Středoškolská technika 2013

Setkání a prezentace prací středoškolských studentů na ČVUT

BEZPEČNÉ OVĚŘENÍ OSOB POMOCÍ MODERNÍ KRYPTOGRAFIE

Pavel Fojtík, Jaromír Kuchyňka

Purkyňovo gymnázium Strážnice

Masarykova 379, Strážnice

Bezpečné ověření osob pomocí moderní kryptografie

Safe verification of persons by using modern cryptography

Autoři:	Pavel Fojtík Jaromír Kuchyňka
Škola:	Purkyňovo gymnázium, Strážnice Masarykova 379 696 62 Strážnice
Konzultant:	Ing. Jan Hajný, PhD.

Strážnice 2013

Prohlášení

Prohlašujeme, že jsme svou práci vypracovali samostatně, použili jsme pouze podklady (literaturu, SW atd.) uvedené v přiloženém seznamu a postup při zpracování a dalším nakládání s prací je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V …… dne ……

podpisy: …….
Pavel Fojtík

…….
Jaromír Kuchyňka

Poděkování

Rádi bychom touto formou vyjádřili velkou vděčnost Ing. Janu Hajnému za jeho odborné vedení, obětavou pomoc, velkou trpělivost a především za nezanedbatelné množství času, které se rozhodl nám věnovat. Bez jeho pomoci by tato práce nikdy nespatřila světlo světa.

Dále bychom také chtěli poděkovat Ing. Lukáši Malinovi za jeho důležité rady a mnoho podnětných připomínek, díky nimž jsme se mnohokrát pohnuli z místa, když už jsme nevěděli jak dále.

Nesmíme zapomenout ani na vedení a pedagogy Purkyňova gymnázia Strážnice, kteří nás v naší práci plně podporovali a projevili nejvyšší možné pochopení pro naši aktivitu. Především bychom rádi vyzdvihli pomoc RNDr. Jany Hálkové.

A v neposlední řadě patří velký dík Jihomoravskému kraji a Jihomoravskému centru pro mezinárodní mobilitu, bez jejichž finanční podpory a pomoci by naše práce byla o mnoho hůře uskutečnitelnou.

ABSTRAKT

Informace a elektronické systémy hrají v dnešním světě stále důležitější roli. Myšlenky převedené do podoby nul a jedniček jsou zapisovány, šířeny, zpeněžovány, a proto také v některých případech, bohužel, kradeny. Aby se zabránilo takovémuto úniku informací do nesprávných rukou, dochází k navrhování a tvorbě bezpečnostních systémů. Ty bývají často velmi nákladné, složité pro uživatele a také nespolehlivé.

Právě složitost a nespolehlivost jsou řešeny v této práci. Hlavní důraz je kladen na systémy, které k ověření identity vyžadují vlastnictví určitého předmětu (karty, čipu apod.). Důvodem je jejich nízká bezpečnost. K vniknutí do systému stačí předmět odcizit. Proces ověření identity je tak složitý a nebezpečný.

Za použití systému vyvinutého v rámci projektu uskutečníme celý proces autentizace mnohem bezpečnějším, užitím aktivního a výpočetně silného zařízení. Tímto jej zároveň i zpříjemníme pro uživatele, který již nebude nadále muset vlastnit nadbytečné pomůcky. Nahradíme je předmětem každodenní potřeby – mobilním telefonem, přesněji smartphonem s operačním systémem Android.

Pro samotnou realizaci jsme se rozhodli využít výhod poskytovaných QR kódy, jež jsou v současné době velmi často používané a jejichž význam neustále nabývá na síle. Tyto kódy zajistí uživatelsky přívětivý přístup k systému i při využití kryptograficky silných, bezpečných protokolů autentizace.

Konečným výsledkem naší práce je program, na němž demonstrujeme rozdíly mezi stávajícími bezpečnostními systémy a tím námi vytvořeným, čímž zvýšíme informovanost veřejnosti o možném zlepšení zabezpečení a rovnou nabídneme řešení v podobě našeho produktu.

Klíčová slova

QR kód; kryptografie; Android; smartphone; bezpečnostní systémy

ABSTRACT

Information and electronic systems are nowadays getting more and more important. Ideas transformed into zeros and ones are recorded, spread, sold and because of that sometimes unfortunately stolen. To prevent leak of information to wrong person security systems are designed and built. However those are mostly expensive, unreliable and bring inconveniences to their users.

The complexity and uncertainty are solved in this work. The main emphasis is put on systems, which needs the person to own one certain object (card, chip, etc.) for verification. The reason is their low security level, because system intrusion is simply possible by stealing object. Process of identity verification is there for complicated and dangerous.

By using system developed within this project, we will make whole process of authentication safer, using active and computationally powerful device. It also has benefits for users. They don't need to carry superfluous utilities anymore. We replace them with object of everyday use - mobile phone, precisely smartphone with OS Android.

For very own realization we have chosen to utilize benefits of QR codes, which are currently very often used and whose importance significantly grows. Those codes provide user-friendly access to system even with use of cryptographically strong, safe authentication protocols.

Final result of our work is program, where we will demonstrate differences between already existing systems and the one created by us. With that we will inform the public of possible ways of improving security systems and offer solution in the form of our product.

Key words

QR code; cryptography; Android; smartphone; safety systems

Obsah

Úvod

Ochrana dat je jednou z nejdůležitějších, snad dokonce esenciální, oblastí v informatice, která jako taková má vedoucí úlohu v jednoduše všem. I přes to je veřejností velmi přehlížena a zájem o ní je vyhrazen několika málo specialistům. Z toho důvodu jsme se v naší práci soustředili na to přiblížit ji běžnému člověku. Za pomocí nových poznatků v oblasti moderní kryptografie jsme navrhnuli a vytvořili plně funkční a použitelný program, který slouží jako zabezpečovací systém. Díky užití transparentního způsobu je celý proces velmi názorný a snadno pochopitelný. Stejný princip jsme se snažili aplikovat i na textovou část a učinit ji co nejsrozumitelnější.

Svůj výklad musíme započít u vědního oboru nazvaného kryptografie, neboť bez ní by téměř jakákoliv počítačová ochrana nebyla použitelná. Představíme vám ji tedy jako celek. Zvláštní důraz dáme na to, o co se v ní vlastně jedná. Rozčleníme kryptografii do dvou hlavních směrů a každý detailně popíšeme. Následně tuto část dokončíme popisem postupů dříve užívaných i těch, které zajišťují vaši bezpečnost v současnosti.

Na kryptografii navážeme informacemi o termínu QR kód a všem s ním spojeném. Začneme jednoduchým představením a rozšifrováním názvu, ke kterému přidáme i několik techničtějších informací, jež jsou podány velmi přijatelnou formou i pro laika. V konečné části už jen shrneme všeobecné poznatky a nastíníme možnou budoucnost QR kódů.

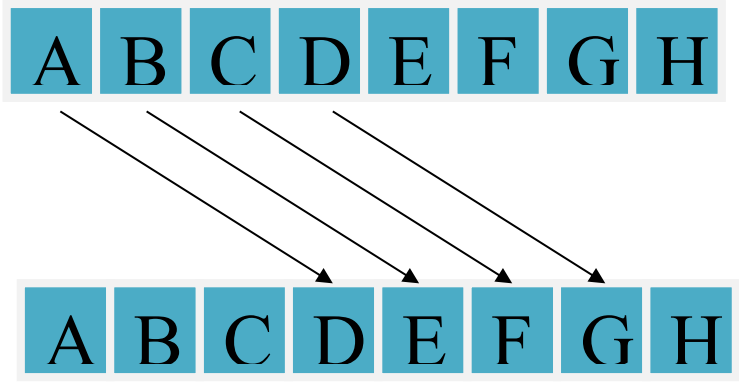
Další oddíl je uvedením do světa smartphonů, především těch vybavených OS Android. Detailně a krok za krokem vám ukážeme možné řešení programování pro smartphony a zároveň s tím přejdeme k počátkům OS Android. Navážeme základními informacemi o systému a okamžitě poté vás začneme provádět vývojovým prostředím Eclipse užívaným pro tvorbu aplikací na zařízení s OS Android. Podrobným návodem vám popíšeme jak stáhnout všechen potřebný software, který poté díky ještě podrobnějšímu návodu správně nainstalujeme, nastavíme a uvedeme do provozu. Celý tento proces zakončíme vytvořením první aplikace. Na ní demonstrujeme kompilaci, emulaci a především nahrání vaší aplikace do reálného zařízení.

