidí dává přednost této formě platby oproti jiným způsobům. Co se však týče platby v kamenných obchodech, přítomnost platebního terminálu v současnosti stále nebývá pravidlem, zejména u menších prodejců. Obdobná situace nastává i u různých automatů, které platbu prostřednictvím platebních karet umožňují pouze zřídka.

Hlavním důvodem této absence je především cena takového zařízení. Jednou složkou této ceny jsou poplatky bankám za jednotlivé transakce. Tyto poplatky jsou dané každou bankou která tyto služby poskytuje a záleží na konkrétní smlouvě mezi bankou a zákazníkem. Cena za tyto služby nelze mimo rámec smlouvy ovlivnit, nicméně druhá část provozních nákladů již ovlivnit jde – jedná se o cenu za pořízení nebo pronájem platebního terminálu.

Podobný problém nastává i u platebních systémů založených na vlastních kartách a na ně napojevených vnitrofiremních systémů. Tyto systémy většinou pracují na kreditovém systému, kdy si zákazník tyto kredity zajistí platbou předem a po nákupu mu budou z jeho účtu odečteny. Typicky se tyto systémy využívají v prostředí, kde je vyžadováno co nejrychlejší odbavení zákazníka, např. hromadné stravovací provozy. Takové systémy nejsou napojeny na bankovní instituce, takže odpadá jedna ze složek zvyšující cenu za pořízení a provoz systému, avšak u vlastní hardwarové a programové výbavy je stále prostor pro inovace a snižování pořizovacích nákladů.

Účelem této práce je studie a realizace integrace takového terminálu do produktů firmy Ki-Wi Digital, s.r.o., kde jedním z hlavních faktorů kromě funkčnosti je i minimalizace ceny takového zařízení. Samotné elektronické platby pak budou probíhat přes Komerční banku, a.s., v jejímž projektu pro propagaci platby pomocí NFC karet je tento projekt řešen. Cílem práce je tedy vytvoření a integrace hardwarové a softwarové platformy, která bude schopna přijímat a zpracovávat elektronické platby prováděné prostřednictvím platebních karet.

V kapitole Platební terminály a elektronické platby rozeberu teoretické vlastnosti elektronických transakcí, hardwarové požadavky na platební terminál, platební karty a specifika bezdotykových platebních karet NFC.

Kapitola Rozhraní a bezpečnost platebních transakcí rozebírá bezpečnostní stránku věci. Je zde popsán postup provedení platební transakce a protokoly které jsou k této činnosti využity. V závěru kapitoly je pak popsáno programové rozhraní pro komunikaci mezi platebním terminálem a platební kartou, potažmo platebním systémem poskytovaném bankou.

Kapitola Specifikace požadavků obsahuje souhrn vlastností, které musí výsledná aplikace splňovat pro účely jejího nasazení do ostrého provozu. Dále popisuje hardwarové vlastnosti zařízení, na kterých bude aplikace provozována, a také specifikace čtečky platebních karet, která byla vybrána pro použití v aplikaci.

V kapitole Návrh řešení jsou zobrazeny návrhové UML diagramy, které vysvětlují podstatu problému a princip jeho řešení. Je zde také diskutována možnost rozšíření aplikace na mobilní telefony.

Následuje kapitola Použité technologie, ve které postupně podrobněji rozebírám technologie, prostředky a programy, jež jsem využil pro návrh a vývoj mého řešení. Zdůvodňuji zde, proč jsem se rozhodl zrovna pro tyto technologie a jaké jsou jejich výhody (případně nevýhody) oproti ostatním použitelným technologiím. Zároveň uvádím licence, pod kterými technologie u kterých to je relevantní využívám.

Kapitola Implementace knihovny představuje hlavní vlastní přínos práce. Popisuje způsob a přístup, s jakým jsem provedl samotnou implementaci knihovny pro integraci platebních terminálů. Rozebírá požadavek rozšířitelnosti této knihovny, představuje funkčnost knihovny a využité třídy, kterými byla tato funkčnost dosáhnuta. Dále popisuje doprovodné a podpůrné funkce nad rámec zadání, které jsem do knihovny implementoval na základě požadavků zákazníku společnosti Ki-Wi Digital, s.r.o. Popisuje postup testování knihovny a porovnává ji s ostatními systémy pro provoz platebních terminálů.

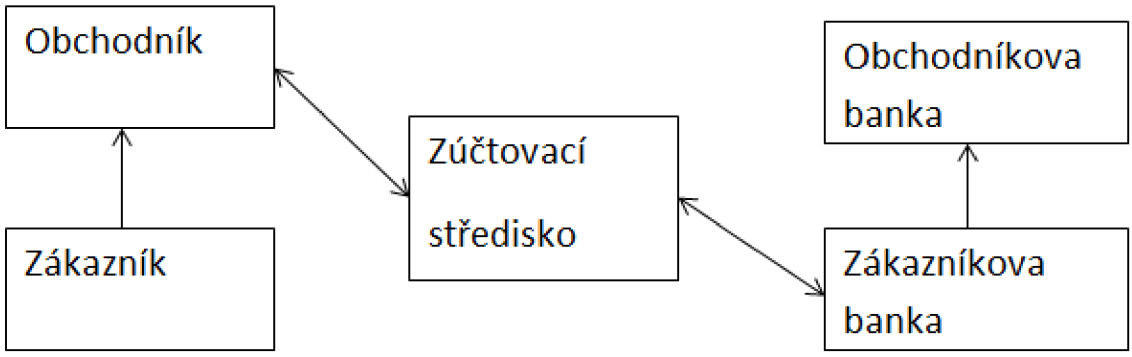
Kapitola Implementace vzorových aplikací představuje 3 realizované a k práci přiložené ukázkové aplikace, které jsem v rámci své diplomové práce vyvinul. Jedná se o ukázkovou demo aplikaci, která demonstruje základní funkčnost knihovny a jejich doprovodných funkcí, dále o hotový ale nerealizovaný projekt informačního kiosku s podporou platebních karet pro město Tábor a na závěr pak o interaktivní aplikaci s podporou klientských NFC karet pro společnost Tescoma, s.r.o., která byla nasazena do ostrého provozu.

Poslední kapitola Závěr pak shrnuje vlastnosti vytvořené knihovny a aplikací. Rozebírá jednotlivé body zadání a zmiňuje se také o vlastnostech, které byly implementovány nad rámec těchto bodů. Diskutuje možná rozšíření knihovny a jejího využití, a to jak u stávajících aplikací, tak u nově vytvářených.

2 Platební terminály a elektronické platby

2.1 Elektronická platební transakce

Transakce elektronické platby platební kartou se řídí podle základního platebního modelu [1]. V tomto modelu figuruje zákazník, který si kupuje určité zboží a který je vlastníkem platební karty vázané na bankovní účet. Oproti němu stojí obchodník který mu toto zboží prodává a který vlastní platební terminál schopný provádět elektronické platební transakce. Obchodník spravuje vlastní bankovní účet a dále obstarává komunikaci s poskytovatelem platebního terminálu (zúčtovacím střediskem) – tento poskytoval nemusí být nezbytně nutně shodný s poskytovatelem jeho bankovního účtu. Platební terminál u obchodníka po zahájení transakce komunikuje se zúčtovací bankou, která tento platební terminál provozuje. Tato banka kontaktuje finanční instituci, u které má zákazník vydanou platební kartu a ověří informace o této kartě a dále zůstatek na zákazníkově účtu, případně možnost přečerpání zůstatku. Po ověření těchto informací kontaktuje zúčtovací středisko, které platbu potvrdí a odešle toto potvrzení na obchodníkův platební terminál, zároveň zadá bance zákazníka platební operaci. Zákazníkova banka poté zablokuje z jeho účtu požadovanou částku a po určité ochrané lhůtě provede převod této částky na účet obchodníka. Tento model je znázorněn na obrázku č.1.



Obrázek č.1: Platební transakce

Důležitým pojmem v oblasti elektronické platby jsou tzv. „elektronické peníze“. Jedná se o virtualizaci finančních prostředků kterými disponuje držitel platební karty a které splňují požadavky potřebné k jejich použití v elektronických platebních transakcích. Těmito požadavky jsou [2]:

- Zpracovatelnost – elektronické peníze jsou plně zpracovatelné adekvátním zařízením (platební terminál).
- Přenositelnost – možnost přenosu elektronických peněz z média na jiné médium (typicky z čipové karty na účet obchodníka prostřednictvím platebního terminálu, který využívá počítač a síťovou komunikaci).
- Dělitelnost – možnost zaplacení libovolné částky až do výše disponovaného zůstatku.
- Decentralizovatelnost – možnost provádění plateb bez nutnosti kontroly jednou globální pevně danou institucí.
- Anonymita – nemožnost asociovat platbu s konkrétní osobou.
- Bezpečnost – ochrana systému proti padělání a falšování platebních operací.
- Monitorovatelnost – možnost monitorovat a vystopovat případné zneužití platebních systému, a to při splnění požadavku na anonymitu samotné platby. Jedná se zejména o vizuální kontrolu prováděných plateb a zařízení které se těchto plateb účastní.

2.2 Platební terminál

Terminál pro čipové karty (Smart Card Terminal) je samostatné hardwarové zařízení schopné zpracovávat komunikaci mezi čipovou kartou a okolním světem [2]. V této práci se zaměřím na terminály, které komunikují s platebními kartami – tedy platební terminály. Tento terminál obsahuje hardwarovou výbavu potřebnou pro bezpečný přístup k platební kartě a k informacím na této kartě, dále mechanickou numerickou klávesnici (či jiné zařízení schopné tuto klávesnici simulovat – například dotyková obrazovka a softwarové řešení) pro zadání PIN (Personal Identification Number) čísla karty, displej zobrazující průběh a údaje o platbě a volitelně pak i další součásti, například tiskárnu pro vytisknutí daňového dokladu. Tento systém je připojen k řídícímu počítači, a to většinou přes USB sběrnici, sériové rozhraní nebo přes síťové rozhraní (ethernet nebo wifi). Řídící počítač pak obstarává komunikaci se zprostředkovatelem platebního systému.

Současné terminály obsahují vlastní procesor a jsou zpravidla uzavřeně naprogramovány výrobcem zařízení. Terminály obsahují i vlastní úložné zařízení, a to většinou na bázi přepisovatelných pamětí typu EEPROM nebo pamětí typu RAM s bateriovým zdrojem. Terminály se dále dělí na tzv. online a offline terminály. Online terminály ke své činnosti vyžadují neustálý a nepřerušovaný přístup k řídícímu počítači a skrze něj ke zprostředkovateli platby, veškeré úkony spojené s platbou jsou prováděny okamžitě a nemohou být úspěšně splněny pokud spolu všechny strany transakce nekomunikují v reálném čase. Offline terminály jsou narozdíl od online terminálů schopné komunikovat se zprostředkovatelem samy, tato komunikace je však značně omezena a většinou jsou i tyto terminály připojeny k řídícímu počítači kvůli potřebě sekundárních funkcí, jako například update software na terminálu. Tento typ terminálů se však využívá naprosto minimálně –

v naprosté většině případů se využívají online terminály, které případně mohou mít již přímo integrován řídící počítač.

2.3 Platební karta

Platební karta je fyzický token který je vydáván zákazníkovi bankovní institucí a umožňuje mu využívat elektronické peníze, kterými disponuje u této bankovní instituce [3]. Má podobu identifikačního tokenu standardizované velikosti vyrobeného z polyvinylchloridu. Na přední straně karty jsou vytisknuty údaje o této kartě a o jejím držiteli – číslo karty, její platnost, jméno a příjmení držitele. Na základě typu karty mohou a nemusí být tyto údaje embosovány – tedy vyrobeny technologií která tyto údaje plasticky vyvýší nad povrch karty a tím umožní jejich zaznamenání pomocí speciálního zařízení. Na zadní straně karty je pak uveden kontrolní kód, který je využíván zejména u internetových plateb, a dále je zde místo vyhrazené pro fyzický podpis držitele, bez kterého je karta považována za neplatnou.

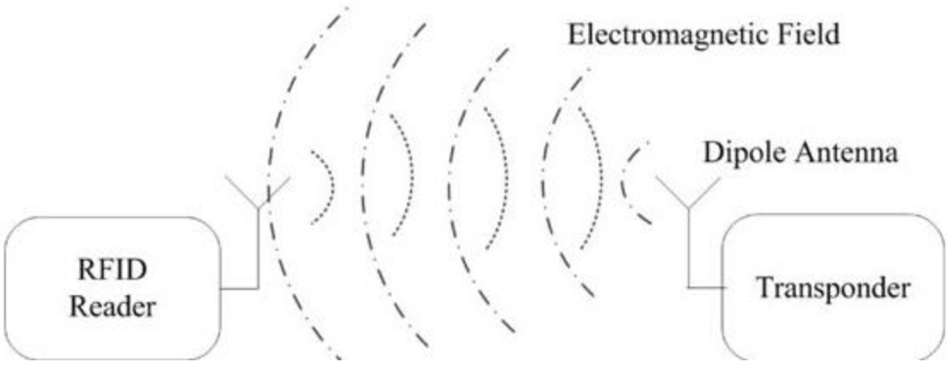
Karta dále disponuje magnetickým proužkem, který obsahuje informace o kartě v digitální podobě. Tyto informace lze strojově přečíst a zpracovat čtečkami těchto karet. Kódování dat a všobecné vlastnosti magnetických proužků jsou definovány ve standardu ISO 7811 [1]. Kapacita magnetických proužků je limitována na cca 1000 bitů [2]. Hlavní nevýhodou magnetických proužků je jejich relativně snadná zneužitelnost pomocí zařízení schopných tyto údaje číst a zapisovat.

Čipové karty obsahují mikrokontrolér, pomocí kterého mohou komunikovat s jiným zařízením přes sériové rozhraní a vzájemně se vůči sobě autorizovat. Míra bezpečnosti je v tomto případně podstatně vyšší než při využití magnetických proužků. Tato metoda také umožňuje uchovat několikanásobně větší množství informací – typicky 256kB a více. Kontakty čipového systému jsou vyvedeny na přední stranu karty na pevně stanovené místo. Čipové karty se dají dále dělit na paměťové karty a mikroprocesorové karty. Paměťové karty uchovávají informace v paměti typu EEPROM a přístup k nim je řízen zabezpečovací logikou, která určuje oblasti paměti s ochranou proti zapsání nebo smazání informací. Data jsou pak přenášena přes vstupně-výstupní port pomocí protokolu definovaném ve standardu ISO 7816 [2]. Mikroprocesorové karty představují komplexnější variantu čipových karet. Obsahují vlastní procesor (případně i koprocesor) a dále několik dalších funkčních prvků – paměť typu ROM, na které je uložen operační systém karty který není možné měnit, paměť typu EEPROM která slouží pro ukládání datových a programových informací a dále paměť typu RAM, která slouží jako pracovní paměť systému a je vymazána po odpojení karty od elektrického zdroje [2].

[1] Dostupné online na adrese
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=31433
[2] Dostupné online na adrese
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54089

2.3.1 Bezkontaktní platební karta

Bezkontaktní platební karty, někdy také označované jako NFC (Near Field Communication) karty, nevyužívají pro komunikaci s jiným zařízením kontakty vyvedené na přední straně (ačkoliv je z důvodu zpětné kompatibility většinou obsahují), ale - jak již název napovídá – přenos je v tomto případě řešen bezkontaktně pomocí technologie RFID. Jedná se nejenom o samotný datový přenos, ale také o generování elektrické energie potřebné pro funkčnost prvků čipové karty na základě indukce antény karty pomocí generovaného magnetického pole. Karta v tomto případě vystupuje jako transpondér, platební terminál pak jako základová stanice [4]. Spojení ve směru mezi základovou stanicí a transpondérem se nazývá „uplink“, v opačném směru pak „downlink“.



Obrázek č.2: Elektromagnetická indukce u NFC karty [3]

2.4 Legislativní situace

V České republice figuruje jako hlavní dozorce nad elektronickými transakcemi a elektronickými penězi Česká národní banka. Tato instituce řídí přidělování povolení na provozování elektronických platebních operací a elektronických peněž dalším finančním institucím (bankám), které pak tyto služby poskytují koncovým zákazníkům. Finanční instituce jsou poté povinny ve čtvrtletních intervalech informovat Českou národní banku o finančním pohybu v rámci povolení, které jim bylo přiděleno. Z hlediska koncového zákazníka který si zřizuje platební terminál působí Česká národní banka jako transparentní subjekt, veškerá jeho komunikace probíhá pouze s platební institucí která vlastní danou licenci.

Platební operace všeobecně (včetně těch elektronických) se v České republice řídí zákonem č. 284/2009 Sb. o platebním styku. Podmínky elektronických platebních operací a elektronických peněz navíc upravují vyhlášky České národní banky č. 141/2011 Sb. o výkonu činnosti platebních institucí, institucí elektronických peněz, poskytovatelů platebních služeb malého rozsahu a vydavatelů elektronických peněz malého rozsahu, a dále č. 142/2011 Sb. o předkládání informací platebními institucemi, institucemi elektronických peněz, poskytovateli platebních služeb malého

[3] Zdroj: http://i.stack.imgur.com/tliLG.png

rozsahu a vydavateli elektronických peněz malého rozsahu České národní bance. Všechny uvedené zákony a vyhlášky jsou k dispozici na zdroji [8].

3 Rozhraní a bezpečnost platebních transakcí

Rozhraní a standardy pro platební operace popisuje specifikace EMV. Tato zkratka obsahuje první písmena tří iniciátorů této specifikace – společností Europay, MasterCard a Visa [2]. První verze této specifikace vznikla již v říjnu roku 1994, na konci roku 2002 pak byla vytvořena čtvrtá verze, která se s různými úpravami používá dodnes. Aktuální verze této specifikace má označení 4.3 a pochází z listopadu roku 2011 [5]. Tato specifikace se skládá ze 4 knih:

1. Application Independent ICC to terminal interface requirements – tato kniha popisuje elektro-mechanické vlastnosti karet, jejich logické rozhraní a přenosové protokoly.
2. Security and key management – definice zabezpečení karet a jejich komunikace a dále systém práce s bezpečnostními klíči.
3. Application specification – příkazy a procesy pro provádění elektronických plateb na základě EMV specifikace.
4. Cardholder, attendant, and acquirer interface requirement – poslední kniha popisuje rozhraní pro platební terminály mezi zákazníkem a prodejcem.

3.1 Průběh platební transakce

Platební transakce na platebním terminálu probíhá podle specifikace EMV v těchto základních krocích a za pomocí těchto příkazů [2]:

1. Výběr aplikace
    - Reset
    - SELECT FILE (directory file)
    - READ RECORD (directory file)
    - SELECT FILE (DF of the EMV application)
2. Načtení směrodatných dat
    - GET PROCESSING OPTIONS
3. Statická asymetrická autentifikace
    - GET DATA (for unilateral authentication of the smart card by the terminal)
4. Ověření PIN
    - VERIFY
5. Rozhodnutí o platbě (online/offline)
    - GENERATE APPLICATION CRYPTOGRAM (for online authorization)

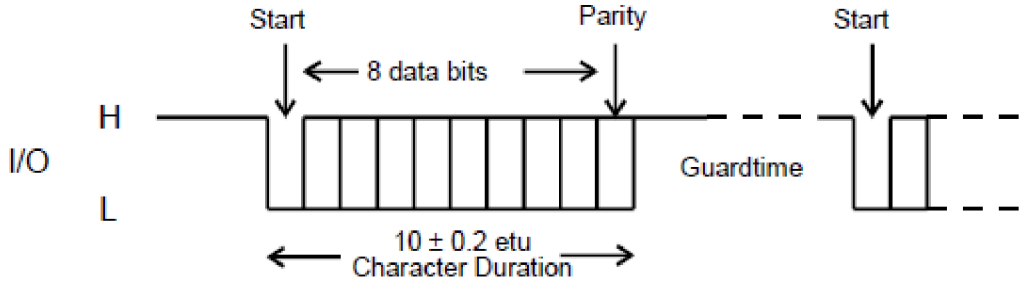
6. Dokončení platební transakce
    - GENERATE APPLICATION CRYPTOGRAM (for transaction certificate)

3.2 Přenosové protokoly

Pro přenos dat z platební čipové karty do terminálů definuje specifikace EMV protokoly, které jsou podrobně popsány v knize [5]. Specifikace umožňuje jak přenos po jednotlivých znacích (označované jako T=0), tak po blocích dat (T=1). Čipová karta musí umožňovat právě jeden z těchto způsobů přenosu, platební terminály musí podporovat oba typy přenosů. Přenosové protokoly pracují na čtyřech vrstvách.

3.2.1 Fyzická vrstva

Fyzická vrstva zajišťuje samotnou výměnu znakových rámců mezi zařízeními prostřednictvím přenosové linky. Specifikace této vrstvy je shodná pro oba typy přenosu (T=0 i T=1). Znak je odesílán v 10-bitovém rámci, před přenosem každého rámce se přenosová linka musí nacházet ve stavu H. Rámec je složen z 1 start bitu ve stavu L, následuje 8 datových bitů a 1 kontrolní paritní bit. Znázornění takového rámce je k vidění na obrázku č.3 [5].



Obrázek č.3: Rámec fyzické vrstvy specifikace EMV (převzato z [5])

3.2.2 Linková vrstva

Linková vrstva zajišťuje výměny znaků (u T=0) nebo bloků dat (u T=1). Každý z těchto způsobů je zajištěn svým vlastním a specifickým protokolem a každý z těchto protokolů má také vlastní způsob detekce a opravy chyb [5].

3.2.3 Transportní vrstva

Transportní vrstva obstarává výměnu příkazů a odpovědí mezi terminálem a čipovou kartou. Definice této výměny je opět specifická pro každý ze způsobů přenosu T=0 a T=1. Tyto zprávy jsou označovaný souhrně jako APDU (Application Protocol Data Unit). Terminál posílá na čipovou kartu příkazovou zprávu (C-APDU), opačným směrem je pak odeslána zpráva s odpovědí (R-APDU) [5].

3.2.4 Aplikační vrstva

Aplikační protokol se stará o generování a párování C-APDU a R-APDU zpráv. Ke každé C-APDU zprávě (příkaz) náleží právě jedna R-APDU zpráva (odpověď). Přítomnost dat v obou typech zpráv je však volitelná.

C-APDU zpráva obsahuje povinnou hlavičku složenou z čísla instrukční třídy (CLA), instrukčního kódu v rámci dané třídy (INS) a 2 referenčních bytů které doplňují instrukční kód (P1 a P2). Následuje nepovinné tělo zprávy, které se skládá z 1 bytu definující počet bytů, které budou odeslány v datové části C-APDU (Lc), řetezec datových bytů o délce definované v Lc (Data) a 1 byte definující maximální množství bytů očekávaných v odpovědi R-APDU (Le). Možné kombinace těchto údajů jsou k vidění na obrázku č.4.

Case	Structure
1	CLA INS P1 P2
2	CLA INS P1 P2 Le
3	CLA INS P1 P2 Lc Data
4	CLA INS P1 P2 Lc Data Le

Obrázek č.4: Možné formy příkazové zprávy CAPDU (převzato z [5])

R-APDU zpráva (odpověď) obsahuje nepovinný řetězec znaků o délce definované v Le bytu C-APDU zprávy a dále povinnou dvojici bytů SW1 a SW2, které signalizují stav čipové karty po provedení daného příkazu [5].

3.3 Zabezpečení platebních karet

3.3.1 Statická datová autentifikace

Statická datová autentifikace (Static Data Authentication – SDA) je proces ověření pravosti statických aplikačních dat uložených na kartě. Statická aplikační data obsahují neměnné informace o kartě, které ji jednoznačně idetifikují – zejména pak číslo karty, datum její platnosti, jméno uživatele kterému byla karta vystavena a její bezpečnostní kód CVC. Toto zabezpečení je schopné detekovat neautorizované změny dat na kartě provedené po jejím přidělení konkrétnímu uživatelovi. Proces verifikace probíhá přímo v terminálu, a to v režimu offline – terminál neprovádí žádnou komunikaci s jiným zařízením, vše je řešeno vnitřně na hardwarové úrovni. Verifikace se provádí na základě soukromých a veřejných klíčů a vyžaduje existenci certifikační autority, která tyto klíče poskytuje.

Samotná autentifikace pak probíhá ve třech hlavních krocích:

1. Terminál získá veřejný klíč certifikační autority.

2. Karta předá terminálů veřejný klíč vystavovatele karty a zabezpečená statická aplikační data karty (Signed Static Application Data).
3. Terminál ověří pravost zabezpečených statických aplikačních dat [6].

3.3.2 Dynamická datová autentifikace

Dynamická datová autentifikace (Offline Dynamic Data Authentication – ODDA) je proces ověřování pravosti dynamických dat uložených na kartě. Dynamická data obsahují informace, které se po vydání karty konkrétnímu uživatelovi mohou měnit. Autentifikace se provádí na terminálu pomocí systému klíčů a podobně jako u statické datové autentifikace probíhá offline – ověřuje tedy jenom neautorizovaný přepis dat bez nutnosti komunikace s jiným zařízením. I tento proces vyžaduje existenci certifikační autority.

Rozpoznáváme 2 typy této autentifikace:

- Dynamická datová autentifikace (Dynamic Data Authentification – DDA) – tato autentifikace je prováděna okamžitě po přiložení karty k terminálu. Karta vygeneruje digitální podpis pro její dynamická data a data přijatá z terminálu.
- Kombinovaná dynamická datová autentifikace (Combined Dynamic Data Authentication – CDA) – tato autentifikace je prováděna při prvním a druhém přijmutí požadavků na vygenerování příkazu. Karta v tomto případě opět generuje digitální podpis pro její dynamická data, který obsahuje transakční certifikát (Transaction Certificate – TC) nebo kryptogram pro autoriovaný požadavek (Authorisation Request Cryptogram – ARQC) a dále „nepředvídatelné číslo“ vygenerované příkazem GET CHALLENGE.

U obou typů autentifikace se postupuje na základě těchto čtyřech hlavních kroků:

1. Terminál získá veřejný klíč certifikační autority.
2. Karta předá terminálů veřejný klíč vystavovatele karty.
3. Karta předá terminálu vlastní veřejný klíč (ICC Public Key) a dynamická data k ověření.
4. Terminál ověří pravost předaných dynamických dat [6].

3.3.3 Ověření PIN kódu

Ověření PIN (Personal Identification Number) kódu je proces, který autentifikuje fyzického uživatele karty na základě znalosti tohoto kódu. Spočívá v zadání PIN kódu (typicky čtyřmístného) uživatelem do vstupního zařízení (hardwarová klávesnice, dotykový displej…) a porovnání zadaného kódu s kódem uloženým na kartě. Samotný kód na kartě pak spadá mezi dynamická data a řídí se podle jejich podmínek, při procesu ověřování je však navíc nutné zajistit zabezpečený přenos kódu z uživatelského vstupu do terminálu a ke kartě.

Platební karta poskytuje veřejný šifrovací klíč pro šifrování PIN kódu, který je poskytnut uživatelskému vstupnímu zařízení a použit k zašifrování PIN kódu. Samotná karta pak použije svůj soukromý šifrovací klíč pro dešifrování PIN kódu, porovná ho se svým uložebým kódem a s výsledkem porovnání obeznámí terminál [6].

3.3.4 Zabezpečené odesílání zpráv

Zabezpečené odesílání zpáv je proces, který je dodržován při komunikaci terminálu a vzdáleného zařízení. Při této komunikaci je potřeba pro odesílaná data splnit tyto podmínky:

- Datová integrita – zajištěna pomocí autentifikačního kódu zprávy (Message Authentication Code – MAC).
- Datová důvěrnost – zajištěna šifrováním zprávy, podporované šifrovací algoritmy jsou popsány v kapitole 3.3.5.
- Autentifikace odesílatele dat – zajištěna pomocí autentifikačního kódu zprávy MAC [6].

3.3.5 Podporované šifrovací algoritmy ve specifikaci EMV

Specifikace EMV umožňuje pro účely šifrování využití pouze podporovaných šifrovacích algoritmů. Pro šifrování PIN kódu navíc musí být použit pouze asymetrický šifrovací algoritmus.

Podporované algoritmy jsou tyto [6]:

- Symetrické algoritmy
  - Data Encryption Standard (DES - 8 bytový)
  - Advanced Encryption Standard (AES – 16 bytový)
- Asymetrické algoritmy
  - RSA algoritmus
- Hašovací algoritmy
  - Secure Hash Algorithm (SHA-1)

3.4 Platební rozhraní

Platební rozhraní je souhrn funkcí a procesů, pomocí kterých komunikuje platební terminál s platební kartou a s bankovní institucí. Komunikace probíhá prostřednictvím příkazů a odpovědí na tyto příkazy, konkrétně pak přes C-APDU a R-APDU zprávy, které jsou důkladněji popsány v kapitole 3.2.4.

3.4.1 Příkazy pro komunikaci s platební kartou

Rozhraní pro komunikaci mezi terminálem a platební kartou definuje tyto příkazy (v závorkách jsou uvedeny jejich C-APDU čísla instrukční třídy a čísla instrukčního kódu) [7]:

- APPLICATION BLOCK (třída *8C*, instrukční kód kód *18*) – tento příkaz lze použít pouze v případě, že karta již byla vydána koncovému uživatelovi a tento stav je v ní patřičně uložen. Příkaz zablokuje aktuálně zvolenou aplikaci pro použití v systému. Oba referenční byty instrukce je zapotřebí nastavit na *00*, datová složka zprávy pak musí obsahovat MAC kód. V odpovědi (R-APDU) se očekává kód *9000*, který signalizuje úspěšné zablokování aplikace bez ohledu na fakt, jestli již aplikace byla zablokována před použitím příkazu.
- APPLICATION UNBLOCK (třída *8C*, instrukční kód *18*) – tento příkaz obdobně jako v předchozím případě spadá do kategorie příkazů použitelných až po vydání karty uživatelovi. Příkaz zruší zablokování právě zvolené aplikace. Veškeré parametry zprávy včetně odpovědního kódu jsou shodné s popisem uvedeným v príkazu APPLICATION BLOCK.
- CARD BLOCK (třída *8C*, instrukční kód *16*) – i tento příkaz je možné použít až po vydání karty zákazníkovi. Příkaz zablokuje všechny aplikace dostupné na kartě. Referenční byty P1 a P2 této instrukce se nastavují na *00*, datová část zprávy obsahuje zabezpečený MAC kód. Návratový kód *9000* značí úspěšné zablokování karty.
- EXTERNAL AUTHENTICATE (třída *00*, instrukční kód *82*) – tento příkaz vyšle na karetní aplikaci požadavek k ověření kryptogramu vystavovatele platební karty. Oba referenční byty instrukce se nastavují na hodnotu *00*, datová složka se skládá z povinného 8-bytového kryptogramu k ověření a z dalších nepovinných 1-8 bytů vyhrazených pro aplikační účely. Odpověď obsahuje kód *9000* v případě úspěšného provedené příkazu, nebo kód *6300* v případě neúspěšné autentifikace.
- GENERATE APPLICATION CRYPTOGRAM (třída *80*, instrukční kód *AE*) – příkaz odešle na kartu data týkající se zpracovávané transakce a karta na jejich základě vypočítá a vrátí kryptogram. Vygenerovaný kryptogram je právě jedním ze tří možných typů na základě výsledku příkazu – *Transaction Certificate (TC)* v případě schválení transakce, *Application Authentication Cryptogram (AAC)* v případě zamítnutí transakce, nebo *Authorisation Request Cryptogram (ARQC)* v případě dodatečné nutnosti online autorizace. Příkaz j e dále možné vytvořit pro vygenerování kryptogramu pro kombinovanou dynamickou datovou autentifikaci (CDA). Referenční byte P1 signalizuje typ požadovaného kryptogramu (viz obrázek č.5), referenční byte P2 se nastavuje na hodnotu *00*. Datová část zprávy obsahuje informace o transakci a její délka se různí na základě typu transakce a aplikačních specifikací. Odpověd

na tento příkaz pak obsahuje kód *9000* označující úspěšné vykonání příkazu a ve své datové části obsahuje vygenerovaný kryptogram. Tento kryptogram nemusí být nezbytně nutně stejného typu, jaký byl vyžadován v příkazové části zprávy – typickým případem je očekávaný typ kryptogramu TC, který však nemohl být vrácen z důvodu zamítnutí transakce a místo něj byl tedy odeslán kryptogram AAC.

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0							AAC
0	1							TC
1	0							ARQC
1	1							RFU
		x						RFU
			0					CDA signature not requested
			1					CDA signature requested
				x	x	x	x	RFU

Obrázek č.5: Nastavení referenčního bytu P1 v příkazu GENERATE APPLICATION CRYPTOGRAM (převzato z [7])

- GET CHALLENGE (třída *00*, instrukční kód *84*) – tento příkaz žádá kartu o vygenerování „nepředvídatelného čísla“, které se využívá v zabezpečovacích procedurách transakce (například ODDA, viz kapitola 3.3.2). Oba referenční byty jsou nastaveny na hodnotu *00* a příkaz neobsahuje žádnou datovou složku. V odpovědi na příkaz se nachází kód *9000* značící úspěšné provedené příkazu a v datové části odpovědi nalezneme vygenerované 8-bytové číslo.
- GET DATA (třída *80*, instrukční kód *CA*) – tento příkaz žádá kartu o vrácení primitivního datového objektu, který se týká informací na kartě které nemají přímou souvislost s právě prováděnou aplikací. Jedná se o 4 datové typy, které se specifikují hodnotou referenčních bytů P1 a P2:
    - 9F36 – informace o počtu provedených transakcí (Application Transaction Counter)
    - 9F13 – časové razítko poslední provedené transakce (Last Online ATC Register)
    - 9F17 – počítadlo počtu zbývajících pokusů o zadání PIN kódu (PIN Try Counter)
    - 9F4F – logovací formát podporovaný kartou (Log Format)

Tento příkaz neobsahuje žádnou datovou složku. V odpovědi je očekáván kód *9000* signalizující úspěšné provedení příkazu a v datové části pak navrácený primitvní datový typ který byl požadován v příkazu.