A konečně v posledním úseku naleznete podrobný popis námi vytvořeného programu. Jsou zde rozepsány problémy, jež jsme museli vyřešit, aby bylo možné začít programovat, a spolu s nimi se samozřejmě vyskytují návrh a struktura aplikace. Pomocí nich přejdeme k užitým postupům a celkové realizaci programového řešení, doplněné ukázkami zdrojového kódu. Všechno je zakončeno analýzou našeho programu, návrhem možných vylepšení a nastíněním budoucnost našeho výtvoru.

1 Kryptografie

Utajování a snaha o bezpečnou komunikaci patří k lidem snad už od počátku věků, a je tudíž jakousi jejich přirozeností. Již starověké národy používaly mnoho způsobů, počínaje klasickou Caesarovou šifrou (posunutí znaků v abecedě o *n* znaků, obrázek 1) až po některé absurdní užívané v Řecku (ostříhanému otrokovi napsali vzkaz na hlavu a počkali, než vlasy dorostou).[1] A historie také nesčetněkrát prokázala, že význam kryptografie a kryptologie může být klíčový. Vyzrazení nebo naopak úspěšné utajení informací téměř jakéhokoliv druhu (válečné plány, přípravy atentátu, politické intriky, …) tvořilo dějiny.

A proto také v současnosti tento obor nabírá stále více a více na důležitosti, neboť velmi úzce souvisí s informatikou, která se bez něj takřka neobejde a která je hybnou silou této doby.



Obrázek 1 – Caesarova šifra, posunutí znaku o tři pozice

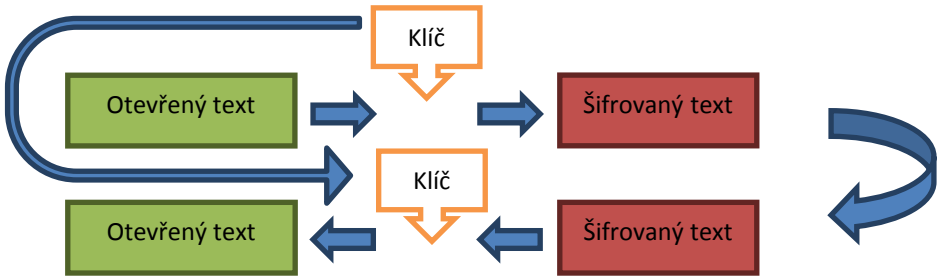
1.1 Základní pojmy

Kryptografie je věda zabývající se způsoby utajování textu zpráv, především jejich převodem do podoby, která je čitelná pouze se speciální znalostí. Někdy je tento pojem zaměňován s kryptologií, do níž patří všechno spojené s šiframi, hlavně tedy kryptoanalýza (luštění šifrovaných zpráv) a již zmiňovaná kryptografie.

Pro pochopitelnost následujícího textu je také třeba zmínit pár elementárních pojmů, bez nichž se v kryptografii neobejdeme. Začneme odesílatelem**,** tudíž člověkem, který chce poslat zprávu. Pro bezpečnost vybaví svou zprávu šifrou, která označuje kryptografický algoritmus, jenž převádí čitelnou zprávu neboli otevřený text na její nečitelnou podobu neboli šifrový text a až ten je poté poslán příjemci**,** který využije klíče k převedení šifrového textu na zprávu.[2]

1.2 Symetrická kryptografie

U symetrické kryptografie se k šifrování i dešifrování textu používá jediný klíč.[3] Případnému útočníkovi tedy stačí zjistit pouze tento klíč a šifra je prolomena. Tato nepříjemnost nutí odesílatele i příjemce tajné zprávy k dohodnutí se na tajném klíči, který musí oba sdílet. Nezanedbatelnou výhodou naproti tomu je nízká výpočetní náročnost, čili nároky na technické vybavení a čas nejsou tak vysoké.



Obrázek 2 – Obecné schéma symetrické kryptografie

Symetrické šifry můžeme, podle toho jakým způsobem zpracovávají otevřený text, rozdělit na dva druhy:

- Proudové zpracovávající text po jednotlivých bitech.
- Blokové, které rozdělí text na stejně velké bloky a v případě potřeby vhodným způsobem doplní poslední blok na velikost ostatních.

1.2.1 DES

Data Encryption Standard je šifrovací algoritmus vyvinutý Národním úřadem pro standardizaci roku 1975. Patří mezi blokové šifry. Od roku 1977 sloužila k šifrování dat v nevojenských státních organizacích v USA a posléze pronikla i do komerční sféry. Celkem záhy ale začala být kritizována pro nedostatečnou velikost klíče (používá 64 bitů z toho 8 kontrolních a jen 56 efektivních) a neodolnost proti útoku hrubou silou.[4]

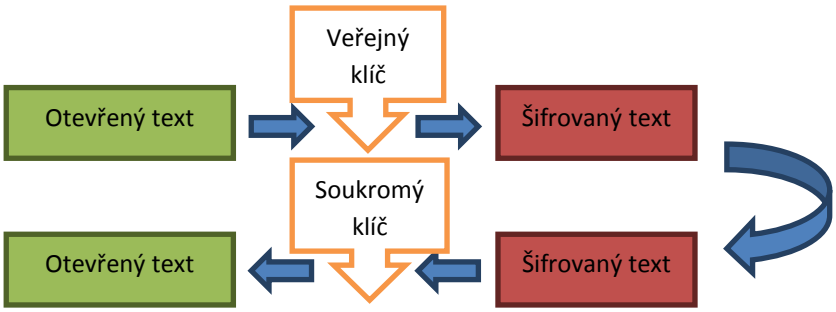
1.2.2 AES

Advanced Encryption Standard je bloková šifra (o velikosti bloku 128 bitů), která pro šifrování i dešifrování využívá klíč s velikostí 128,192 anebo 256 bitů.[4] Nahradila dříve užívanou šifru DES. Je velmi rychlá a snadno implementovatelná. V současnosti se používá například k zabezpečení bezdrátové Wi-Fi sítě.

1.3 Asymetrická kryptografie

U asymetrické kryptografie (kryptografie s veřejným klíčem) se pro šifrování i dešifrování používají rozdílné klíče. Šifrovací klíč v asymetrické kryptografii je tvořen ze dvou částí: jedna část se používá k šifrování zpráv (příjemce tuto první část nemusí nutně znát), druhá pro dešifrování (ve většině případů ji odesílatel šifrované zprávy nezná). Z toho vyplývá, že není zapotřebí, aby odesílatel zprávy sdílel tajemství s příjemcem. Ruší se tak nutnost výměny klíčů, což je hlavní výhodou asymetrické kryptografie.

V praxi se v asymetrické kryptografii nejčastěji využívá tzv. veřejný (šifrovací) a soukromý (dešifrovací) klíč. Šifrovací klíč příjemce zprávy dá volně k dispozici, a kdokoli mu chce zaslat zprávu, ji jím může zašifrovat. Dešifrovací klíč příjemce udržuje v tajnosti a využívá jej pouze ke čtení šifrovaného textu. Pro funkčnost šifry je nezbytné, aby šifrovací klíč s dešifrovacím klíčem spolu byly matematicky svázány, avšak k tomu, aby byla šifra užitečná, nesmí jít (přinejhorším alespoň velmi obtížně) ze znalosti veřejného klíče určit klíč soukromý.[3] Toto všechno má ovšem za následek velkou nevýhodu této šifry, a totiž její velkou náročnost na výpočet.