- GET PROCESSING OPTIONS (třída *80*, instrukční kód *A8*) – tento příkaz se používá k inicializaci transakce v rámci karty. Referenční byty instrukce jsou nastaveny na hodnotu *00*, v datové části zprávy je zakódován datový typ *Processing Options Data Object List (PDOL)*, který obsahuje dotazy na funkční možnosti karty. Návratová odpověď opět obsahuje kód *9000* v případě úspěšného provedené příkazu a dále vrací ve své datové části hodnoty *Application Interchange Profile (AIP)* a *Application File Locator (AFL)* nutné pro provedené transakce.
- INTERNAL AUTHENTICATE (třída *00*, instrukční kód *88*) – příkaz na kartě zahájí výpočet zabezpečených dynamických aplikačních dat a v případě úspěchu tato data vrátí. Pro výpočet se využívá „nepředvídatelné číslo“ které je odeslané spolu se správou, dále data na samotné kartě a privátní klíč, který je také uložený na kartě. Oba referenční byty instrukce jsou nastaveny na hodnotu *00*, v datové části je odeslanáno vygenerované číslo a případně další aplikačně-specifické informace. V případě úspěšného provedené příkazu je navrácen kód *9000* a vygenerovaná zabezpečená dynamická aplikační data.
- PIN CHANGE/UNBLOCK (třída *8C*, instrukční kód *24*) – tento příkaz je možné použít pouze po vydání karty cílovému uživatelovi. Umožňuje odblokovat kartu, nebo souběžně změnit PIN a odblokovat kartu. V případě úspěchu je počítadlo zbývajícího počtu pokusů o zadání PIN kódu resetováno na hodnotu limitu počtu zadání a pokud to bylo vyžadováno, tak je změněn PIN kód karty. Hodnota referenčního bytu P1 je nastavena na *00*, referenční byte P2 je nastaven na jednu z hodnot *00*, *01* nebo *02*. Hodnota *00* značí žádost o odblokování karty, hodnoty *01* a *02* jsou aplikačně specifické pro konkrétní platební systémy. Datová část zprávy obsahuje zabezpečovací MAC kód a dále PIN kód v případě požadavku na jeho změnu. Odpověd na příkaz obsahuje kód *9000* v případě úspěchu.
- READ RECORD *(*třída *00*, instrukční kód *B2*) – příkaz přečte a vrátí datový záznam v lineárním souboru uloženém na kartě. Referenční byte P1 je nastaven na číslo požadovaného záznamu, byte P2 pak kontrolu reference záznamu podle tabulky na obrázku č.6. Odpovědí je kód *9000* v případě úspěšného vykonání příkazu a požadovaná hodnota v datové části odpovědi.

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x				SFI
					1	0	0	P1 is a record number

Obrázek č.6: Nastavení referenčního bytu P2 v příkazu READ RECORD (převzato z [7])

- VERIFY (třída 00, instrukční kód 20) – tento příkaz je využíván pro offline kontrolu zadaného PIN kódu. Referenční byte P1 je nastaven na hodnotu *00*, referenční byte P2 je pak nastaven na hodnotu uvedenou v tabulce na obrázku č.7 podle způsobu formátování PIN kódu v datové části příkazu. Odpověď obsahuje kód *9000* v případě úspěšného provedené příkazu, v datové části je pak v případě neúspěšného porovnání předaného a uloženého PIN kódu navrácen počet zbývajících možných opakovaných zadání PIN kódu.

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	0	0	0	0	As defined in ISO/IEC 7816-4 [3]
1	0	0	0	0	0	0	0	Plaintext PIN, format as defined below
1	0	0	0	0	x	x	x	RFU for this specification
1	0	0	0	1	0	0	0	Enciphered PIN, format as defined in Book 2
1	0	0	0	1	0	x	x	RFU for this specification
1	0	0	0	1	1	x	x	RFU for the individual payment systems
1	0	0	1	x	x	x	x	RFU for the issuer

Obrázek č.7: Nastavení referenčního bytu P2 v příkazu VERIFY (převzato z [7])

4 Specifikace požadavků

Výsledkem této práce by měl být produkt, který jak po softwarové tak po hardwarové stránce umožní uživatelům provádět platby pomocí platebních či zákaznických karet. Na diplomové práci spolupracuji s firmou Ki-Wi Digital, s.r.o., s jejíž pomocí bude výsledný produkt nasazen do ostrého provozu. Firma Ki-Wi Digital, s.r.o. dále na tomto projektu spolupracuje s finanční institucí Komerční banka, a.s. v rámci jejího projektu na propagaci plateb pomocí bezkontaktních NFC platebních karet. Aplikace bude též umožňovat jednoduché ukládání základních informací o transakcích za účelem logování.

V první fázi půjde právě o integraci NFC platebních terminálů do produktů firmy Ki-Wi Digital, s.r.o., aplikace však bude navržena i pro použití s klasickými kontaktními kartami. Výsledný produkt se bude integrovat jak do již existujících zařízení (zejména informační kiosky), tak do nových produktů firmy Ki-Wi Digital, s.r.o., např. interaktivní tablety.

4.1 Specifikace požadavků na aplikaci

Aplikace musí umět zprostředkovat platební operace pomocí kontaktních a bezkontaktních platebních a zákaznických karet. Bude se jednat o knihovnu, která se poté bude využívat v různých softwarových řešeních firmy Ki-Wi Digital, s.r.o. Knihovna tedy musí poskytovat patřičné rozhraní, přes které budou moci ostatní programy s touto knihovnou komunikovat. Aplikace musí splňovat požadavky standardu EMV a musí mít k dispozici přípravu pro implementaci na připojení k bankovnímu systému finančních institucí.

Aplikace bude založena na technologii Microsoft .NET ve verzi 4.0. Programovacím jazykem bude jazyk C# a grafické rozhraní bude zajišťovat technologie Windows Presentation Foundation (WPF). Aplikace bude vyvíjená pro operační systémy Microsoft Windows 7 Embeded a Microsoft Windows 8 Embeded. Využití vlastních minimálních instalačních obrazů Embeded verzí těchto operačních systémů bude dosaženo snížení prostorových a výpočetních nároků na řídící počítače a tedy možnost provozování aplikace na levnějším hardwaru, což je důležitý faktor, se kterým se musí při návrhu aplikace počítat. Tyto požadavky plynou z nutnosti integrace do již existujících produktů firmy Ki-Wi Digital, s.r.o. Zadávání uživatelského vstupu bude řešeno aplikačně pomocí dotykových obrazovek.

4.2 Hardwarová platforma

Aplikace se bude integrovat na již existující informační kiosky, které obsahují řídící počítač Lenovo IdeaCentre Q180, a dále na tablety Acer Iconia Tab W510. Obě zařízení podporují operační systémy

Microsoft Windows a na každém z nich poběží jiný softwarový produkt, který bude brát zřetel na specifická výkonostní kritéria. Systém pro integraci platebních operací musí být schopen běhu na obou počítačích, vzhledem k absenci grafického rozhraní či jiného výpočetně náročného procesu však nebude pro tato zařízení výkonově limitující.

Průmyslové mini počítače IdeaCentre Q180 z produkce firmy Lenovo umožňují díky své velikosti instalaci například připevněním k VESA držáku zobrazovacích zařízení. Pro účely aplikací firmy Ki-Wi Digital, s.r.o. jsou na ně instalovány operační systémy Microsoft Windows 7 Embeded. Z důležitého hardwaru pro účely této práce osahují tyto komponenty (kompletní hardwarová specifikace k dispozici na [4]):

- Procesor Intel Atom D2700 (2 jádra, 4 vlákna, frekvence 2,13 GHz),
- paměť 2GB DDR3 1333MHz,
- grafická karta AMD Radeon HD 6450A, 512 MB RAM.

Obrázek č.8: Lenovo IdeaCentre Q180 [5]

Tablety Iconia Tab W510 od společnosti Acer jsou pro účely aplikací firmy Ki-Wi Digital, s.r.o. vybaveny operačním systémem Microsoft Windows 8 Embeded. Hardwarová specifikace těchto tabletů vypadá takto (kompletní hardwarová specifikace k dispozici na [6]):

[4] http://www.lenovo.com/shop/americas/content/pdf/system_data/q180_tech_specs.pdf
[5] Zdroj: http://www.lenovo.com/shop/americas/content/img_lib/prod-launch/530x430/eol-q180.jpg

- Procesor Intel Atom Z2760 (2 jádra, 4 vlákna, frekvence 1,50 GHz),
- paměť 2GB LPDDR2,
- grafická karta integrovaná na procesoru, 533 MHz.

Obrázek č.9: Acer Iconia Tab W510 [7]

Jako čtečka bezkontaktních NFC karet byla zvolena čtečka ACR122U USB NFC Reader od společnosti Advanced Card Systems Ltd. Toto zařízení se k počítači připojuje přes USB port, přes který je také napájeno. Kromě vlastního čtecího zařízení obsahuje také LED diodu signalizující aktuální stav zařízení. Je kompatibilní s desktopovými operačními systémy Microsoft Windows (včetně verzí 7 a 8), dále MacOS, Linux a Android 3.1 a vyšší.

[6] http://us.acer.com/ac/en/US/content/professional-model-datasheet/NT.L0SAA.002
[7] Zdroj: http://www.tabletblog.de/wp-content/uploads/2012/11/acer-iconia-tab-w510-tablet.jpg

Obrázek č.10: ACR122U USB NFC Reader [8]

[8] Zdroj: http://www.acs.com.hk/action.downloads.php?page=product-image-library&id=1744&type=20121229145436acr122u_side2.png

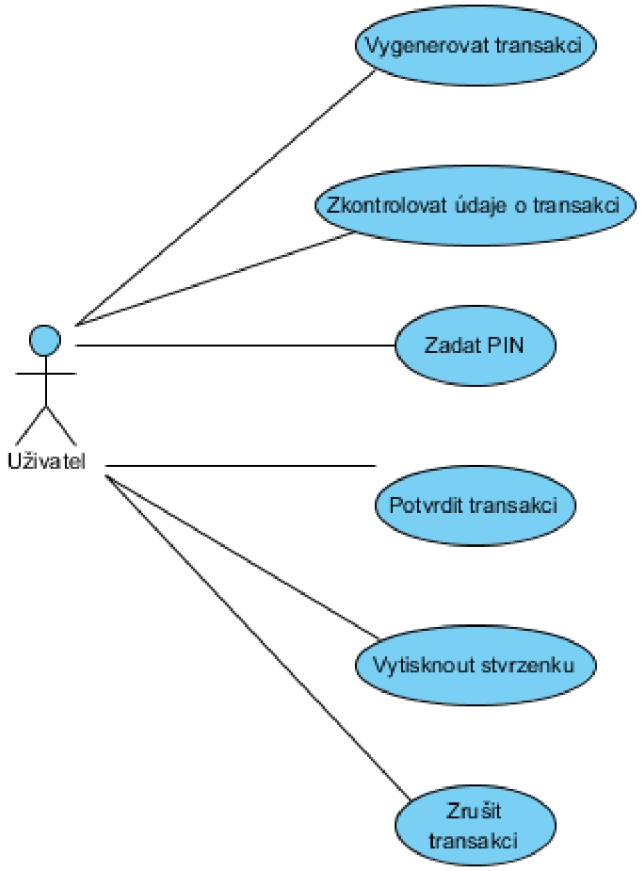
5 Návrh řešení

5.1 Návrh aplikace pro kiosky a tablety

Následující diagramy zobrazují návrh samotné knihovny pro zprostředkování platebních operací. Zapojení knihovny do softwarových řešení firmy Ki-Wi Digital, s.r.o. bude podrobněji rozebráno v kapitole Implementace.

5.1.1 Diagram případů užití

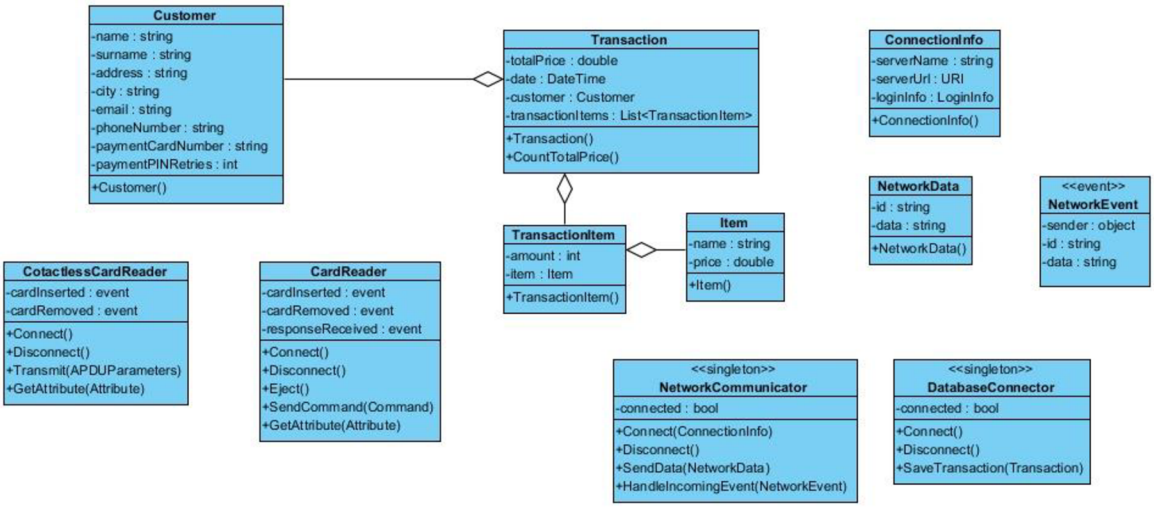
Diagram případů užití je zobrazen na obrázku č.11. Z podstaty anonymity elektronických plateb bude rozeznávat pouze jednu uživatelskou roli – uživatele který provádí platbu. Systém nemá žádnou administrátorskou část a v rámci zachování anonymity není schopen rozpoznat kdo ho využívá. Diagram zobrazuje případy užití v momentě předání řízení knihovně od navázaného programu, ve kterém si uživatel vybírá položky k zaplacení.



Obrázek č.11: Diagram případů užití

5.1.2 Diagram tříd

Návrhový diagram tříd je zobrazen na obrázku č. 12. Obsahuje třídu *Customer*, která uchovává informace o nakupujícím zákazníkovi – je to z důvodu možnosti uchování informací na přání prodejce, například z důvodu různých soutěží nebo promoakcí. Předpokládá se ale, že naprostá většina transakcí bude probíhat anonymně a tedy položky v této třídě zůstanou nevyplněny. Další důležitou třídou je třída *Transaction*, která obsahuje informace o jednotlivých transakcích a částce, která byla v rámci transakce zaplacena. Tyto informace jsou poté ukládány do jednoduché databáze, která slouží pro kontrolu hospodaření provozovatele zařízení. Možnost ukládání je opět volitelná. Dvojice tříd *CardReader* a *ContactlessCardReader* slouží pro komunikaci se čtečkami kontaktních, respektive bezkontaktních karet. Ačkoliv se na první pohled může zdát, že jejich činnost funguje na obdobném principu a je tedy výhodné využít společného základu pro návrh tříd, uchýlil jsem se k jejich oddělenému návrhu. Hlavním důvodem pro tento krok je odlišný způsob činnosti obou typů čteček. Zatímco kontaktní čtečka karet pracuje na asynchroním principu, činnost bezkontaktní čtečky karet je synchronní. To s sebou nese i odlišné type poskytovaných událostí a metod. Systém dále obsahuje adaptéry *NetworkCommunicator* a *DatabaseConnector* sloužící ke komunikaci se vzdáleným serverem, respektive databází.

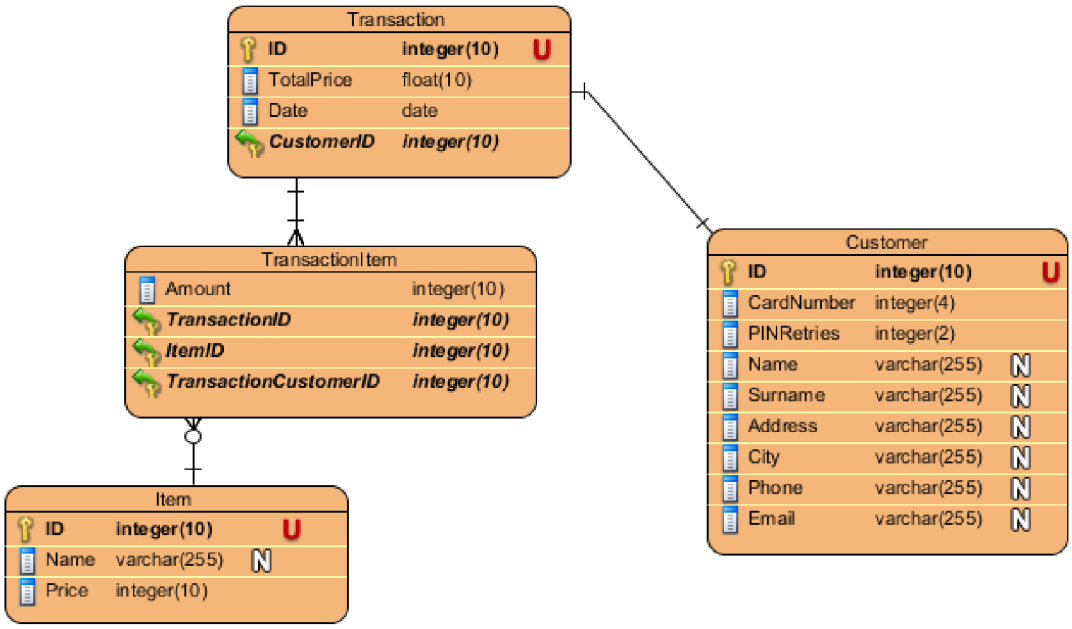


Obrázek č.12: Diagram tříd

5.1.3 ER diagram

ER diagram databáze je zobrazen na obrázku č. 13. Jedná se o schéma jednoduché databáze, která slouží pro zpětnou kontrolu provedených transakcí a částek, které v jejich rámci byly zaplaceny. Využití ukládání dat do databáze bude nepovinná doplňková služba na přání provozovatele systému. Z pohledu popisu je zajímavá entita *Customer*. Do ní se ukládají pouze poslední 4 čísla platební karty tak, aby nedocházelo k ukládání neoprávněných dat. Z tohoto důvodu je každý zákazník zpárován s transakcí v poměru 1 k 1, poslední 4 čísla karty totiž nemohou sloužit k jednoznačné identifikaci

zákazníka a zákazník je tedy namísto čísla karty identifikován jednoznačným identifikačním číslem. Pokud stejný zákazník tedy provede 4 transakce, bude v systému uložen 4x. Entita dále obsahuje položku PINRetries, která udává počet opakovaného zadávání PIN čísla v rámci transakce a umožňuje tak detekovat neobvyklé stavy v systému.



Obrázek č.13: ER diagram

5.2 Integrace pro mobilní telefony

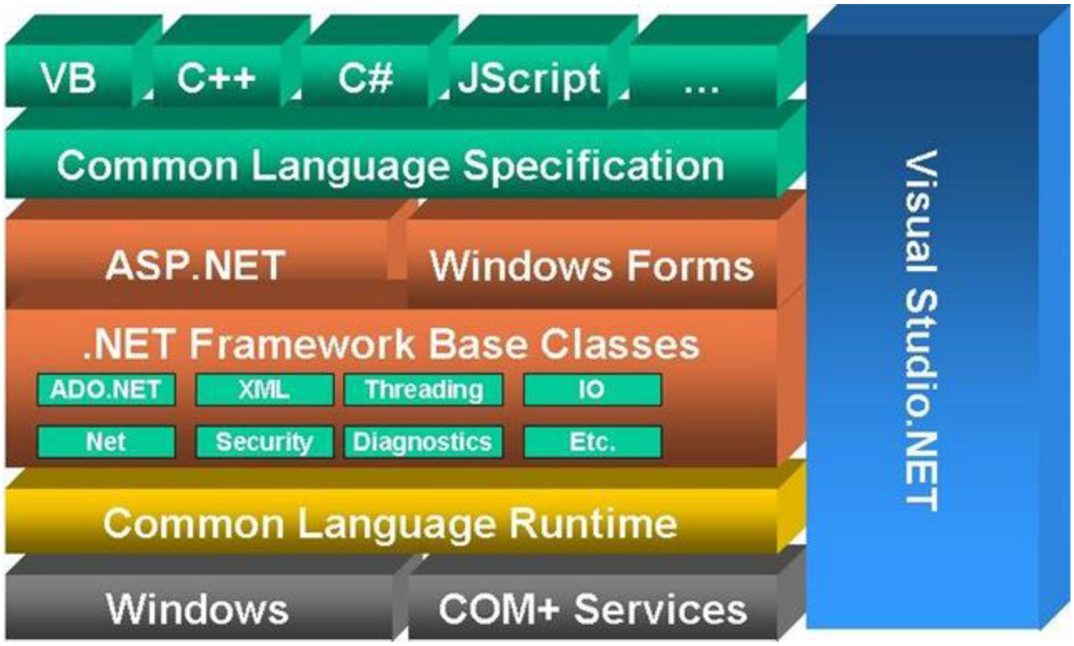
Integrace programu pro mobilní telefony není cílem této práce, nicméně v budoucnosti se s tímto rozšířením počítá, a tak je vhodné provést na toto téma diskuzi. Toto rozšíření bude spočívat v portování programu na operační systém Microsoft Windows Phone. Hlavním úskalím této činnosti je fakt, že tento systém nepodporuje všechny možnosti technologie Microsoft .NET 4.0, ale pouze podmnožinu této technologie pod názvem Microsoft .NET 4.0 Client Profile. Tento fakt může přinést problémy zejména s napojenými systémy pro komunikace s databázemi a servery, při návrhu a implementaci je tedy potřeba dávat pozor na možnost jednoduché změny těchto služeb na jiné prostředky. Dalším problémem jsou jiné požadavky na výkon a ovládání takového systému, to je však problém hlavně navazujících programů které budou obstarávat grafické uživatelské rozhraní.

Mobilní verze knihovny by zároveň měla podporovat integrované NFC čipy, jež jsou součásti některých chytrých mobilních telefonů. Pokud bude tento čip v telefonu přítomen, nabídne aplikace možnost jeho využití. Tento systém by pak výrazně zvýšil komfort plateb pro koncové zákazníky, aplikace na něm však nemůže být založena zejména z důvodu, že takovéto mobilní telefony v současné době nejsou příliš rozšířeny.

6 Použité technologie

6.1 Microsoft .NET framework

Pro implementaci samotné knihovny a jádra přiložených demo aplikací jsem zvolil framework Microsoft .NET. Tato volba byla učiněna z důvodu nutnosti napojení knihovny na softwarová řešení, která byla v této technologii již vyvíjena. Jedná se o rozšířenou vývojou technologii, která je v současnosti využívána v řadě desktopových i webových aplikací. Obdobně jako Java pracuje na principu předkompilovaného kódu do tzv. mezikódu (Intermediate Language), který je pak v reálném čase zpracováván běhovým prostředím architektury .NET.



Obrázek č.14: Architektura Microsoft.NET framework [9]

Na obrázku č.14 je znázorněna struktura architektury Microsoft .NET ve verzi 4.0. Zelená vrstva představuje programovací jazyky použitelné v této architektuře. Tyto jazyky a jejich překladače musí splňovat definovanou specifikaci Common Language Specification. Z programátorského pohledu se vlastně jedná o rozhraní k jazyku, přes které framework Microsoft .NET přistupuje ke kódu při jeho předkompilování do mezikódu. Díky této vlastnosti je umožněna vzájemná spolupráce tříd napsaných a přeložených v překladačích splňujících specifikaci CLS, např. tedy tříd napsaných v jazycích Visual Basic a C++. Je dokonce možné míchat i různé programátorské přístupy, protože kromě objektově orientovaných jazyků (C++, C#, VB…) splňují specifikaci CLS i některé neobjektové jazyky, jako třeba funkcionální jazyk F# [9]. Z této sady

[9] Zdroj: http://2.bp.blogspot.com/_vUz3DOc9B-Q/THYjGIRdRlI/AAAAAAAAAgg/kcvDsjq6wI8/s1600/DotNetArchitecture.jpg

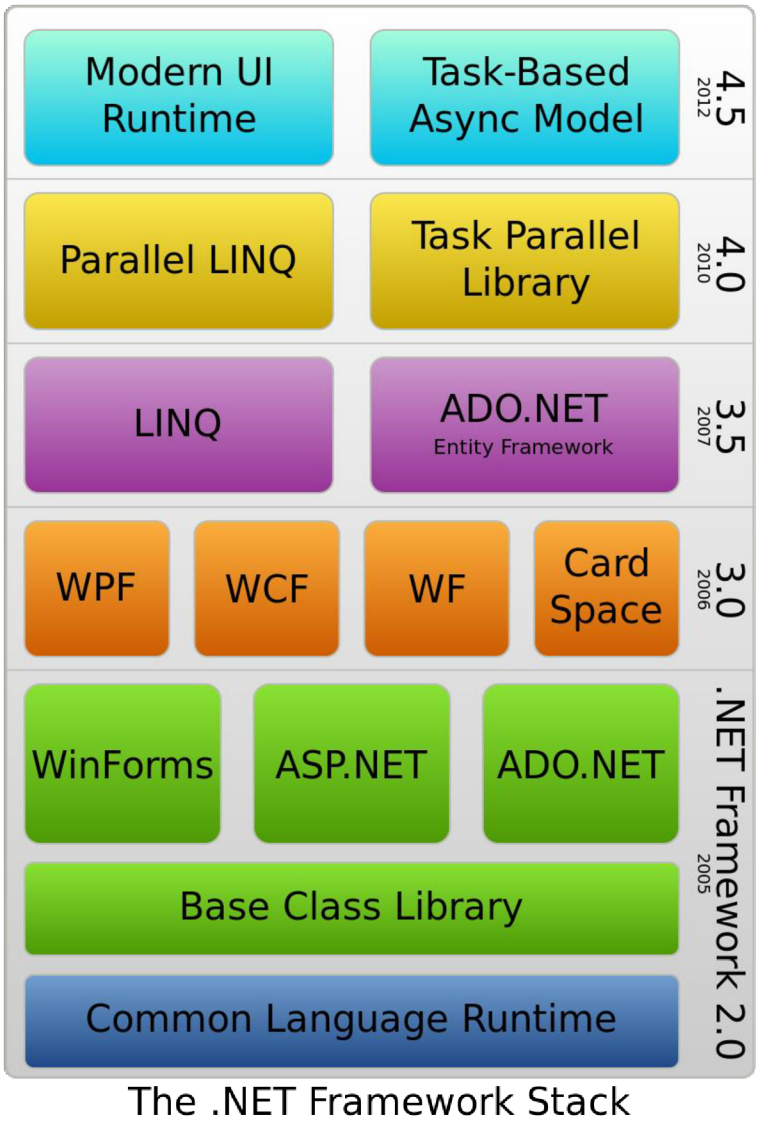
jazyků jsem zvolil jazyk C# pro samotné jádro knihovny a desktopové demo aplikace a jazyk JScript pro některé webové demo aplikace. Důvodem této volby je široká rozšířenost zvolených jazyků v rámci frameworku .NET, moje dlouholetá zkušenost s těmito jazyky a také jejich využívání v rámci programů vyvíjených společností Ki-Wi Digital, s.r.o., se kterou na této knihovně spolupracuji.

Červená vrstva na obrázku č.14 zobrazuje některé z celé řady služeb a funkcí, které framework Microsoft .NET poskytuje prostřednictvím specifikace Common Language Specification. Jde především o základní sadu knihoven, která obsahuje prostředky pro práci se soubory, síťovou komunikaci, databázové rozhraní, zpracování standardizovaných dokumentů (např. XML, JSON, HTML), prostředky pro práci s více vlákny atd. Druhou součástí této vrstvy jsou balíky pro vytváření grafického rozhraní programu a podpůrné funkce v závislosti na cílenou platformu. Jedná se především o webové rozhraní ASP.NET (Active Server Pages) a o větší množství grafických rozhraní pro desktopové aplikace, např. Windows Forms, Windows Presentation Foundation (WPF) či Modern UI [10]. Demo aplikace v rámci mé diplomové práce využívají desktopové grafické rozhraní Windows Presentation Foundation.

Žlutá vrstva – Common Language Runtime – představuje běhové prostředí frameworku Microsoft .NET, které zpracovává předkompilovaný platformově nezávislý mezikód Intermediate Language. Tato běhová prostředí se instalují pomocí instalačních balíků pro konkrétní verze .NET, jsou zpětně kompatibilní se staršími verzemi a musí být přítomny na hostujícím webovém serveru v případě webových stránek nebo na klientském počítači v případě desktopových aplikací. Microsoft balík s běhovým prostředím přidává do instalace operačního systému Windows (konkrétně se jedná o .NET 3.5 v instalaci Windows 7 a o .NET 4.5 v instalaci Windows 8 [10]), je však možné nainstalovat novější verze frameworku, stejně jako je možné nainstalovat běhové prostředí i pod jiným operačním systémem, je-li pro tento systém běhové prostředí k dispozici (např. prostředí Mono pro operační systémy založené na linuxu).

Vývoj platební knihovny a demo aplikací v této práci je cílen na verzi .NET frameworku 4.0, nikoliv na nejnovější verzi 4.5. Důvodem tohoto kroku je požadavek firmy Ki-Wi Digital, s.r.o., která má na většině svých klientských počítačů nainstalovánu právě verzi .NET frameworku 4.0, na které by kód přeložený pod verzí 4.5 nešel spustit. Verze 4.5 však z pohledu této práce nepřináší nic podstatného, tato volba tedy nemá na vývoj zásadní vliv. Jak je vidět na obrázku č.15, nevyužitím .NET frameworku 4.5 přicházím o možnost využití grafického rozhraní Modern UI (místo něj využívám rozhraní Windows Presenation Foundation představené již ve verzi 3.0) a nového přepracovaného asynchroního modelu (na jeho místě využívám Reactive Extensions framework, viz patřičná podkapitola).

[10] Dostupné online na adrese http://blogs.msdn.com/b/astebner/archive/2007/03/14/mailbag-what-version-of-the-net-framework-is-included-in-what-version-of-the-os.aspx



Obázek č.15: Vývoj verzí Microsoft .NET frameworku [11]

6.2 UML

UML je zkratkou pro Unified Modeling Language a slouží pro návrh aplikací. Byl vytvořen za účelem sjednocení postupů v modelovacích technikách a softwarovém inženýrství a od roku 1997 je standardizován [11]. Tento standard se neustále aktualizuje, v době psaní této práce je aktuální verzí UML 2.4.1 ze srpna roku 2011, jeho specifikace je k dispozici na [12].

Jazyk UML používá grafickou syntaxi, jeho výstupem jsou tedy různé diagramy, schémata a grafické značky. Díky jeho standardizaci existuje řada nástrojů, které jsou schopny z UML diagramů generovat kostru kódu programu či naopak z již vytvořeného kódu generovat patřičné UML diagramy. Jedná se o tzv. CASE aplikace (computer-aided software engineering) a může se jednat jak

[11] Zdroj: http://upload.wikimedia.org/wikipedia/commons/thumb/d/d3/DotNet.svg/900px-DotNet.svg.png
[12] http://www.omg.org/spec/UML/2.4.1/

o aplikace sloužící čistě k účelu návrhu (např. Visual Paradigm for UML), tak o součást vývojových prostředí (např. CASE podpora v Microsoft Visual Studio).

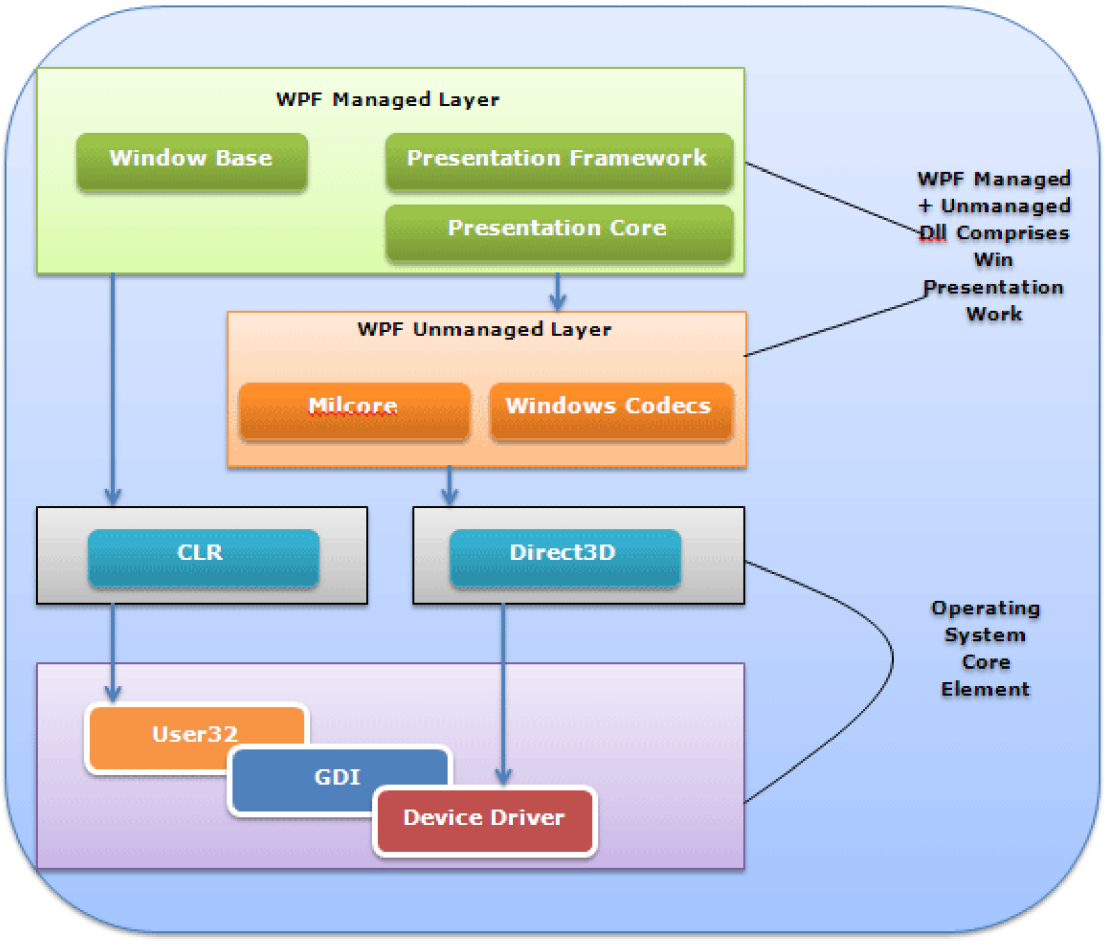
6.3 Windows Presentation Foundation

Windows Presentation Foundation (WPF) je grafické uživatelské rozhraní představené v Microsoft .NET frameworku 3.0. Poskytuje prostředky pro vývoj desktopových grafických aplikací a jejich funkcí.

```
<Window x:Class="DPDemoApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:cardlib="clr-namespace:CardLib.Support;assembly=CardLib"
        Title="DPDemoApp" Height="350" Width="525" Closed="Window_Closed">
    <Grid>
        <TabControl>
            <TabItem Header="Card mode">
                <StackPanel Orientation="Vertical">
                    <TextBlock x:Name="cardText" FontSize="18" Margin="6">Card removed</TextBlock>
                    <Button Content="Create Transaction" Width="150" HorizontalAlignment="Left" Margin="6"
                            x:Name="transactionButton" Visibility="Hidden" Click="transactionButton_Click"></Button>
                    <TextBlock x:Name="printText" FontSize="18" Margin="6" Visibility="Hidden">Do you wish to print receipt?</TextBlock>
                    <Button Content="Print receipt" Width="150" HorizontalAlignment="Left"
                            Margin="6" x:Name="printButton" Visibility="Hidden"></Button>
                </StackPanel>
            </TabItem>
            <TabItem Header="Network mode">
                <DockPanel>
                    <StackPanel Orientation="Horizontal" DockPanel.Dock="Top" Margin="6">
                        <TextBox x:Name="tbUrl" Height="25" Margin="6" Width="280">http://www.fit.vutbr.cz</TextBox>
                        <Button x:Name="testButton" Content="Test" Width="80" Margin="6"></Button>
                        <Button x:Name="navigateButton" Content="Navigate" Width="80" Margin="6"></Button>
                    </StackPanel>
                    <cardlib:WebBrowser VerticalAlignment="Stretch" HorizontalAlignment="Stretch"
                                        x:Name="browser" Source="http://www.fit.vutbr.cz"></cardlib:WebBrowser>
                </DockPanel>
            </TabItem>
            <TabItem Header="XML mode">
                <TextBox x:Name="xmlParser"/>
            </TabItem>
        </TabControl>
    </Grid>
</Window>
```