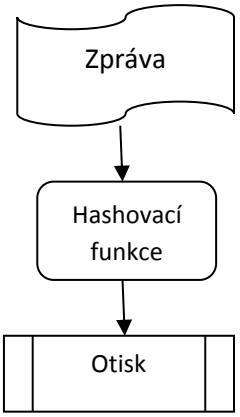
Obrázek 3 – Obecné schéma asymetrické kryptografie

1.3.1 RSA

Algoritmus RSA je první šifrovací systém vytvořený na základě uveřejnění myšlenky asymetrické kryptografie. Lze jej velmi pravděpodobně označit za nejpoužívanější šifrovací algoritmus v oblasti bezpečnosti komunikace. RSA je v základech postaven na nenalezení dostatečně rychlého algoritmu pro rozklad hodně velkých čísel na prvočísla. Připojí-li se k tomu velký klíč, lze RSA považovat za bezpečný. Tím ovšem vzniká podstatná nevýhoda, kterou je velmi nízká rychlost šifrování velkých objemů dat. Algoritmus RSA se v průběhu času lehce měnil, avšak s těmito úpravami se používá dodnes a jeho pole působnosti je opravdu široké – od mobilních telefonů, přes bankomaty až k elektronickým podpisům atd.

1.4 Hashovací funkce

Je matematická funkce (přesněji algoritmus) sloužící k převodu dat do (poměrně) malého čísla. Výsledek hashovací funkce se nejčastěji označuje otisk či hash a jeho velikost je vždy konstantní. Použití hashovacích funkcí v počítačovém světě je opravdu velmi mnoho. Nejvíce se užívají k naleznutí chyb, pro implementaci tabulek a samozřejmě také v kryptografii. Všechna tato užití vyžadují, aby hashovací funkce poskytovala vhodnou náhodnou distribuci výsledků, rovnoměrné pokrytí oboru hodnot a aby nepatrná změna na vstupných datech vedla k výrazné změně na výstupním hashi. U kryptografie sice odpadá důraz na rychlost funkce, ovšem o to větší požadavky jsou kladeny na kryptografické vlastnosti.



Obrázek 4 – Hashovací funkce

1.4.1 MD5

Message-Digest algorithm tvoří skupinu hashovacích funkcí. Algoritmus MD5 našel využití v mnoha různých programech (kontrola celistvosti souborů, ukládání hesel…). Výsledný otisk vytvořený pomocí MD5 má vždy velikost 128 bitů. Hned po svém uvedení do komerční sféry roku 1991 dosáhl značné obliby a ve velké míře byl používán ještě před pár lety. V roce 1996 však byla zjištěna chyba v logice MD5, a i když nedošlo k prolomení šifry, ale pouze k nalezení kolizí, kryptologové se postupně začali klonit k jiným algoritmům (například SHA). Později ovšem došlo k objevení mnohem závažnějších nedostatků a užívání MD5 spěje ke konci.

1.4.2 SHA

Secure Hash Algorithm patří mezi vylepšené hashovací funkce. Do SHA řadíme pět algoritmů. Nejnovější čtyři varianty jsou společně nazývány SHA-2. Nejvíce se užívají verze SHA-256 a SHA-512 například pro datové schránky. SHA coby nástupce MD5, plní jeho dřívější funkci a navíc slouží i v některých protokolech a aplikacích.

1.5 Digitální podpis

Digitální podpis je zpráva zašifrovaná pomocí tajného klíče odesílatele. Práce s celou zprávou by ale kladla zbytečně velké nároky na výkon, tudíž se zpracovává pouze její hash.[5] Vzhledem k tomu, že jediným vlastníkem klíče je (nebo by přinejmenším měl být) odesílatel, původ zprávy lze zaručit. K ověření digitálního podpisu poté stačí jen veřejný klíč autora. Jde o to, že pokud hash ze zprávy je totožný s hashem, který vzniknul dešifrováním hashe elektronického podpisu veřejným klíčem, pak autorem zprávy musí být odesílatel.

1.6 Autentizace

Autentizace je proces sloužící k určení osoby přistupující k systému (počítač, terminál…). Cílem je, aby systém dokázal určit, kdo se s ním pokouší komunikovat. Autentizace patří k základním bezpečnostním opatřením.

Používají se tyto základní metody pro zjištění identity:

- uživatel něco zná (uživatelské jméno a heslo, PIN…)
- uživatel něco vlastní (klíč, čipová karta, USB klíč…)
- uživatel je nějaký (biometrické vlastnosti: otisk prstu, snímek oka, držení těla…)
- uživatel něco umí (opsat text z obrázku…)
- důvěryhodný prostředník

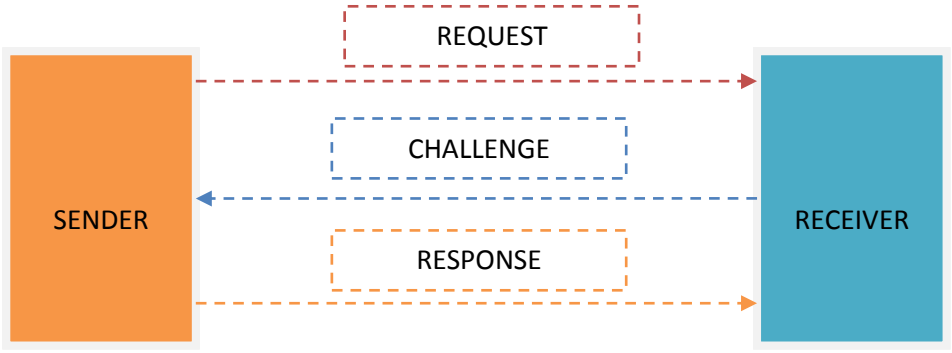
Obrázek 5 – Klasická autentizace

1.7 Protokol

Zjednodušeně řečeno protokol z hlediska informatiky je domluvou dvou stran na způsobu, jakým si budou vyměňovat informace. Protokol z hlediska kryptografie je pak to, jakým způsobem a co si budou odesílatel s příjemcem sdělovat. Hlavně zahrnuje úkony, které je nutno vykonat, aby mohlo dojít k výměně informací. Protokolem je tedy potřeba stanovit, ke kterým informacím má která strana přístup a jaké akce může provést.[6] Zároveň by mělo být lehce zjistitelné každé narušení pravidel protokolu. K naplnění těchto cílů se využívají prostředky, o nichž se zmiňuje v předchozím textu (šifrovací algoritmy, hashovací funkce…)

1.7.1 Challenge - Response

Je typickým zástupcem skupiny protokolů umožňujících výměnu informací mezi dvěma stranami, kdy jedna strana zasílá výzvy (challenge) k odpovědi na otázku a očekává od druhé strany ve stanoveném časovém limitu správnou reakci (response).[7] Znázorněno obrázkem 6. Probíhá-li vše jak má dojde k povolení přístupu. Nejjednodušší formou challenge-response je případ, kdy uživatel v rámci autentizace zadá své uživatelské jméno, přičemž počítač ho následně vyzve k zadání odpovídajícího hesla. Nevýhodou těchto metod je jejich provádění na větší vzdálenost než jen z klávesnice pod stůl (např. pomocí internetu), protože do hry nám zde může vstoupit třetí strana, která se bude snažit komunikaci zachytit. Z tohoto důvodu je nutné využívat metody chránící proti těmto útokům. Jmenovitě je to například používání jednorázových hesel, využívání solí (přidání pseudonáhodného čísla k heslu) nebo to, o čem se tu celou dobu hovořilo – užití šifry.



Obrázek 6 – Schéma Challenge-Response

2 QR kód

S rozvojem na poli informačních technologií se postupně začala vyskytovat potřeba komunikace mezi fyzickým světem (předměty) a jeho virtuálním protějškem (počítači). Nejčastěji šlo o nutnost uložit na předmět nějaká data (označení pro evidenci, …), která by se následně dala pokud možno co nejjednodušším způsobem „přečíst“ a zpracovat. To, jak byl tento problém vyřešen, vídáme možná nevědomky dennodenně na většině nakoupených produktů. Věřím, že sousloví „čárový kód“ vám dokonale objasní naši problematiku.

Popularita, kterou si čárové kódy v průběhu času získaly, vedla ke snaze rozšířit jejich využití. Během let se přidávaly šifrovatelné znaky a zvětšoval se objem dat přenositelných čárovým kódem. Z tohoto důvodu jednorozměrné kódy začaly být nedostačující, což vedlo k vytvoření „nové generace“ dvourozměrných čárových kódů, reprezentovaných především QR kódy.

2.1 Náležitosti

Samotný název QR kódů je velmi vypovídající, protože dešifrováním zkratky (QR – quick response) a jejím přeložením do jazyka českého, se dostaneme k označení „rychlá odpověď", z čehož nám hned vyplyne jedna z charakteristik těchto kódů. V praxi totiž stačí čtečkou namířit na obrázek QR kódu a data uložená na něm se zpracují téměř okamžitě a vyvolají požadovanou odezvu (zobrazí se obrázek/text, uloží se kontakt …).

Další vlastností patrnou na první pohled je zapisování kódů do čtverce, jenž je tvořen třemi stejně velkými, soustřednými čtyřúhelníky ve třech vrcholech, které určující pozici dat a tudíž směr čtení čtečkou, ve čtvrtém vrcholu značkou tvaru menšího čtyřúhelníku, a na spojnicích mezi těmito velkými čtyřúhelníky jsou úsečky tvořené střídavě bodem (černým čtverečkem) a mezerou (bílým čtverečkem).[8] Viz obrázek 7.

Obrázek 7 – Příklad QR kódu

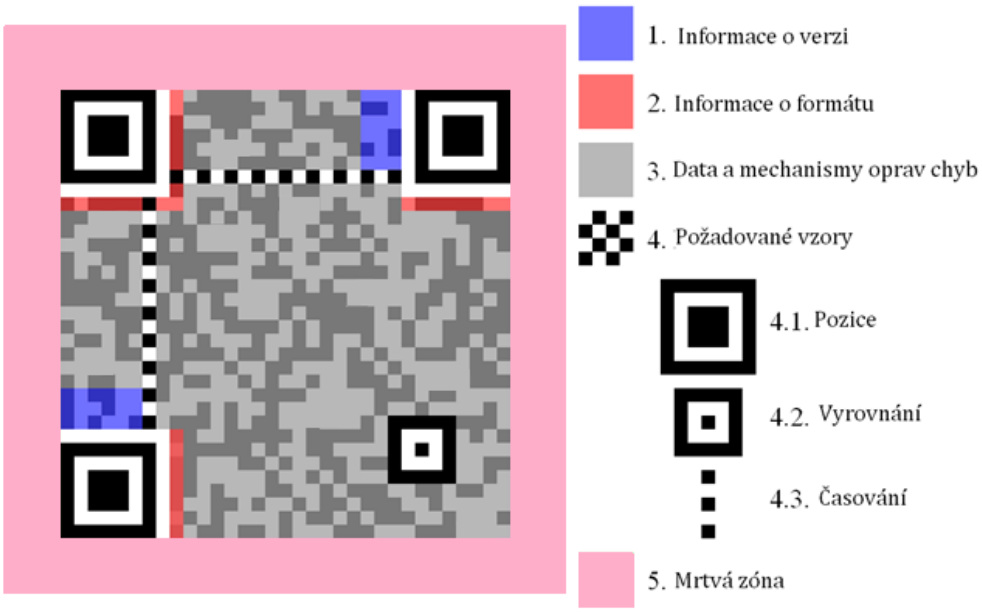
2.2 Parametry

Důležitým údajem pro QR kódy jsou bezesporu informace o verzi a „vzory" pro opravy chyb. Hovoříme-li o verzích, v podstatě nemáme na mysli nic jiné, než to jak „hustý" se kód jeví (viz obrázek 8). Verze jsou rozděleny od 1 do 40, přičemž nejmenší verze 1 má velikost 21x21 bodů a je schopná přenést nejméně dat, zatímco verze 40 má velikost 177x177 bodů a přenese dat mnohonásobně více. Pro představu je možné binárně uložit do kódu až 3000 bajtů. [8] Ani u oprav chyb nezabrousíme do ničeho obzvláště záludného. Určitě si dokážete představit, že celistvost kódu může být velmi snadno narušena nebo i jen při umísťování kódu není vždy k dispozici dokonale rovná plocha vhodná ke snímání. Aby se co nejvíce prodloužila životnost či umožnily další využití kódů, bylo nutné vyvinout systém, který by dokázal „nahradit" chybějící čtverečky. Z tohoto důvodu se do QR kódů, začaly přidávat „opravné mechanismy". Existuje jich více druhů, kdy ty nejnižší umožňují opravit 7% ztracených dat, u těch vyšších se objevují čísla kolem 30%.[9] Bohužel ale čím vyšší oprava chyb, tím méně dat pak lze ještě uložit do QR kódu.

Obrázek 8 – Srovnání verze 1 s verzí 10

2.3 Demonstrace

Všechny dříve zmíněné informace tedy shrneme a demonstrujeme na obrázku číslo 9. Modře máme odlišeny části QR kód informující čtečku, která verze je použita. Červeně jsou označeny čtverečky nesoucí informaci o formátu (jiný bude mít obrázek, jiný text). Růžovou barvou je znázorněna „mrtvá zóna“, tudíž ta část, v níž už se žádné data nevyskytují. Sytě černé a bílé jsou vybrané vzory, důležité pro čtečku, jejich popis je výše. A nakonec vše co zůstává šedým odstínem, je prostor pro uložení samotných dat.[10]



Obrázek 9 – Popis prvků QR kódu

2.4 Doslov

V QR kódech je určitě do budoucna uložen velký potenciál, který čeká na vhodnou příležitost naplno se rozvinout. Ovšem už teď stále více a více roste obliba těchto malých čtverečků, které se objevují na čím dál více místech a nabízejí mnoho informací.

3 Začínáme s Androidem

3.1 Slovo úvodem

Svět informačních technologií se v posledních letech rozvíjí až neuvěřitelným tempem. Dá se předpokládat, že tento trend bude i nadále pokračovat a řada z nás bude s tímto světem stále více propojena. Je tomu již pár let co se na trhu objevily smartphony (chytré telefony). Tato zařízení posunula naše možnosti neuvěřitelným způsobem vpřed. A využívá je stále více uživatelů, viz obrázek číslo 10. Cílem této části je seznámit Vás s právě těmito chytrými zařízeními a především s jejich softwarovým řešením, konkrétně s operačním systémem Android a umožnit Vám podílet se na vývoji aplikací pod OS Android.

3.2 Smartphone jako technologie, která dobývá svět

Smartphone neboli „chytrý telefon“ je nový typ mobilního telefonu, který používá mnoho nových pokročilých funkcí, které u běžných telefonů nenaleznete, a to hlavně z důvodu jejich ať už hardwarové nebo softwarové náročnosti. Na rozdíl od klasických mobilních telefonů je smartphone vybaven operačním systémem a chová se tedy jako malý počítač. Jeho velikost není větší než velikost klasického mobilního telefonu, a také proto je označován jako Personal Pocket Computer (PPC) neboli malý kapesní počítač.