Obrázek č.16: Ukázka jazyka XAML

Každá uživatelská grafická třída se skládá ze 2 částí – z definice komponenty ve značkovacím jazyku XAML a z kódu na pozadí (code-behind), který je napsán v některém z jazyků podporujících specifikaci Common Language Specification. XAML (Extensible Application Markup Language) je značkovací jazyk sloužící k definici grafického rozhraní v prostředí Windows Presentation Foundation. Kromě definice rozmístění jednotlivých prvků umožňuje také přímo v rámci značek definovat různé animace, spouštěče akcí (triggery) či programové zdroje. Kód na pozadí má větší programové možnosti – vše, co lze definovat v XAMLu, lze vytvořit také pomocí kódu na pozadí, opačně to však neplatí. Pro programovou změnu grafických komponent za běhu programu se využívá třída Dispatcher, ze které dědí všechny grafické objekty. Tato třída zapouzdřuje asynchronní přístup ke grafickým komponentám, protože každá z nich běží ve svém vlastním grafickém vlákně a přístup z ostatních vláken programu, které zpracovávají logickou část kódu, není napřímo možný. Veškeré grafické změny probíhají asynchronně, třída Dispatcher však poskytuje i synchronní metody, které tento stav simulují a blokují volající vlákno do doby, než se provedou požadované změny v grafickém vlákně.



Obrázek č.17: Struktura Windows Presentation Foundation [13]

Na obrázku č. 17 je zobrazena struktura tohoto systému. WPF se skládá ze 2 hlavních vrstev. Řízená (managed) vrstva obsahuje poskytované třídy a grafické objekty, které tvoří grafické uživatelské rozhraní. Programátor má přístup právě k této vrstvě. Výstup této vrstvy se pak promítne jako část předkompilovaného mezikódu, který je zpracováván behovým prostředím Common Language Runtime frameworku Microsoft .NET. Druhou částí balíku WPF je tzv. neřízená (unmanaged) vrstva, která zapouzdřuje systémové části grafickéhu subsystému, komunikuje s grafickým rozhraním Direct3D a funguje bez zásahu programátora.

6.4 XHTML, CSS, jQuery

Z důvodu možnosti běhu některých demonstračních aplikací na zařízeních třetích stran a v nativních webových prohlížečích jsem u těchto součástí nemohl využít grafického rozhraní WPF. Tyto příklady jsem realizoval jako webové stránky, kde je komunikace s knihovnou realizována pomocí vlastních aplikačních URL odkazů. Pokud jsou stránky spuštěny v nativních webových prohlížečích, pak navigace na tyto odkazy nevyvolá žádnou akci, pokud jsou však stránky spuštěny ve vytvořené

[13] Zdroj: http://www.c-sharpcorner.com/UploadFile/nipuntomar/wpf-an-introduction-part-2/Images/WPF3.gif

aplikaci s navázanou knihovnou, pak jsou tyto odkazy zpracované vestavěným kontrolerem a proběhne akce, která je k danému odkazu nadefinována.

Pro tvorbu webových stránek jsem využil značkovací jazyk XHTML (eXtensible HyperText Markup Language), tabulky kaskádových stylů CSS (Cascading Style Sheets) a JavaScriptovou knihovnu jQuery.

Jazyk XHTML je novější derivací jazyka HTML (HyperText Markup Language). Tyto jazyky jsou nejrozšířenějším prostředkem používaným pro popis hypertextových dokumentů. Zatímco jazyk HTML je aplikací jazyka SGML (Standard Generalized Markup Language), mnou používaný XHTML je aplikací novějšího XML (eXtensible Markup Language). Při vytváření jazyka XHTML bylo myšleno na co největší syntaktickou podobnost s jeho předchůdcem, hlavní změnou jsou pak tedy obecné vlastnosti jazyků, na které jsou tyto popisovací jazyky aplikovány. Mezi nejzásadnější změny tedy patří [12]:

- XHTML je striktní, jeden konkrétní úkon je možné popsat pouze jedním možným syntaktickým způsobem – tato vlastnost značně usnadňuje validaci dokumentu.
- XHTML nepodporuje nepárové značky, některé nepárové značky z původního HTML je však možné zapsat zkráceným způsobem (např.
).
- Všechny značky musí být napsány malými písmeny.
- Je zakázáno křížení značek.
- Atributy jednotlivých značek musí být uvedeny v uvozovkách.
- Byly odstraněny značky jazyka HTML určené pro definici vzhledu, jejich funkcionalitu je možné suplovat pomocí CSS stylů.

Tabulky kaskádových stylů CSS poté slouží pro popis vzhledu jednotlivých elementů značkovacích jazyků. Nejčastější způsob jejich implementace spočívá v uložení do externího souboru, který se pak do jednotlivých XHTML dokumentů příkazem importuje. Výhoda tohoto způsobu spočívá ve sdílení těchto tabulek mezi více dokumenty, a tedy dosažení shodného vzhledu jednotlivých prvků napříč celým webovým systémem [12].

Pro vylepšení programových možností stránek jsem využil možností JavaScriptu, konkrétně pak knihovny jQuery. Tato knihovna obsahuje např. funkce pro přístup k DOM elementům a funkce podporující zpracování událostí a uživatelských akcí. Ve své práci využívám knihovnu ve verzi 2.1.0 z ledna roku 2014. Tato knihovna je distribuována jako volně dostupná s otevřeným zdrojovým kódem a je licencována pod licencí MIT [14].

[14] Licenční podmínky dostupné na https://jquery.org/license/

6.5 LINQ

Dotazovací jazyk LINQ (Language-Integrated Query) je dotazovací jazyk, jež je součástí frameworku Microsoft .NET. Poskytuje unifikovaný přístup k těmto zdrojům dat [13]:

- kolekce poskytované frameworkem Microsoft .NET (např. *List<T>*, *Dictionary<T,T>*…),
- databáze SQL,
- datové sety komponent ADO.NET,
- XML dokumenty.

Vzhledem k možnosti vlastní implementace komponent ADO.NET je možné přistupovat i k jiným zdrojům dat, např. JSON objektům. Díky unifikaci poskytované tímto jazykem je možné měnit zdroje dat pro vytvořené aplikace, a to dokonce i dynamicky za běhu programu. Od verze 4.0 framework Microsoft .NET obsahuje navíc jazyk LINQ nativní podporu paralelního zpracování dotazů, což výrazně pomáhá zejména v práci s objemově většími zdroji dat.

6.6 Reactive Extensions Framework

Reactive Extensions Framework (označovaný též jako Rx Framework) je nadstavba pro asynchronní model platformy .NET, která je vyvíjena korporací Microsoft Open Technologies, Inc. ve spolupráci s komunitou. Jedná se o vylepšení v současné době již nevyhovujícího systému událostí v původním .NET frameworku až do verze 4.0 včetně. Nejnovější verze 4.5 již obsahuje nový asynchronní model, který přímo vychází právě z Reactive Extensions Framework.

Tento framework dokáže pracovat jak s požadavky na synchronní činnost – v tomto případě vrací přímo výsledek (*T*) nebo kolekci výsledků (*IEnumerable<T>*), tak s požadavky na asynchronní zpracování – v tomto případě vrací odkaz na úlohu, která bude vykonána v separátním vlákně a v nedefinovaném čase vrátí výsledek typu T (*Task<T>*), nebo sledovatelnou kolekci těchto úloh v případě požadavku na více návratových hodnot (*IObservable<T>*). Zároveň umožňuje asynchronní spouštění událostí na základě oznamujících prvků (publishers) a členů, jež jim naslouchají a provádějí definované akce (subscribers). Celý framework funguje paralelně.

Tento framework je distribuován jako volně dostupný s otevřeným zdrojovým kódem a je licencován pod licencí Apache License 2.0 [15].

6.7 Awesomium

Pro zobrazování webových stránek v demo aplikacích jsem zvolil webový prohlížeč Awesomium. Framework .NET 4.0 sice nabízí integrovaný webový prohlížeč, ten je však postavený na prohlížeči

[15] Licence dostupná na https://rx.codeplex.com/license

Microsoft Internet Explorer, který nesplňuje požadavky pro námi zobrazované webstránky. Prohlížeč Awesomium je oproti tomu postaven na webovém jádru Chromium a nabízí rozhraní pro jazyk C++ a prostředí .NET a jeho jazyky (C#, Visual Basic…). V rámci prostředí .NET poskytuje grafický výstup pro prostředí Windows Forms a Windows Presentation Foundation. Pro účely této práce vzhledem ke zvoleným technologiím využívám Awesomium pro .NET s grafickým výstupem WPF.

V aplikacích v rámci této práce využívám Awesomium ve verzi 1.7.3.0, která je přeložena s jádrem Chromium v18. Awesomium je volně použitelné pro nekomerční využití a pro komerční využití u firem s obratem menším než 100 000 dolarů za rok. Awesomium je licencované pod vlastní licencí [16].

6.8 Použité nástroje

6.8.1 Microsoft Visual Studio 2010

Microsoft Visual Studio 2010 je integrované vývojové prostředí sloužící především pro tvorbu aplikací založených na Microsoft .NET frameworku. Kromě prostředků pro psaní programového kódu a jeho překlad obsahuje také prostředí pro tvorbu uživatelských prostředí v podporovaných technologiích, dále integrovaný debugger, automatický našeptávač IntelliSense, prostředky pro sdílení a verzování kódu či připojování a nastavování databází, včetně integrované testovací verze databázového systému Microsoft SQL Server.

Aktuální verzí Microsoft Visual Studia je verze 2013 [17]. Ve své práci využívám starší Microsoft Visual Studio 2010 ve verzi Professional, a to z důvodu licenčních ujednání v rámci společnosti Ki-Wi Digital, s.r.o. Hlavní změnou v novější verzi je podpora Microsoft .NET Frameworku 4.5 a vývojové prostředky pro jeho nové vlastnosti, zejména pro grafické rozhraní Modern UI. Vzhledem k faktu, že knihovna je cílená pro .NET verze 4.0 a využívá grafické uživatelské rozhraní Windows Presentation Foundation, které je ve verzi Microsoft Visual Studio 2010 plně podporované, nemá použití této verze na výslednou knihovnu zásadní vliv.

6.8.2 Visual Paradigm for UML 11.1

Pro tvorbu návrhů aplikace v jazyku UML využívám program Visual Paradigm for UML od společnosti Visual Paradigm International, a to ve verzi 11.1 Community [18]. Tento software splňuje standard UML verze 2.1. a přidává některé vlastní prostředky pro tvorbu návrhů, zejména pak

[16] Licence dostupná na http://wiki.awesomium.com/licensing/licensing-overview.html
[17] Dostupné na http://www.visualstudio.com/downloads/download-visual-studio-vs
[18] Dostupné na http://www.visual-paradigm.com/download/

v oblasti týmové spolupráce. Software Visual Paradigm for UML je ve verzi Community dostupný zdarma pro nekomerční účely [19].

[19] Licenční podmínky dostupné na http://www.visual-paradigm.com/support/documents/vpumluserguide/12/13/7572_licensing.html

7 Implementace knihovny

7.1 Rozšířitelnost knihovny

Při implementaci knihovny a potřebných součástí jsem bral zejména zřetel na její budoucí rozšířitelnost, a to jak po stránce funkční, tak po stránce podporovaných hardwarových zařízení. Vzhledem k jejímu nasazení do praxe, které probíhá již v rámci této diplomové práce a ve větší míře bude pokračovat i po jejím dokončení, je tato její vlastnost prioritní a bylo potřeba na ní brát při implementaci zřetel.

V průběhu teoretického řešení práce jsem narazil na několik problémů. Ačkoliv se platební model a integrované platební karty řídí standardem EMV, většina ostatních součástí tohoto procesu standardizována není. Tento problém je ještě umocněn požadavkem na zpracování údajů z vlastních zákaznických karet, zejména pak z různých klubových karet některých zákaznických společností firmy Ki-Wi Digital, s.r.o. Prakticky všechny logické součásti knihovny bylo tedy zapotřebí napsat obecně za pomoci prostředků, které k tomuto účelu poskytuje framework Microsoft .NET tak, aby jejich rozšíření či záměna nečinila v budoucnosti problém. K těmto obecným implementacím jsem pak vytvořil i některé konkrétní implementace pro specifické karty a jejich čtečky a tyto jsem využil v ukázovkých demo aplikacích.

Hlavním prostředkem, který jsem k tomuto účelu využil, je programové rozhraní, neboli *Interface*. Jedná se o předpis metod a vlastností (*properties*), které musí třída implementující toto rozhraní obsahovat. U vlastností se jedná o její typ, název a o nutnost implementace přístupových modifikátorů *get* a *set*. U metody se pak jedná o její hlavičku, tedy návratový typ metody, její název a typy a názvy parametrů metody. Všechny vlastnosti a metody definované v rozhraní jsou automaticky považovány za veřejně přístupné (*public*). Program pak ke třídám, které rozhraní implementují, přistupuje právě skrze definici tohoto rozhraní a konkrétní implementace pro něj není důležitá.

Dalším prostředkem, který jsem v implementaci knihovny využil, jsou modifikátory tříd a metod. Tento přístup se využívá v případě, že chci poskytnout nějaký základní kód, který se dá ve většině případů využít, ale chci dát autorovi konkrétní implementace možnost tento kód ignorovat nebo doplnit tak, aby stále původní třída splňovala podmínky rozhraní. K tomuto účelu využívám abstraktních tříd, které jsou označeny klíčovým slovem *abstract*. Toto klíčové slovo uvozující třídu znamená, že tato třída nemůže být použita přímo, ale pouze prostřednictvím jejího zdědění implementovanou třídou. Po jejím zdědění je možné volat metody základní třídy, stejně tak jako je přepsat vlastním kódem. Přepisování metod se provádí klíčovým slovem *override*.

U abstraktních tříd využívám 3 druhy metod, které určují nutnost jejich implementace ve zděděné třídě:

- Metody bez uvozujícího modifikátoru – tyto metody mají v abstraktní třídě vlastní implementaci, kterou není možné ve zděděné třídě přepsat. Vždy bude použita implementaci metody uvedená v abstraktní třídě. Tento způsob využívám v případě jasně dané implementace pro konkrétní případ, zejména pak u standardizovaných situací.
- Metody uvozené klíčovým slovem *virtual* – jedná se o virtuální metody, které mají v abstraktní třídě vlastní implementaci. Tuto implementaci je však možné ve zděděné třídě přepsat. Pokud ve zděděné třídě není uvedeno přepsání takovéto metody, bude použita implementace uvedená v abstraktní třídě, v opačném případě bude použita implementace ve zděděné třídě. Zároveň je možné provést v přepisované metodě pouze její rozšíření, a to pomocí kombinace naprogramování takového rozšíření a zavolání původní implementace prostřednictvím syntaxe *base.NazevMetody()*. Tento způsob využívám v případech, ve kterých je vhodné uvést implementaci (například z důvodu její využitelnosti pro většinu zařízení), zároveň však není garantována funkčnost této implementace pro všechna zařízení či karty, a proto chci dát autorovi implementace možnost ji upravit.
- Metody uvozené klíčovým slovem *abstract* – jedná se o abstraktní metody, které nemají v abstraktní třídě vlastní implementaci, ale pouze definici metody. Zděděná třída poté musí obsahovat vlastní implementaci této metody. Tento způsob využívám u metod natolik specifikých pro dané zařízení, že není možné využít obecnou implementaci.

```
public abstract class BankAPI : IBankAPI
{
    public virtual void Connect(Uri url)...

    public virtual void Disconnect()...

    public abstract AuthorizeResponse AuthorizeCard(CardInfo cardInfo);

    public abstract AuthorizeResponse AuthorizeRequest(BankRequest bankRequest);

    public abstract AuthorizeResponse BlockCard(CardInfo cardInfo);
}
```

Obázek č.18: Ukázka rozšířitelné abstraktní třídy

7.2 Podpora hardwarových zařízení

S ohledem na využívání značně specifických hardwarových zařízení je potřeba zajistit pro ně v knihovně patřičnou podporu. Jedná se zejména o vstupní zařízení pro zadávání PIN či jiných kódů nebo textu, čtečky kontaktních a bezkontaktních platebních karet, čtečky RFID a NFC bezkontaktních zákaznických karet a tiskárny.