Vůbec první smartphone by vyroben v roce 1992 společností IBM a nesl jméno Simson. Dalším mezníkem na poli chytrých telefonů byl smartphone od společnosti Nokia, a to model Nokia 9210, který byl prvním smartphonem, vyrobený s barevným displejem a byl to první smartphone s tzv. „otevřeným“ operačním systémem.[11] Od té doby šel vývoj značně kupředu až do podoby, jakou známe dnes.

V dnešní době již všechny smartphony vlastní operační systém. Díky operačnímu systému je možné do smartphonu nahrát různé aplikace a využívat tak potenciálu těchto telefonů nejen k obyčejné komunikaci. Smartphony nabízí také personalizaci a zároveň umožňují vytvořit vlastní spustitelnou aplikaci a tím podporují růst skupiny vývojářů mobilních aplikací.

Obrázek 10 - Zastoupení chytrých telefonů na trhu

3.3 Operační systémy pro mobilní zařízení

Trh dnes nabízí spoustu operačních systémů, a to jak známých a rozšířených tak i průkopníků v různých specializovaných oblastech. Níže jsou uvedeny nejznámější z nich.

Stručný přehled OS pro smartphony:

- Android – Platforma na linuxovém jádru určená jak pro mobilní zařízení, tak pro tablety. Tuto platformu vyvíjí společnost Google.
- iOS – Platforma od Applu, na které běží iPhone, iPad, iPod. Platforma je dostupná jak pro Tablet PC tak i mobilní a kapesní zařízení.
- Windows Phone 7 – Platforma do Microsoftu. Značně ztrácí na své popularitě a je stále méně distribuován.
- BADA – OS od společnosti Samsung, který je postaven stejně jako Android na linuxovém jádře, kombinuje dohromady grafické prostředí z Androidu a iOS. Na trhu je zatím jen pár telefonů s OS BADA.
- Symbian - Dříve velmi populární. Osazován hlavně do mobilních telefonů značky Nokia. S nástupem iOS či Androidu nastal velký úpadek této platformy a stále ubývá zařízení používajících Symbian.
- MAEMO (MeeGo**)** – Opět linuxové jádro (distribuce Debian). Vývoj: Nokia.
- Web OS – Dříve Palm OS. Podpora od společnosti HP. V budoucnu by měli Web OS obsahovat dokonce i tablety.
- RIM – Blackberry OS je populární hlavně v Americe. Slouží především manažerům nebo pro firemní účely.

3.4 Android - nástroj ke zkrocení smartphone

Android je jedním z výše uvedených operačních systémů, který si na trhu vybojoval v posledních letech významnou pozici. Jedná se o software s otevřeným zdrojovým kódem, (open source platforma) což umožňuje při dodržení daných podmínek systém zdarma užívat, přetvářet, popřípadě za splnění jistých podmínek distribuovat.

Podobně jako se samotným systémem, který je postaven na linuxovém jádře, je to i s danými aplikacemi, jejichž vývojem se tato práce bude zabývat (Android aplikace jsou programovány v jazyce Java, jehož knihovny jsou taktéž distribuovány zdarma). Aplikace, na kterých uživatel pracuje, nepřistupují k jádru přímo, ale komunikují pomocí Android API. Samotným jádrem se zpočátku není potřeba zabývat. Pozornost však bude věnována spíše systémovým vrstvám (OS Android je postaven na pěti vrstvách-Linux Kernel; Libraries; Android Runtime; Application Framework; Applications), které bude potřeba znát k vytvoření základních aplikací spustitelných na smartphonech.[12] Nejdůležitější vrstvou pro vývojáře je vrstva s názvem Application Framework. Ta umožňuje přistupovat jak k hardwarovému vybavení přístroje tak jako využít možnosti pracovat s obsahem jiných aplikací či ovládat celý životní cyklus aplikace. Důležité jsou i vrstvy Android Runtime a Applications. Android runtime obsahuje virtuální stroj DVM a základní knihovny Apache pro práci ze sítí. Poslední, nejvyšší vrstva zahrnuje již samotné aplikace, a to ať již předinstalované, stažené z Android Market nebo právě vývojáři vytvořené aplikace.

3.5 Stručná historie OS Android

Android je operační systém vytvořený, každému jistě známou, firmou Google. Jak bylo řečeno, je založen na open source platformě a již to naznačuje, že byl vybudován zcela od základů. V roce 2003 byla v Kalifornii založena společnost Android několika soukromníky, kteří začali vyvíjet mobilní aplikace. Rok 2005 byl pro firmu Android velkým mezníkem, jelikož se o něj začala zajímat společnost Google a nakonec došlo i k odkupu.[13] Firma Android se tak stala dceřinou společností Googlu a došlo k převzetí veškerých závazků i zaměstnanců.

K samotnému představení operačního systému Android došlo až v roce 2007, a to v den vzniku Open Handset Aliance, která sdružuje desítky firem pohybujících se na mobilním trhu, a to jak samotné operátory, tak i výrobce jednotlivých součástek pro smartphony. Cílem této aliance bylo vytvořit a udržet otevřené standardy pro chytrá zařízení.

V průběhu let si Android dobýval své místo na trhu a bojoval s konkurencí. První oficiální verze OS Android šla na trh 3. září 2008 a to v mobilním zařízení nesoucí název HTC Dream. V průběhu let pak vycházely další verze OS nesoucí s sebou vylepšení, které uživatelé jistě ocení, přidání nových funkcí a vyladění chyb. Nejnovější verze androidu je Android 4.2.

3.6 Seznámení s potřebným softwarem

Možná nikoho nepřekvapí, že vývoj aplikací běžících pod OS Android probíhá na PC nikoliv přímo ve smartphonu samotném. To ovšem s sebou nese jisté komplikace. OS počítače je jiné než mobilního přístroje. Rozlišení monitoru a ovládací prvky včetně periferií jsou také zcela odlišné. Na několika následujících stranách je popsán postup krok po kroku od samotné instalace potřebného software až po nahrání funkční aplikace do reálného zařízení.

Zpočátku je potřeba nahrát do PC potřebný software, který zajistí vývoj, kompilování, testování, ladění a spuštění vytvořených aplikací dovolí. Na internetu je k dohledání nespočet free programů které vývoj aplikací pod OS Android umožňují či usnadňují

3.6.1 Seznam potřebného software

- Java Development Kit (JDK)
- (Android) Software Development Kit (SDK)
- Java Virtual Machine (JVM)
- Program Eclipse + vybrané pluginy
- Android Development Tool (ADT)

3.6.1.1 Java Development Kit (JDK)

Java Development Kit je balíček základních nástrojů a knihoven (v jazyce Java), které budeme využívat pro vývoj našich aplikací. Tento nástroj je potřeba pro to, že Android aplikace se programují v jazyce Java.

3.6.1.2 Software Development Kit (SDK)

SDK je další nástroj, který zjednodušuje vývoj mobilních aplikací. Je to balíček, který umožňuje vytvářet aplikace pro určité operační systémy. Tento balíček obsahuje knihovnu API a další nástroje, které umožňují tvorbu ale také spuštění Android aplikací na PC (ke spuštění slouží emulátor).

3.6.1.3 Java Virtual Machine (JVM)

Pokud na PC není ještě nainstalovaný nástroj JVM, je nutná jeho dodatečná instalace. Tento nástroj je potřeba k překladu jazyka Java do strojového kódu.

Stažení se automaticky nabídne po instalaci vývojového prostředí Eclipse nebo si jej můžete stáhnout z některého internetového úložiště.

3.6.1.4 Vývojové prostředí Eclipse

Na internetu se nachází mnoho nástrojů, jež uživateli usnadní samotný zápis syntaxe (zdrojového kódu daného programu). Pro účely zapsání stačí obyčejný textový editor, jako je například notepad. Zkušený uživatel ale bude využívat výhod, které nabízí program Eclipse. Jedná se o volně stažitelný software a jeho obrovskou předností je možnost rozšíření o spoustu doplňků. Jedná se o program spolupracující přímo se společností vyvíjející OS Android.