Vstupní zařízení pro zadávání znaků a jeho vlastnosti jsou definovány v rozhraní *ITextInput*. K dispozici je i abstraktní třída *TextInput*, která obsahuje sadu metod s implementací pro klasické

klávesnice nebo zařízení, které pracují na principech klávesnice. Využití myši je v knihovně nativní a není potřeba jej implementovat.

Vzhledem k výrazné odlišnosti principu fungování kontaktních a bezkontaktních karet jsem po úvahách přistoupil k jejich separátní implementaci, ačkoliv se na první pohled zdálo výhodné využít společný základ pro obě implementace. Pro kontaktní karty je definováno rozhraní *ICardReader* a abstraktní třída *CardReader*, pro bezkontaktní karty pak analogicky rozhraní *IContactlessCardReader* a abstraktní třída *ContactlessCardReader*. Dále jsem naimplementoval konkrétní řešení pro bezkontaktní čtečku ACR122U ve třídě *ACR122UCardReaderImpl*.

Definice požadavků pro tiskárny je uvedena v rozhraní *IPrinter* a standardní implementace vytvořena v abstraktní třídě *Printer*. Tato třída využívá systémových tiskáren a její metody umožňují tisk jak pomocí systémového tiskového dialogu operačního systému Microsoft Windows, tak bezdialogový tisk na primární systémovou tiskárnu s využitím aktuálně nastavených vlastností této tiskárny.

Knihovna také obsahuje přípravu pro komunikaci s ostatním hardwarem. Definice rozhraní se nachází v *IGenericHardwareCommunicator*.

7.3 Logická část knihovny

Logická část knihovny pracuje s údaji, které získala z karet a navázané aplikace, a zajišťuje provedení požadovaných úkonů. Dále řídí podpůrné funkce, jako např. tisk, programovou spolupráci nebo komunikaci s jiným hardwarem.

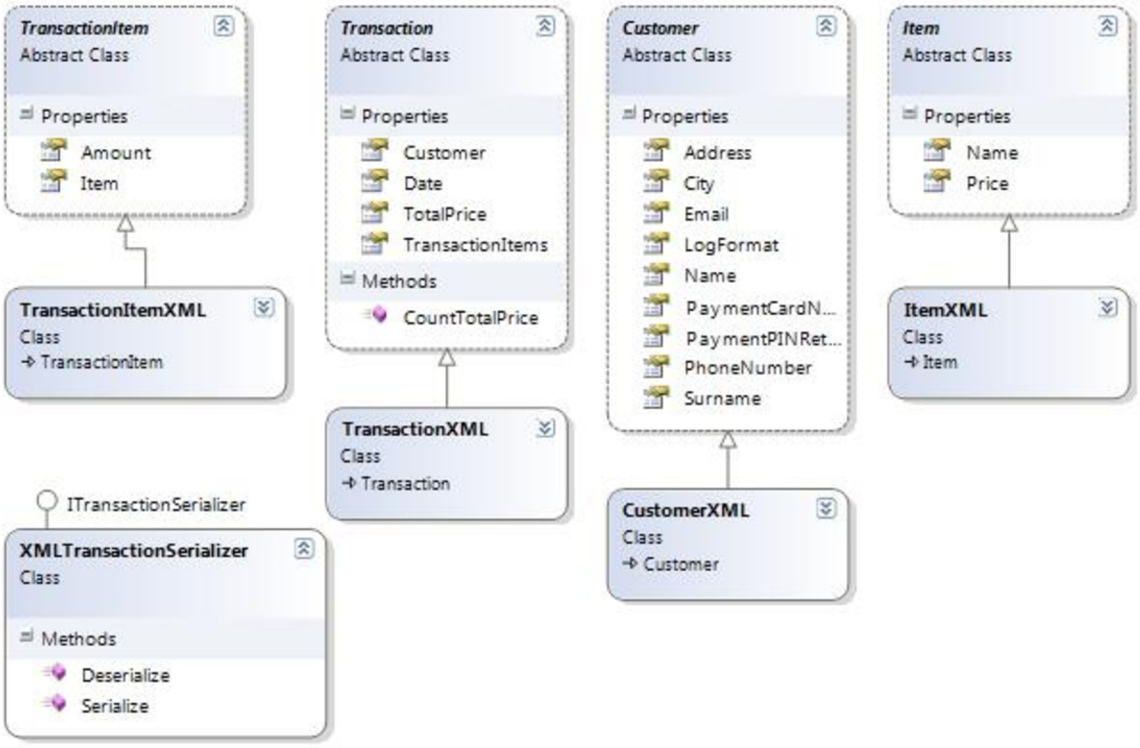
Pro účely evidence transakcí jsou definovány abstraktní třídy *Transaction*, *TransactionItem*, *Item* a *Customer*. Očekává se jejich konkrétní implementace tak, aby splňovaly serializační pravidla systému ADO.NET a byly tím pádem obecně použitelné skrz dotazovací jazyk LINQ. Pro účely serializace je definováno rozhraní *ITransactionSerializer* a konkrétní implementace *XMLTransactionSerializer*.

Další součástí je komunikace s bankovním programovým rozhraním. Jeho definice se nachází v rozhraní *IBankAPI* a implementace pak v abstraktní třídě *BankAPI*. Pro komunikaci s jiným než bankovním systémem je vytvořeno rozhraní *ISystemCommunicator*, které vzhledem k různorodosti těchto systému nemá vlastní abstraktní implementaci a je potřeba jej tedy implementovat přímo.

Pro testování dostupnosti vzdálených stanic je implementován singleton *NetworkManager*, pro databázové operace pak singleton *DatabaseManager*. Aplikace dále obsahuje řadu podpůrných tříd pro úschovu datových struktur. Všechny třídy jsou přiloženy ve formě diagramu tříd v příloze 11.1.

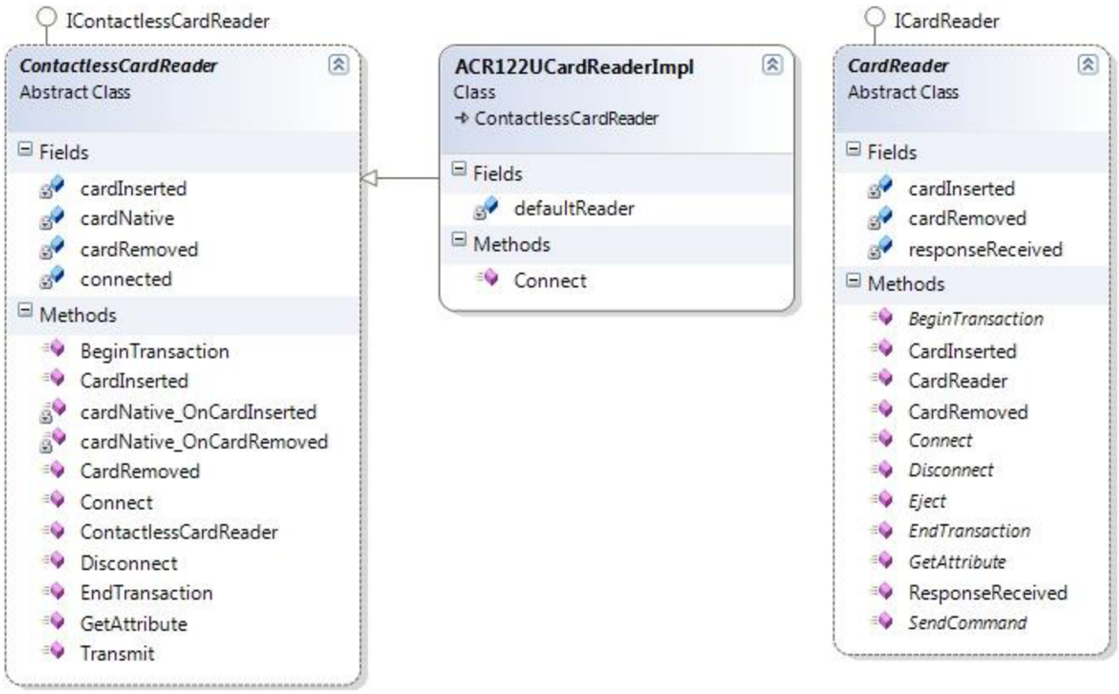
7.4 Popis tříd a rozhraní

Komplexní pohled na diagram tříd je přiložen jako příloha 11.1. V této podkapitole se zaměřím na detailnější popis jednotlivých tříd a rozhraní.



Obrázek č.19: Datové struktury pro ukládání transakcí

První skupinou tříd jsou třídy pro ukládání informací o provedených transakcích. Jedná se o abstraktní datové struktuy *TransactionItem, Transaction, Customer* a *Item*, které slouží k uchování informací o samotných transakcích. Každá z těchto tříd má svoji specifickou implementaci, v pořadí *TransactionItemXML, TransactionXML, CustomerXML* a *ItemXML*. Tyto implementace rozšiřují původní abstraktní třídy o definice kořenových XML elementů. Ukládání těchto tříd do souboru a také znovuvytvoření těchto tříd ze souboru má na starosti třída *XMLTransactionSerializer*, která implementuje rozhraní *ITransactionSerializer*. Tato třída je napsána podle zásad ADO.NET serializace.



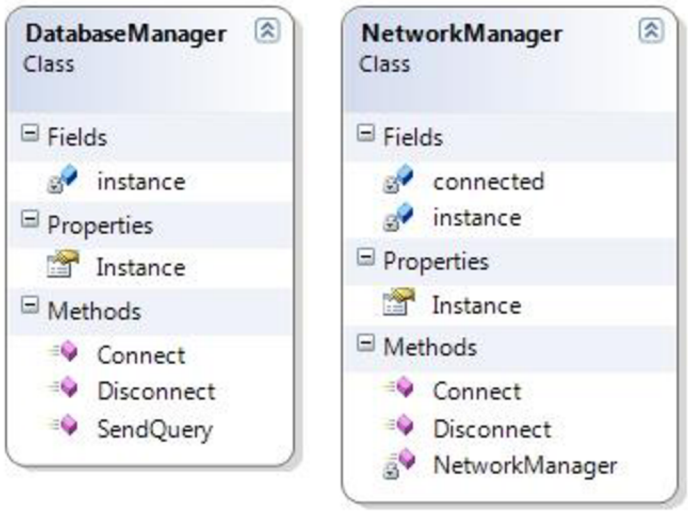
Obrázek č.20: Třídy implementující čtečky karet

Následují třídy, které implementují kontaktní a bezkontaktní čtečky karet. V případě kontaktních čteček se jedná o abstraktní třídu *CardReader*, která implementuje rozhraní *ICardReader*. V této třídě jsou definovány asynchronní Reactive Extensions objekty, ke kterým je možné se přihlásit – jedná se o události vložení karty, vyjmutí karty a obdržení asynchronní reakce od čtečky. Třída dále definuje metody potřebné pro komunikaci s kartou, včetně metod specifických pro kontaktní čtečky, jako např. metoda *Eject()*, která odešle čtečce zprávu pro vysunutí karty (pokud to čtečka podporuje). Posílání příkazů a přijímání odpovědí je řešeno pomocí vlastních datových struktur.

Základní implementaci podpory čteček bezkontaktních karet obsahuje abstraktní třída *ContactlessCardReader*, která implementuje rozhraní *IContactlessCardReader*. Tato třída obsahuje implementaci pro obecné NFC karty, její metody jsou však virtuální a není tedy problém je přepsat na podporu specifických druhů NFC karet. Oproti třídě pro kontaktní čtečky neobsahuje asynchronní událost oznamující odeslání odpovědi ze čtečky, to se totiž děje synchronně jako návratová hodnota metody *Transmit()*. Pro definici příkazů a přijmutí odpovědí využívám datových struktur z knihovny GemCard [20], která zároveň slouží pro komunikaci se čtečkou na hardwarové úrovni. Tato knihovna je licencována pod otevřenou licencí CPOL [21]. Formát těchto datových struktur je podrobně rozebrán v kapitole 3.4 – Platební rozhraní. Pro abstraktní třídu *ContactlessCardReader* jsem dále vytvořil

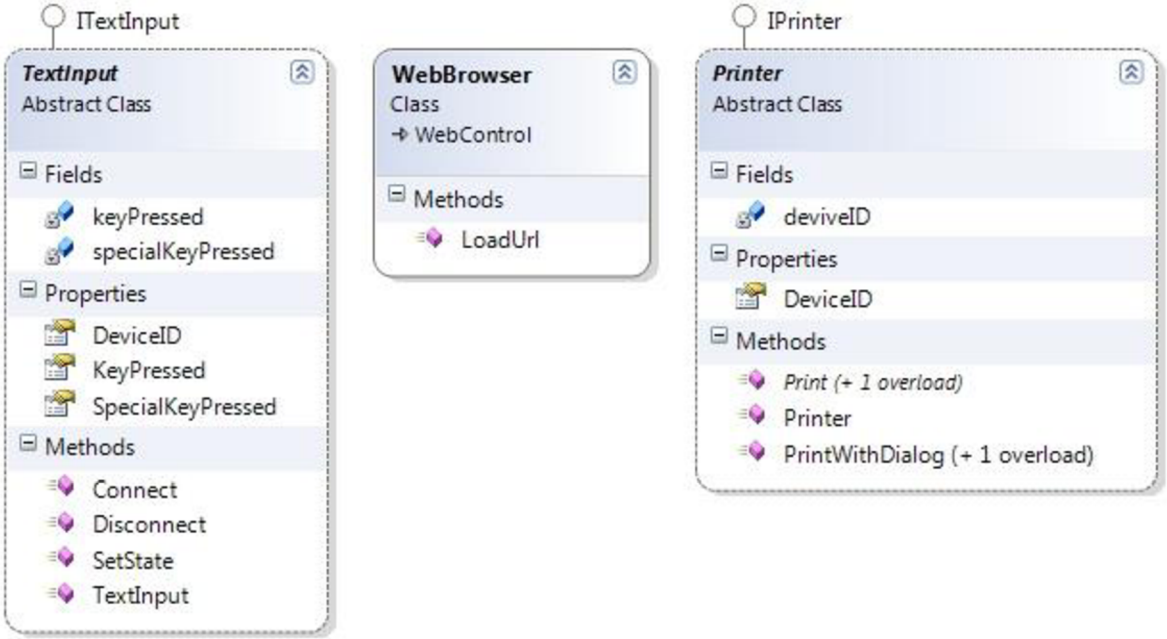
[20] Dostupná na http://www.codeproject.com/Articles/16653/A-Smart-Card-Framework-for-NET
[21] Licence dostupná na http://www.codeproject.com/info/cpol10.aspx

konkrétní implementaci *ACR122UCardReaderImpl*, která implementuje funkčnost a hlavně identifikaci této čtečky v systému.



Obrázek č.21: Síťové singleton třídy

V knihovně jsou dostupné 2 třídy určené pro síťovou komunikaci. Jedná se o třídu *NetworkManager* sloužící k testování a udržování spojení na vzdálený server, a o třídu *DatabaseManager*, která slouží pro připojení a práci s databází. Všechny metody v těchto třídách jsou virtuální a je tedy možné je po zdědění třídy upravit podle vlastních představ. Vzhledem k požadavku na jediné spojení jak se vzdáleným serverem tak s databázovým systémem v celé aplikaci jsem tyto třídy implementoval pomocí návrhového vzoru *singleton*, který zajišťuje přítomnost jediné instance obou zmiňovaných tříd v celé aplikaci [14].



Obrázek č.22: Pomocné třídy

Poslední skupinou tříd, o které se v této podkapitole zmíním, jsou třídy implementující pomocnou funkčnost knihovny. Jedná se o třídu *WebBrowser*, která zpřístupňuje grafický webový prohlížeč Awesomium. Díky tomuto zaobalení je možné implementovat vlastní metody, např. pro automatické bezdialogové přihlašování aplikace do proxy sítě. Třída *TextInput*, která implementuje rozhraní *ITextInput*, slouží pro přijímání stisku kláves ze vstupního zařízení. Implementace v této třídě zajišťuje funkčnost zařízení pracujících na principech běžných klávesnic, jelikož je ale tato podpora v aplikaci nativní, slouží tato abstratkní třída především k implementaci jiných vstupních zařízení. Kromě znakových kláves umožňuje i zasílání informací o stisku speciálních kláves pomocí jejich kódu. Třída *Printer*, implementující rozhraní *IPrinter*, umožňuje tisk textových a grafických dokumentů jak se zobrazením tiskového dialogu, tak bez jeho zobrazení.

7.5 Navázání na bankovní systémy

Navázání na bankovní systémy je popsáno v rozhraní *IBankAPI*. Bankovní rozhraní nejsou standardizována a jejich forma se tedy liší u každé banky, která tyto služby poskytuje. Základ pro implementaci pak poskytuje abstraktní třída *BankAPI*, která však vzhledem k různorodosti těchto bankovních rozhraních obsahuje značné množství metod, které musejí být naimplementovány až v konkrétní implementice pro specifické bankovní rozhraní. Pro připojení ke vzdáleným serverům je možné využít singleton *NetworkManager*, který je ke knihovně přibalený.