3.6.1.5 Android Development Tool (ADT)

Nástroj, který Eclipse naučí vytvářet Android projekty. Tento plugin je třeba doinstalovat pouze v případě, že již není obsažen v instalované verzi Eclipse.

3.7 Stažení a instalace potřebného software

Pokud je většina nástrojů již v PC a chybí jen určitý program, v příloze jsou uvedeny linky, na kterých je možné tyto programy stáhnout nebo lze najít a stáhnout potřebný software na jednom z internetových úložišť. Pokud většina potřebného software chybí, nebo je zastaralý, postup bude následující.

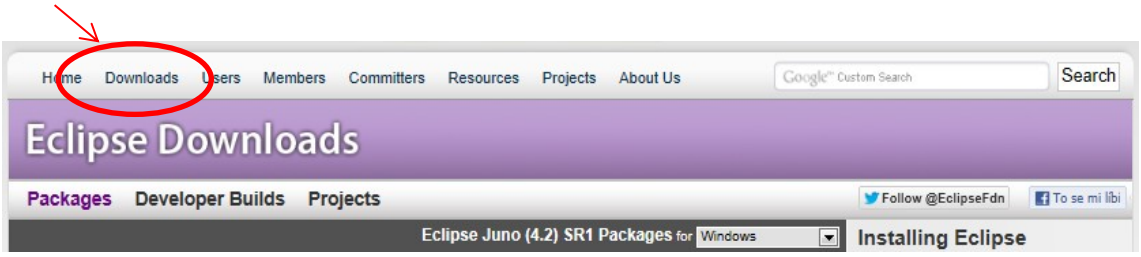
Postup platí pro OS Windows 7 a aktuální verze programů k datu 5.9 2012. V jiných verzích OS Windows či novými aktualizacemi programů by měl být obdobný. Na jiných OS se budou postupy poněkud odlišovat.

3.7.1 Stažení vývojového prostředí Eclipse

Instalační balíček tohoto nástroje se nachází na domovské stránce Eclipse v sekci Downloads (obr 11). Tedy na adrese:

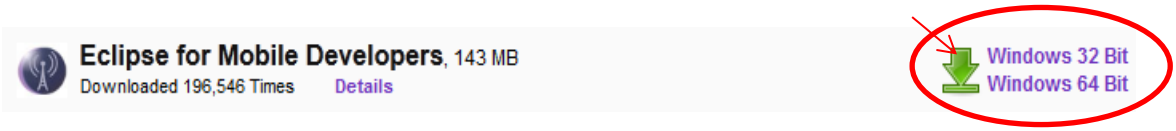
http://www.eclipse.org/downloads/

Výběr je z několika verzí daného programu. Je třeba zvolit tu nejvhodnější a následně ji stáhnout. Eclipse je také nabízeno ve verzi 32bit či 64bit (je třeba vybrat stejnou verzi jako je OS Windows nainstalovaný na daném PC). Je vhodné stáhnout Eclipse pro mobilní vývojáře, tedy Eclipse for Mobile Developers (obrázek 12).



Obrázek 11 - Hlavička Downloads

Po kliknutí na příslušnou verzi programu by se měla načíst stránka, kde lze program stáhnout. Pokud se tak nestane, lze Eclipse získat pomocí následujícího odkazu:



Obrázek 12 - Výběr verze programu

http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/juno/R/eclipse-mobile-juno-win32-x86_64.zip

Pro stažení souboru se zahájí kliknutím na zelenou šipku a zvolením možnosti Uložit. (viz. Obr. 13). Dále je třeba nastavit cestu, kam se má soubor uložit.

Obrázek 13 - Stažení Eclipse

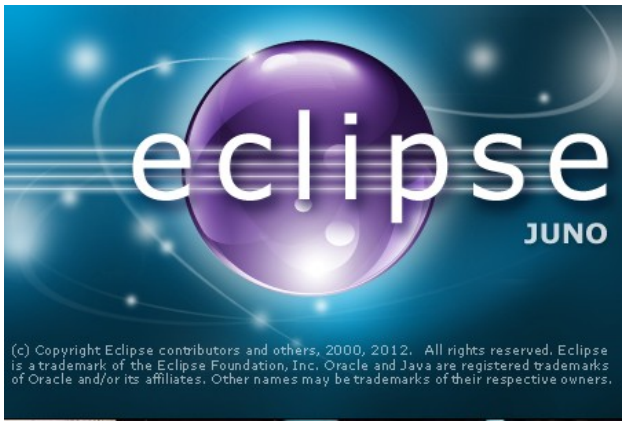
V adresáři, který byl při stahování zadán, bude zobrazen po dokončení stahování soubor s příponou zip. Pokud není v PC potřebný program ke spuštění tohoto archivu, lze jej stáhnout zde:

http://www.stahuj.centrum.cz/utility_a_ostatni/komprese/winrar/?g[hledano]=winrar&g[oz]=4.20

Následně stažený soubor otevřete a archiv extrahujte do vámi zvoleného adresáře.

3.7.2 Spuštění vývojového prostředí

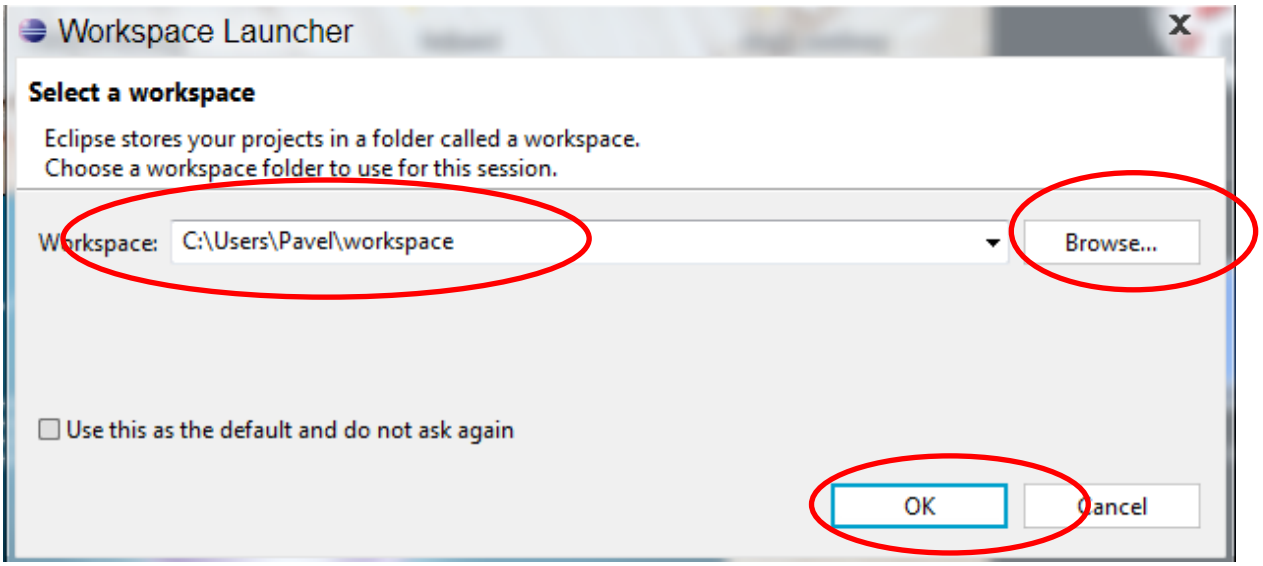
Program Eclipse se spouští jednoduše dvojklikem na ikonku Eclipse.exe Spuštění Eclipse (obr. 14). Při prvním spuštění Eclipse mohou nastat dva stavy. Buď je již na PC nainstalována Java (Java Virtual Machine popř. další komponenty a balíčky) a zobrazí se okno s definováním pracovního místa (obr. 15.) anglicky workspace a nebo Java Virtual Machine potřebná k překladu aplikace do strojového kódu naistalována není a dialogové okno na tuto skutečnost upozorní a nabídne stažení potřebného software. Stažení a instalace souboru probíhá klasickým způsobem. Je potřeba akceptovat licenční podmínky a pak klikat na tlačítka Next a Finish.



Obrázek 14 - Spuštění Eclipse Juno

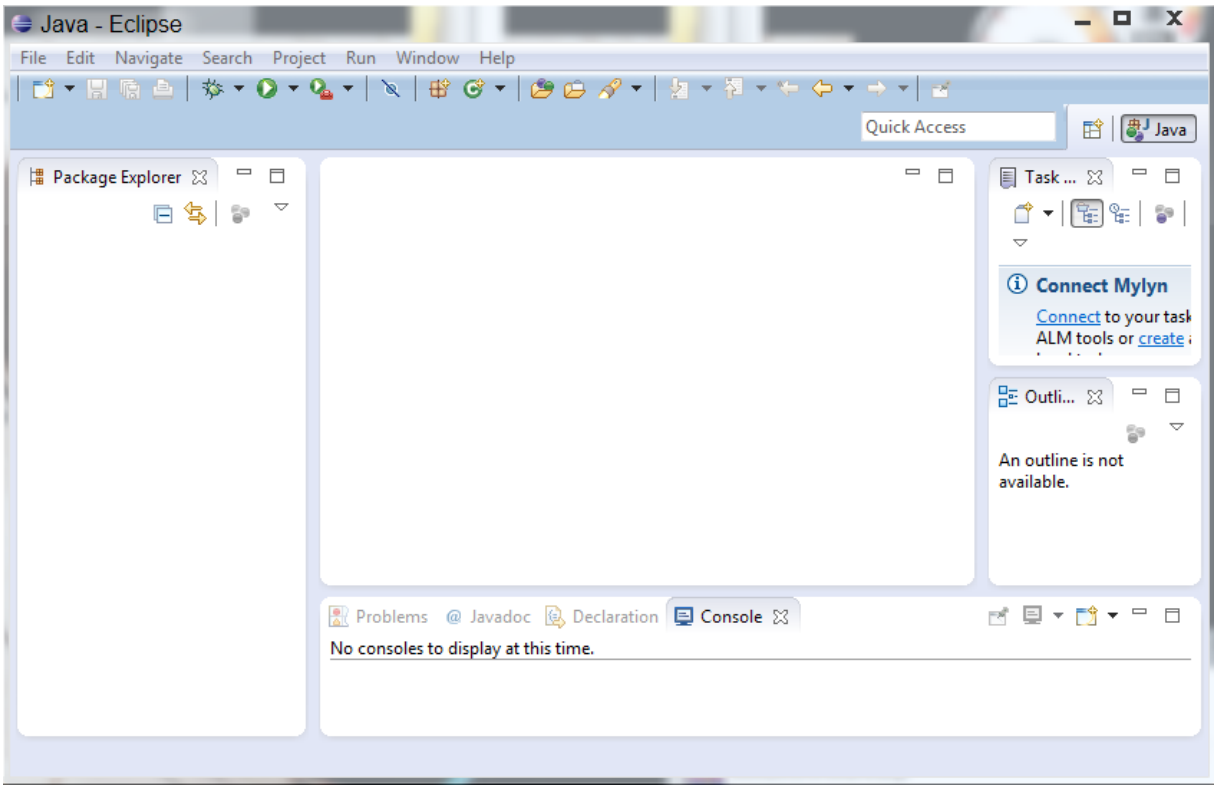
3.7.3 Definování Workspace

V naprosté většině případů však potřebný software již v PC instalován je, a výběrem adresáře, je třeba vytvořit vlastní pracovní místo (Workspace) jež bude využíváno projekty k ukládání dat. Lze nechat buď defaultně nastavenou cestu, anebo kliknutím na Browse zadat cestu vlastní. Dále doporučuji zatrhnout možnost: Use this as the default and do not ask again a tím tak vytvořit stálé pracovní místo (v případě nezaškrtnutí této možnosti se okno zobrazí při každém spuštění). Následně je třeba vše potvrdit kliknutím na tlačítko Ok.



Obrázek 15 - Definování Workspace

Po chvíli se zobrazí uživatelské rozhraní programu Eclipse které vypadá následovně (obr. 16).



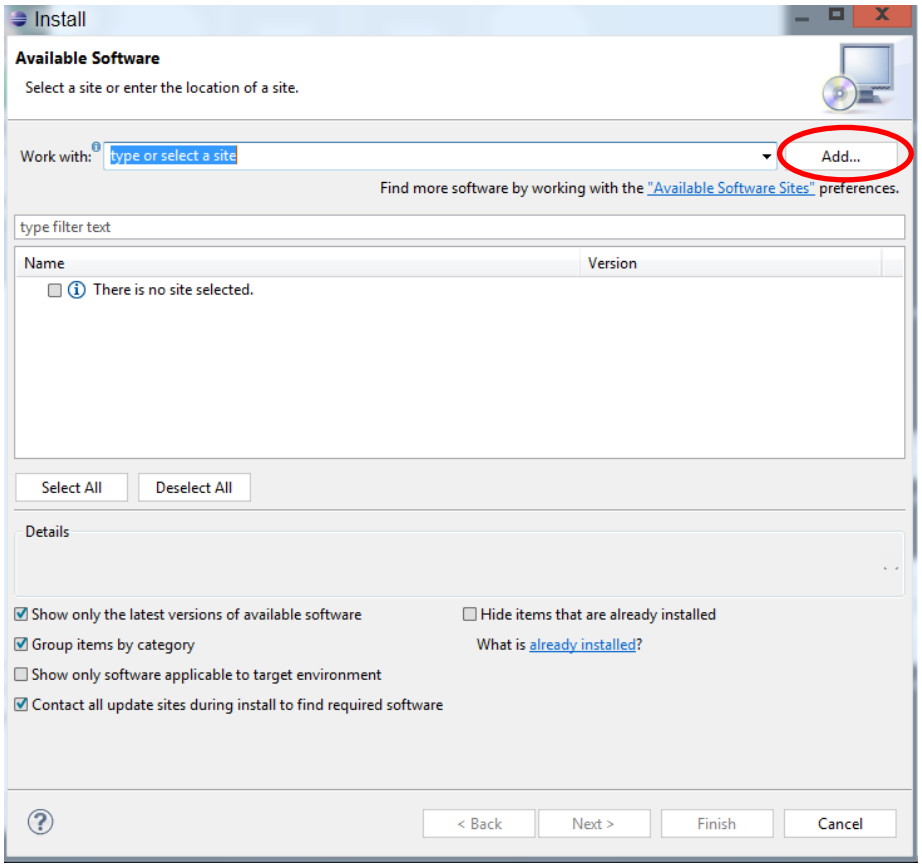
Obrázek 16 - Uživatelské rozhraní Eclipse

3.7.4 Android Development tool (ADT)

Eclipse je již nainstalováno, dalším krokem bude propojení Eclipse se SDK a ADT. Tento krok umožní ve vývojovém prostředí Eclipse vytvářet, kompilovat a spouštět Android aplikace (ke spouštění aplikací dochází v Emulátoru který je součástí SDK).

3.7.4.1 Stažení a instalace ADT

Samotná instalace pak probíhá následovně. Je třeba otevřít Eclipse, v hlavním menu zvolit záložku Help, dále vybrat možnost Install New Software. Následně se otevře dialogové okno, kliknutím na Add se zobrazí okno Add Repository.