Implementace této části knihovny byla nejnáročnější z hlediska rozšiřitelnosti a možnosti napojení na různá bankovní rozhraní. Výsledné rozhraní *IBankAPI* staví kromě teoretického základu také na experimentech a problémech, na které jsem narazil pří vývoji jedné z implementací. Nejednotnost rozhraní jednotlivých bank mě postavila před úkol vyřešení co nejlepšího možného sjednocení jejich funkcí tak, aby bylo možné vykonávat i specifická volání typická pouze pro některé systémy. Z toho důvodu jsem do abstraktní třídy implementoval i obecné metody *SendGenericCommand* a Reactive Extensions událost *ReceiveGenericResponse*, díky kterým je možné pracovat i s příkazy, které nejsou v rozhraní explicitně definovány. V případných budoucích verzích knihovny však nevylučuji možnost rozšíření původního rozhraní *IBankAPI* tak, aby se zvýšil komfort provádění bankovních operací. V takovém případě bude potřeba vyřešit zpětnou kompatibilitu s původním rozhraním.

Při vývoji této části knihovny jsem spolupracoval se společností Ki-Wi Digital, s.r.o., která byla zapojena do projektu pro rozšíření bezkontaktních platebních karet, který organizovala společnost Komerční Banka, a.s.. V průběhu vývoje však byla tato spolupráce ukončena a tím pádem aktuální verze knihovny neumožňuje provádět platby přes tuto bankovní instituci.

```
public interface IBankAPI
{
    void Connect(Uri url);

    void Disconnect();

    IObservable<object> ReceiveGenericResponse();

    void SendGenericCommand(object command);

    AuthorizeResponse AuthorizeCard(CardInfo cardInfo);

    AuthorizeResponse AuthorizeRequest(BankRequest bankRequest);

    AuthorizeResponse BlockCard(CardInfo cardInfo);
}

public class AuthorizeResponse
{
    public bool AuthorizeResult { private set; get; }

    public Dictionary<string, object> ResponseArguments { private set; get; }

    public AuthorizeResponse(bool authorizeResult, Dictionary<string, object> responseArguments)
    {
        this.AuthorizeResult = authorizeResult;
        this.ResponseArguments = responseArguments;
    }
}
```

Obázek č.23: Rozhraní IBankAPI a struktura AuthorizeResponse

7.6 Zaznamenávání transakcí

Jedním z požadavků byla také možnost zaznamenat transakce provedené v rámci systému. Transakce a její prvky jsou v knihovně reprezentovány abstraktními třídami *Transaction*, *TransactionItem*, *Item* a *Customer*, jejich konkrétní implementace pak umožňují tyto prvky nejenom blíže specifikovat, ale také definovat způsob jejich serializace, pokud to vyžaduje způsob jejich uchování.

Pro použití ve větších projektech se jako ideální možnost ukládání jeví využití relačního databázového systému SQL. Ukládání do takové databáze je možné buďto vytvoření ADO.NET konektoru pro SQL a poté serializací uvedených tříd, nebo je možné využít postup bez nutnosti serializace napsáním manager třídy, která bude vyvolávat potřebné SQL dotazy. V obou případech je možné využít v knihovně přítomné třídy *DatabaseManager* pro připojení a obsluhu databáze a *NetworkManager* pro přístup ke vzdáleným serverům.

Při vývoji ukázkových aplikacích jsem musel řešit problém, kdy byl zákazníkem vznešen požadavek na možnost offline běhu aplikace, kdy by se případné transakce realizovaly až dávkově prostřednictvím změn v databázi transakcí. Tento způsob běhu aplikace je možný, protože se nejednalo o bankovní transakce, u kterých by bylo nutné detaily transakce okamžitě online ověřit, ale o vlastní klientský systém zákazníka. Problémem však bylo ukládání těchto transakcí. Vzhledem k offline provozu nebylo možné transakce ukládat na vzdálený SQL server a protože výkon tabletů,

na kterých aplikace pro zákazníka běžela, nebyl dostačující pro běh lokální SQL databáze, musel jsem zvolit jiný způsob. Nakonec jsem se uchýlil k ukládání lokálních souborů ve formátu XML, které jsem provedl pomocí implementace *XMLTransactionSerializer* rozhraní *ITransactionSerializer* a nedefinováním patřičných tříd *TransactionXML*, *TransactionItemXML*, *ItemXML* a *CustomerXML*.

```
<?xml version="1.0" encoding="utf-8"?>
<Transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <TotalPrice>470</TotalPrice>
  <Date>2014-05-26T20:17:41.1209463+02:00</Date>
  <Customer>
    <Name>Jan</Name>
    <Surname>Novák</Surname>
    <Address>Božetěchova 2</Address>
    <City>Brno</City>
    <Email>fit@vutbr.cz</Email>
    <PhoneNumber>+420777111222</PhoneNumber>
    <PaymentCardNumber>6194</PaymentCardNumber>
    <PaymentPINRetries>0</PaymentPINRetries>
    <LogFormat>xml</LogFormat>
  </Customer>
  <TransactionItems>
    <TransactionItem>
      <Amount>3</Amount>
      <Item>
        <Name>Žárovka</Name>
        <Price>60</Price>
      </Item>
    </TransactionItem>
    <TransactionItem>
      <Amount>1</Amount>
      <Item>
        <Name>Lampička</Name>
        <Price>290</Price>
      </Item>
    </TransactionItem>
  </TransactionItems>
</Transaction>
```

Obrázek č.24: Ukázka XML souboru v testovací aplikaci

7.7 Rozhraní pro komunikaci s hardwarem třetích stran

V rozhraní *IGenericHardwareCommunicator* je nastíněna příprava pro komunikaci knihovny s hardwarem třetích stran. Jedná se o velmi obecný způsob komunikace s cizím hardwarem, který bude muset vždy být implementován na míru konkrétnímu zařízení. Z tohoto důvodu pro toto rozhraní neexistuje abstraktní implementace – každé zařízení je natolik specifické, že implementace virtuálních metod postrádá smysl.

Cílem tohoto rozhraní je zprostředkovat zpracování údajů z karet a případně provedení bankovní platby pro ostatní zařízení. Jako příklad uvádím automat na kávu, na kterém si zákazník klasickým způsobem navolí požadovaný nápoj, automat pošle přes rozhraní informace o zvolené položce řídícímu počítači, tento počítač zpracuje platbu pomocí bezdotykové platební karty a úspěch

či neúspěch této činnosti oznamí přes rozhraní zpět automatu. Ten v případě neúspěchu objednávku zruší, v opačném případě vydá požadovaný nápoj.

7.8 Podpůrné funkce

Kromě výše uvedených vlastností disponuje knihovna také některými podpůrnými funkcemi. Tyto funkce neplní primární účel knihovny, ale jsou nějakým způsobem užitečné pro očekávané používání knihovny a usnadňují tak práci s touto knihovnou.

Mezi tyto funkce patří především možnost tisku. Definice požadavků na metody tisku jsou uvedeny v rozhraní *IPrinter* a abstraktní implementace pak ve třídě *Printer*. Pouhým zděděním třídy *Printer* bez dalších zásahů získá uživatel možnost tisknout pomocí systémových tiskáren počítače, a to jak pomocí systémového dialogu tiskáren, tak bezdialogově – v tomto případě bude použita hlavní tiskárna systému ve svém aktuálním nastavení. Systém však umožňuje také přepsat metody třídy *Printer* a napsat si vlastní implementaci například na různé průmyslové mini-tiskárny sloužící k tisku účtenek a potvrzujících dokumentů. Systém pro tisk umožňuje jednoduchý tisk předaného textového řetězce s podporovaným jednoduchým formátováním pomocí znaků konce řádku a tabelátoru, nebo tisk dokumentu, jehož grafický vzhled je nadefinován v jazyku XAML.

7.9 Testování

Vzhledem k modulárnímu návrhu knihovny jsem přistoupil k testování jednotlivých abstraktních tříd implementujících některé z nabízených rozhraní, tříd s konkrétní implementací některého z abstraktních tříd nebo nebo tříd přímo implementujících některé z rozhraní, a to v době jejich dokončení. Po otestování jednotlivých součástí následovalo testování spolupracujících celků a funkčnosti kompletní knihovny. V případě objeveného problému jsem po jeho opravě opět postupoval od testu jednotlivých součástí výše v hierarchii.

Dále jsem testoval spolupráci s externím hardwarem který knihovna podporuje. Opět jsem postupoval od testu konkrétních modulů které s daným hardwarem přímo spolupracují, dále jsem testoval moduly které s hardwarem spolupracují nepřímo a využívají informací získaných ze přímo spolupracujících modulů a na závěr funkčnost spolupráce kompletní knihovny s hardwarem. Jako obzvlášť zajímavá a zároveň problematická se ukázala funkčnost knihovny s větším množstvím připojeného hardwaru zároveň. Chyby zjištěné v průběhu těchto testů si vyžádaly časově nejnáročnější opravy v rámci celého testování.

Neméně důležitou součástí testování bylo též zjištění funkčnosti komunikace s externím softwarem. Zde jsem testoval jednotlivé moduly vůči vlastním testovacím programům, které vyžadovaly nebo naopak zasílaly do knihovny svoje požadavky. Po úspěšném otestování spolupráce

separátních modulů jsem pak testoval knihovnu jako celek ve spolupráci se softwarem Ki-Wi Platform a Ki-Wi Interactive, se kterými jsem zároveň vyvinul některé ukázkové demo aplikace.

7.10 Porovnání s existujícími systémy

Jedním z hlavních důvodů vývoje této knihovny byla minimalizace ceny vlastního systému pro integraci platebních terminálů a bezdotykových zákaznických karet. Tento požadavek byl motivován zájmem některých zákazníků společnosti Ki-Wi Digital, s.r.o. o takový systém, jelikož cena stávajících systémů neodpovídala jejich obchodním požadavkům.

Vlastní implementací jsem docílil tohoto požadovaného cíle. Díky otevřenému návrhu není knihovna vázána na konkrétní hardwarové zařízení a umožňuje tak využití levnějších čteček, které běžně nejsou u ostatních systému k dispozici. Zároveň je možné využít i případných již existujících čteček u klienta, čímž se dá cena takového řešení srazit na ještě nižší hodnotu.

Další odlišností od ostatních existujících systémů jsou podpůrné funkce knihovny a rozhraní pro komunikaci s hardwarem třetích stran. Tyto vlastnosti opět vycházejí z otevřenosti implementované knihovny a umožňují tvorbu komplexních řešení přímo na míru požadavkům zákazníka. Vše vychází z konceptu úpravy aplikace podle zákaznického prostředí, což je obrácený přístup než ten, který nabízí komerční systémy. To s sebou nese výhody větší konkurenceschopnosti výsledných aplikací, avšak také nevýhody zejména v časové náročnosti těchto úprav.

8 Implementace vzorových aplikací

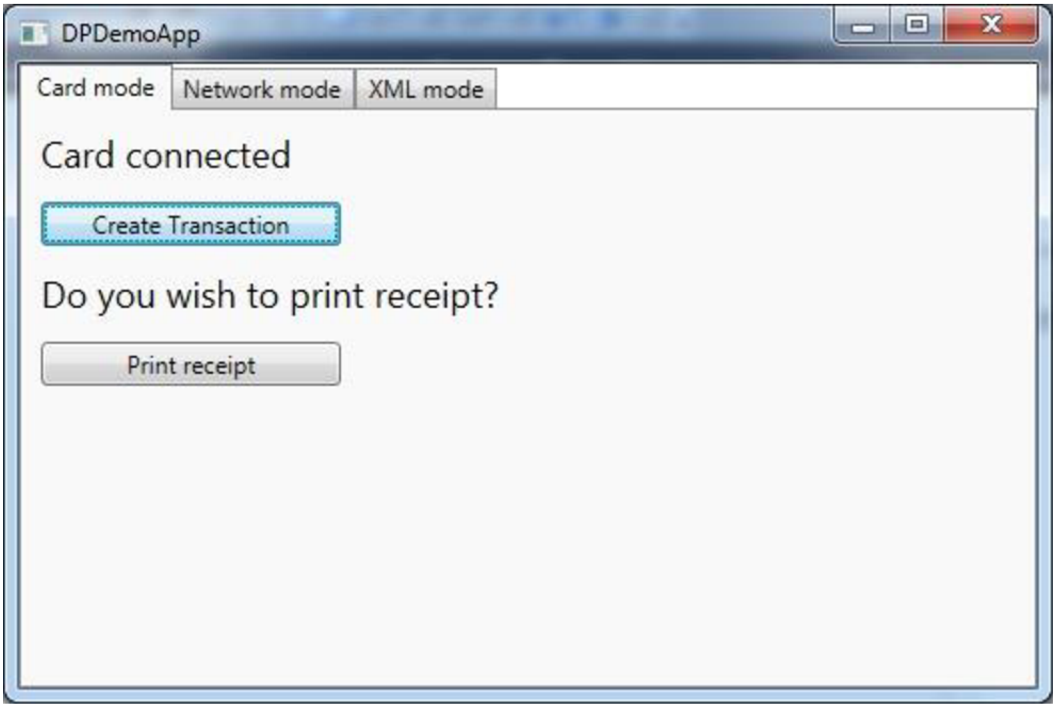
8.1 Demo aplikace funkčnosti knihovny

Tato aplikace slouží pro demonstraci vlastností a funkcí knihovny a v reálném prostředí není nikde používána. Obsahuje jednoduchá grafické uživatelské rozhraní vytvořené ve Windows Presenation Foundation a spolupráci se systémovými tiskárnami a čtečkou bezdotykových NFC karet ACR122U USB NFC Reader.

Aplikace ve svém základním módu čeká na asynchronní událost, která je vyvolána přiložením bezdotykové NFC karty ke čtečce. Po této události zobrazí informaci o přiložené kartě a nabídne vytvoření jednoduchého platebního požadavku. Tento požadavek poté uloží do lokální XML databáze a nabídne možnost vytisknutí jednoduché stvrzenky. Grafické uživatelské rozhraní této části je k nahlédnutí na obrázku č.25.

V testovacím módu pak obsahuje ještě upravovatelné textové pole, ve kterém je možné vyzkoušet správnou funkci připojeného vstupního znakového zařízení, a jednoduchý internetový prohlížeč, ve kterém je možné vyzkoušet funkci třídy *NetworkManager*. Grafické uživatelské rozhraní této části je k nahlédnutí na obrázku č.26.

Poslední záložka obsahuje jednoduchý textový editor, který zobrazuje XML souboru vygenerovaný transakcemi vytvořenými v základním módu aplikace.



Obrázek č.25: Ukázková demo aplikace v základním módu

Obrázek č.26: Ukázková demo aplikace v testovacím módu

8.2 Informační kiosek města Tábor

Na této aplikaci jsem spolupracoval se společností Ki-Wi Digital, s.r.o. Cílem bylo vytvořit aplikaci, která bude sloužit pro interaktivní zobrazování památek ve městě Tábor a jeho okolí. Využití knihovny vytvářené v rámci této diplomové práce mělo sloužit k nákupu vstupenek do vybraných památek přes tuto aplikaci prostřednictvím bezdotykových platebních karet a vlastních NFC karet města Tábor, které pracují na principu dobíjeného kreditu. Tento projekt byl nakonec ukončen ve fázi prototypu, ale pro účely diplomové práce jsem jej v omezené míře dokončil. Oproti původním plánům aplikace neumožňuje platbu pomocí bezdotykových platebních karet a namísto NFC karet města Tábor využívá vlastní NFC karty s vlastní simulací kreditového zůstatku.

Jako základ aplikace byl použit software Ki-Wi Platform, který pro zmiňovanou společnost vyvíjím. Jedná se o modulovou klientskou část systému server-klient této firmy. Zákazník si volí vzhled a funkčnost svého zařízení pomocí serverové webové aplikace s podporou cloudového úložiště, aplikace Ki-Wi Platform se pak řídí předpisy, které jí zašle server, a zobrazuje odpovídající obsah. V rámci této diplomové práce jsem vytvořil modul Ki-Wi Kiosk, který slouží pro prezentaci

webových aplikací a zobrazování různých interaktivních katalogů. Právě na tento modul jsem navazoval svoji knihovnu.

Pro účely prezentace byla vytvořena webová aplikace, která obsahuje kategorizovanou nabídku památek a kulturních akcí ve městě Tábor a jeho okolí. Každá tato položka (kromě akcí týkajících se celého regionu) je zároveň označena na mapě, ze které je možné taktéž k jednotlivým položkám přistupovat. Po výběru položky se zobrazí její detailní informace, zejména pak popis události či památky, otevírací doba, cena vstupného a možnost nakoupit vstupenky. Při volbě nákupu vstupenek je vlastním formátem URL odkazu předána kiosku informace o tomto požadavku. Kiosek je nastavený na zpracování parametrů tohoto URL odkazu a zobrazí adekvátně vygenerované grafické menu, ve kterém je možné navolit počet vstupenek pro jednotlivé vstupenkové kategorie (typicky dospělí a děti). Uživatel je poté vyzván k přiložení NFC karty. Po úspěšném ověření karty a zůstatku, který je na ní k dispozici, se vygeneruje XAML šablona se vstupenkou, která je poté vytisknuta. Identifikace vstupenky na místě určení pak probíhá pomocí čárového kódu, pod kterým je tato vstupenka uložena v systému.

Obrázek č.27: Náhled aplikace pro město Tábor

8.3 Interaktivní aplikace pro společnost Tescoma

Další spoluprácí se společností Ki-Wi Digital, s.r.o. byl vývoj interaktivní aplikace s cílenou podporou prodeje. Tato aplikace byla vyvíjena na míru požadavkům zákazníka zmiňované společnosti, a to firmy Tescoma, s.r.o. Tato společnost se zabývá výrobou a prodejem kuchyňských potřeb a doplňků. Cílem této aplikace byla elektronická prezentace jejich výrobků, zejména pak

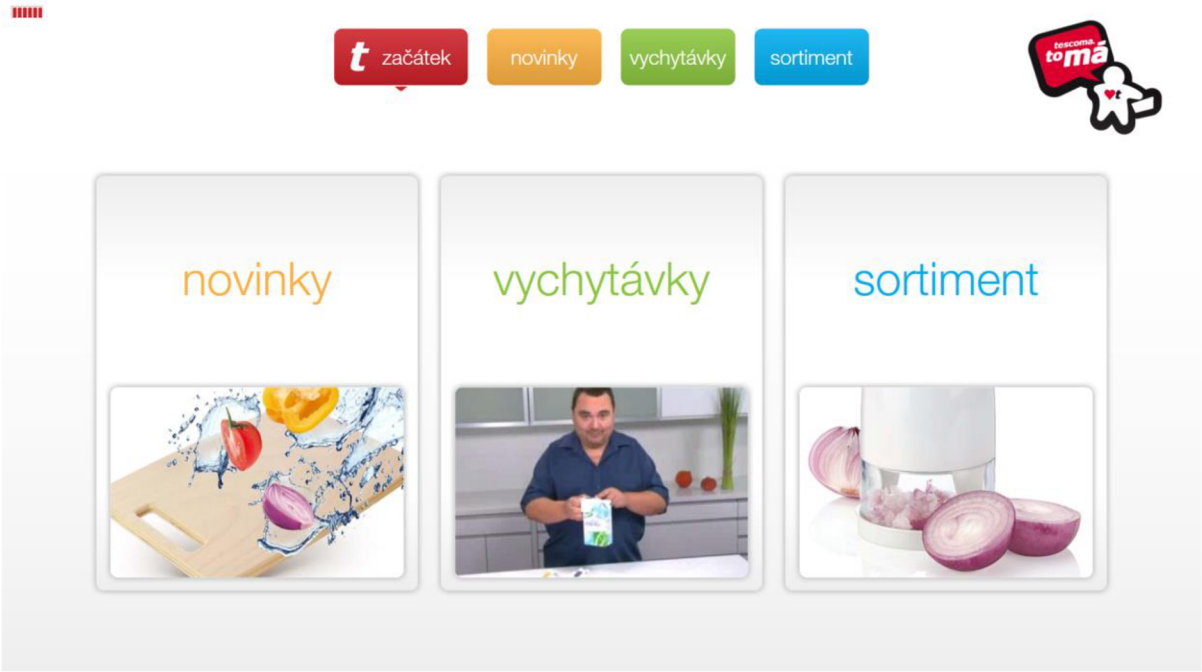
novinek, a dále podpora prodeje na základě přiložených čárových kódů výrobků a zákaznických karet této společnosti.

Pro realizací tohoto požadavku byla využita aplikace Ki-Wi Interactive. Aplikace obsahuje databázi vybraných výrobků společnosti Tescoma, s.r.o., kde u každého výrobku je evidován textový popis a dále instruktážní video. Na prodejně jsou pak umístěny počítače a k nim připojené dotykové obrazovky a dále mají zdejší prodavači k dispozici tablety, se kterými mohou asistovat zákazníkům při výběru produktů. Zákazník si na těchto obrazovkách může selektivně zvolit výrobek pro který mu bude přehráno instruktážní video. Alternativně je možné výrobek navolit jeho fyzickým přenosem k zařízení a přečtení jeho čárového kódu pomocí čtečky těchto kódů, která je u počítačové stanice k dispozici.

Aplikace je opět realizována ve frameworku Microsoft .NET verze 4.0, v programovacím jazyku C# a v grafickém systému Windows Presentation Foundation. Využívá distribuční a updatový systém Microsoft ClickOnce, který je v tomto případě nastaven na automatické udržování aplikace v její nejnovější dostupné verzi. Její původní verze, která umožňuje vybírání výrobků selektivně přes dotykovou obrazovku a přes čárové kódy, je nasazena na vybraných prodejnách společnosti Tescoma, s.r.o.

V rámci této diplomové práce vznikla nová verze aplikace Ki-Wi Interactive, která rozšiřuje její možnosti o využití bezdotykových NFC klientských karet. Kromě určitých technických vylepšení aplikace, jako např. možnosti mít k jednomu výrobků přiřazených více videí, byla implementována možnost využití klientských karet za pomoci knihovny realizované v této práci. Po přiložení klientské karty ke čtečce bezdotykových NFC karet je zákazníkovi zobrazen seznam produktů, které již nakoupil, a k nim přiřazená instruktážní videa. Aplikace dále zákazníka upozorní na případná nová videa, která byla k těmto výrobkům přiřazena v době od jeho posledního přihlášení do systému, a dále ho textově upozorní na případné propagační či slevové akce, které by ho mohly zajímat. Poslední vlastností tohoto rozšíření je doporučení dalších výrobků na základě těch, které si zákazník již pořídil.

Funkčnost doporučování výrobků a slevových akcí je realizována externím systémem společnosti Tescoma, s.r.o., aplikace Ki-Wi Interactive jej pouze graficky prezentuje. Tato aplikace je nasazena v reálném provozu, v době psaní tohoto textu se jednalo o testovací provoz na 2 pobočkách společnosti Tescoma, s.r.o. v Praze.



Obrázek č.28: Náhled aplikace Ki-Wi Interactive pro společnost Tescoma, s.r.o. ve verzi pro tablet

9 Závěr

9.1 Zhodnocení práce a vlastností vytvořené knihovny

V této diplomové práci jsem rozebral problematiku návrhu a implementace systému pro integraci platebních terminálů. Spolupracoval jsem na ní se společností Ki-Wi Digital, s.r.o. a bankovní institucí Komerční banka, a.s. Vytvořil jsem 3 ukázkové aplikace, z nichž jedna je již v době psaní tohoto textu nasazená v praxi u zákazníka společnosti Ki-Wi Digital, s.r.o.

Platební terminály, platební karty a elektronické platební transakce jsem v úvodu práce rozebral nejdříve teoreticky. Zaměřil jsem se zejména na specifika bezkontaktních karet NFC. Zhodnotil jsem také legislativní situaci v České republice týkající se elektronických platebních operací.

V teoretické části práce jsem dále popsal průběh procesu platebních transakcí, rozebral jsem síťové vlastnosti těchto operací a jejich zabezpečení. Důkladně jsem analyzoval platební rozhraní pro komunikaci s platební kartou a bankovní institucí, které jsem poté při implementaci využil.

Ve spolupráci se společností Ki-Wi Digital, s.r.o. jsem vytvořil specifikaci požadavků na výslednou aplikaci a poté jsem provedl návrh její implementace. K tomu jsem využil prostředků jazyka UML a do práce jsem přiložil diagramy, které jsou výstupem této činnosti. Diskutoval jsem problematiku integrace této aplikace na mobilní telefony a potřebné úpravy, které bude při tomto procesu zapotřebí provést. Dále jsem popsal technologie a vývojové nástroje, které jsem v průběhu tvorby diplomové práce využil.

Navrženou aplikaci jsem poté implementoval a v textu práce jsem popsal její jednotlivé součásti a postup jejich tvorby. Zaměřil jsem se na popis vytvořených tříd a funkcionality, kterou vytvořená knihovna poskytuje. Popsal jsem způsob navázání na bankovní systémy a komunikaci s hardwarem třetích stran. Uvedl jsem způsob zaznamenávání proběhlých transakcí a využití podpůrných funkcí, které jsem do knihovny implementoval nad rámec zadání. Popsal jsem průběh testování a hotovou knihovnu jsem porovnal s existujícími systémy pro integraci platebních terminálů. Dále jsem vytvořil 3 ukázkové aplikace, které jsem k práci přiložil a v textu jsem je v patřičných kapitolách popsal.

Vytvořená knihovna umožňuje připojení hardwarových zařízení, které jsou zapotřebí pro realizaci elektronických transakcí, a poskytuje logiku pro práci s těmito zařízeními. Zaměřuje se jak na vlastní zákaznické bezdotykové NFC karty, tak na karty platební. Zvýšenou pozornost jsem věnoval rozšiřitelnosti knihovny vzhledem k jejímu nasazení do reálného provozu a neúplné

standardizaci některých procesů v tomto odvětví. Knihovna je dále připravena pro napojení na bankovní transakční programové rozhraní bankovních institucí. Nad rámec zadání jsem do knihovny implementoval podpůrné funkce pro tisk stvrzenek a jiných dokumentů, podporu pro vstupní zařízení které nepracují na principech klávesnice a rozhraní pro komunikaci s hardwarem třetích stran.

V rámci diplomové práce jsem vytvořil ukázkovou aplikaci, která demonstruje vlastnosti a funkce knihovny. Dále jsem ve spolupráci se společností Ki-Wi Digital, s.r.o. vytvořil modul Ki-Wi Kiosk do jejich klientské aplikace Ki-Wi Platform a prototyp realizace tohoto modulu pro město Tábor, který slouží pro propagaci památek a společenských akcí v tomto městě a jeho okolí. Poslední přiloženou aplikací je nová verze softwaru Ki-Wi Interactive, kterou jsem za pomocí vyvinuté knihovny rozšířil. Tato verze slouží pro přímou podporu prodeje v kamenných obchodech společnosti Tescoma, s.r.o. a v době psaní tohoto textu je již nasazena v ostrém provozu ve 2 pobočkách této společnosti v Praze.

9.2 Rozšíření knihovny a její aplikace

S hotovou knihovnou nastává také otázka jejího budoucího využití, které budeme dále vyhledávat ve spolupráci se společností Ki-Wi Digital, s.r.o. Kromě využití v interaktivní aplikaci pro podporu prodeje, které je již uvedeno do praxe v některých pobočkách společnosti Tescoma, s.r.o. se chystáme využít i prototypové aplikace pro propagaci památek, která byla taktéž v rámci této práce vytvořena.

Rozšíření funkčních možností knihovny se týká zejména jejího napojení na bankovní systémy finančních institucí. Dále je vhodné postupně zvyšovat množství podporovaných hardwarových zařízení, zejména pak čteček karet. Knihovna je napsána rozšiřitelně s pomocí sady programových rozhraní a abstraktních tříd, pro které je možné jednoduše dopsat konkrétní implementaci.

Zajímavým odvětvím využití této knihovny je spolupráce s hardwarem třetích stran. Zatímco aktuální aplikace spolupracují s knihovnou na softwarové úrovni, společnost Ki-Wi Digital, s.r.o. se na základě aktuálního požadavku bude v nejbližších dnech zabývat možností jejího využití pro spolupráci s hardwarem. Konkrétně se bude jednat o možnost platby pomocí bezdotykových NFC karet za nákup uskutečněný v automatu na kávu. Knihovna je pro tuto spolupráci již připravena díky navrženému rozhraní, implementaci tohoto rozhraní však bude zapotřebí přizpůsobit konkrétnímu hardwaru.

10 Použitá literatura

[1] LAUDON, K.C. a C. G. TRAVER. *E-Commerce business, technology, society*. Boston: Addison-Wesley, 2001. ISBN 0-201-74815-0.

[2] RANKL, W. a W. EFFING. *Smart card handbook*. Chichester: John Wiley & Sons Ltd, 2003. ISBN 0-470-85668-8.

[3] JURGENSEN, T. a S. GUTHERY. *Smart cards : the developer's toolkit*. New Jersey: Prentice Hall PTR, 2002. ISBN 0-13-093730-4.

[4] PARET, D. *RFID and contactless smart card apllications*. Chichester: John Wiley & Sons Ltd, 2005. ISBN 0-470-01195-5.

[5] *EMV Integrated Circuit Card Specifications for Payment Systems: Book 1 - Application Independent ICC to Terminal Interface Requirements* [online]. 2011, 2014-01-19 [cit. 2014-01-06]. Dostupné z: http://www.emvco.com/download_agreement.aspx?id=652

[6] *EMV Integrated Circuit Card Specifications for Payment Systems: Book 2 – Security and Key Management* [online]. 2011, 2014-01-19 [cit. 2014-01-06]. Dostupné z: http://www.emvco.com/download_agreement.aspx?id=653

[7] *EMV Integrated Circuit Card Specifications for Payment Systems: Book 3 - Application Specification* [online]. 2011, 2014-01-19 [cit. 2014-01-06]. Dostupné z: http://www.emvco.com/download_agreement.aspx?id=654

[8] Právní předpisy - Česká národní banka. [online]. 2014-01-19 [cit. 2014-01-11]. Dostupné z: https://www.cnb.cz/cs/dohled_financni_trh/legislativni_zakladna/platebni_instituce_a_institu ce_el_penez/pravni_predpisy.html

[9] PETRICEK, T. a J. SKEET. *Real-world functional programming: with examples in F# and C#*. Greenwich: Manning, 2010. ISBN 19-339-8892-4.

[10] MACDONALD, M., A. FREEMAN a M. SZPUSZTA. *ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně*. Překlad J. Pokorný. Brno: Zoner Press, 2011. ISBN 978-80-7413-131-8.

[11] ARLOW, J.a I. NEUSTADT. *UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru*. Brno: Computer Press, 2003. ISBN 80-722-6947-X.

[12] BURGET, R. a D. ZEMAN. *Tvorba webových stránek, studijní opora* [online]. 2006 [cit. 2014-05-16]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ITW-IT/texts/opora_itw_061020.pdf

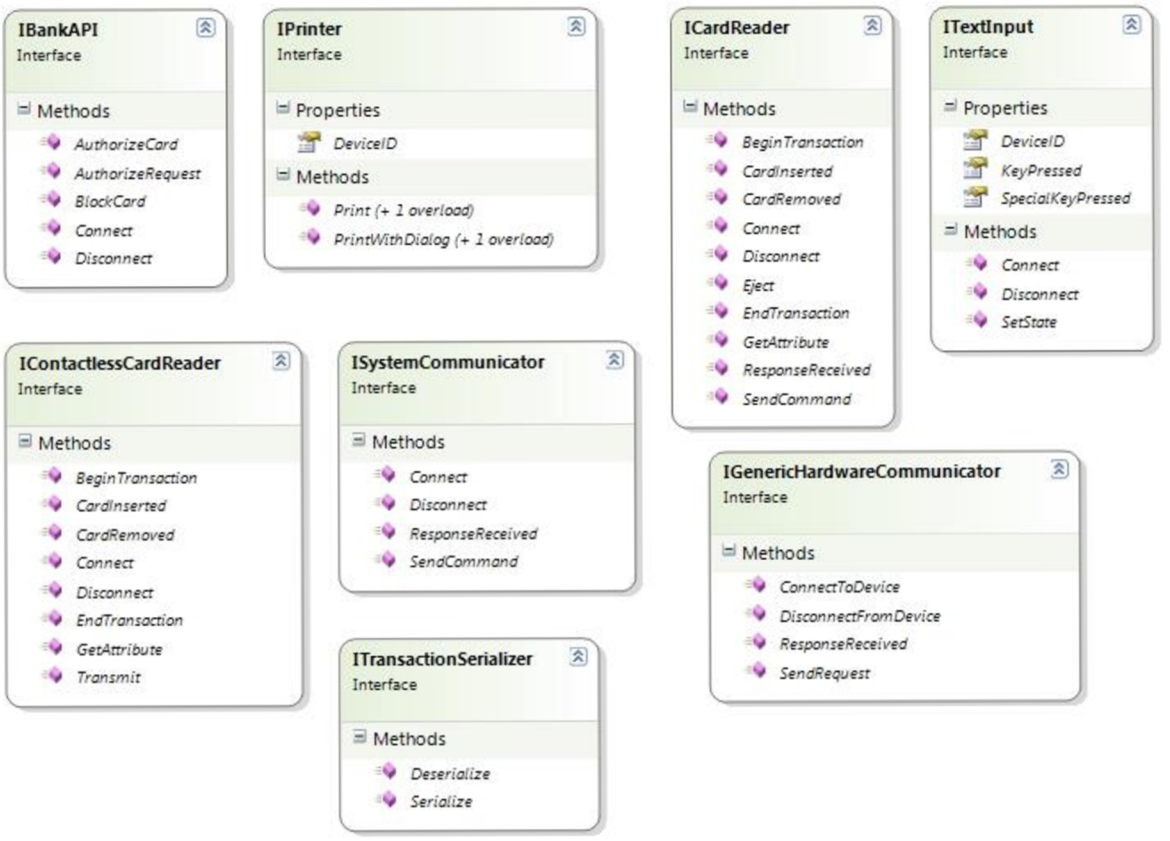
[13] MSDN. *LINQ (Language-Integrated Query)* [online]. 2013. [cit. 2014-05-18]. Dostupné z: http://msdn.microsoft.com/en-us/library/bb397926.aspx

[14] BISHOP, J. *C#: návrhové vzory*. Brno: Zoner Press, 2010. ISBN 978-80-7413-076-2.

11 Přílohy

11.1 Diagram tříd

11.2 Diagram rozhraní



11.3 Obsah přiloženého CD

Přiložené CD obsahuje:

- zkompilovanou knihovnu pro prostředí .NET,
- zdrojové kódy knihovny,
- zdrojové kódy demo aplikací,
- instalátory demo aplikací pro operační systémy Windows,
- elektronickou verzi této práce.