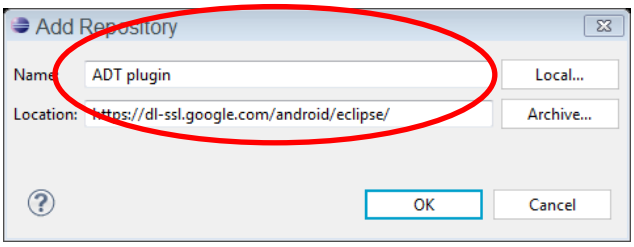


Obrázek 17 - Instalace ADT

Do kolonky s názvem Name je třeba zadat jméno pluginu, tedy ADT plugin a do kolonky Location zadat adresu, odkud má být doplněk stažen. V tomto případě je možné použít následující adresu:

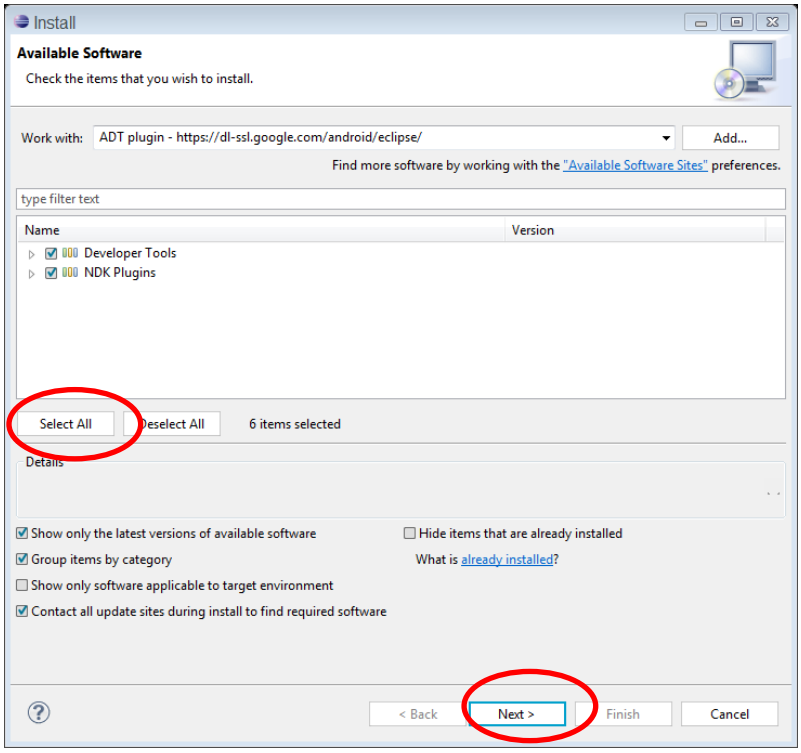
https://dl-ssl.google.com/android/eclipse/

Nastavení se uloží kliknutím na OK.



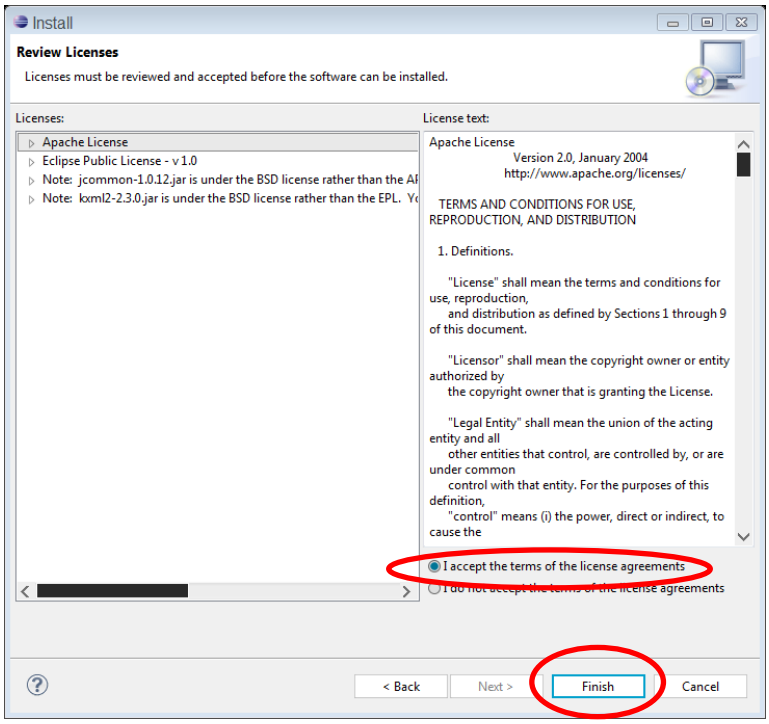
Obrázek 18 - Zadání jména a adresy

Potom se objeví další dialogové okno. Kliknutím na položku Select All dojde k vybrání všech položek a následným klinutím na Next, dojde k přesunu na odsouhlasení licenčních podmínek.



Obrázek 19 - Volba pluginu

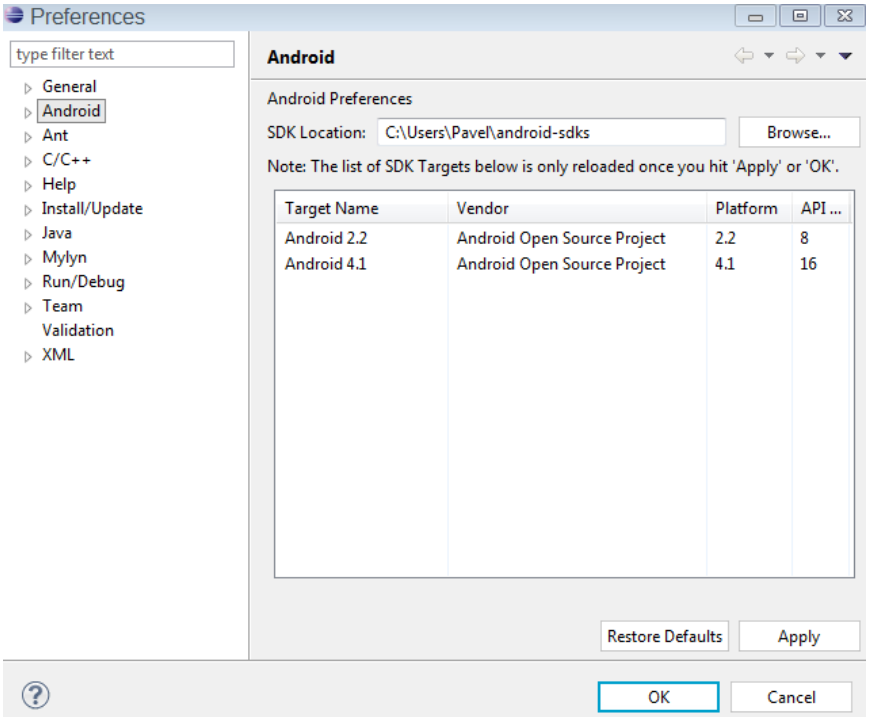
Dále je třeba ještě odsouhlasit licenční podmínky. Učiní se tak zatržením položky I accept the terms of the license agreements. Akce se dokončí zvolením tlačítka Finish.



Obrázek 20 - Odsouhlasení licenčních podmínek

3.7.4.2 Konfigurace ADT

Po stažení ADT je potřeba program Eclipse restartovat a ADT plugin správně nakonfigurovat. Po znovu-spuštění Eclipse dojde k zobrazení dialogového okna s konfigurací SDK, jelikož je potřebné pro správný běh ADT. Následně se nabízí dvě možnosti. Buď je možné použít existující SDK, anebo nechat Eclipse vyhledat a stáhnou SDK dle jeho volby. Pokud již SDK na PC instalováno je, zvolíme použít existující SDK, tedy Use existing SDKs, v ostatní případech je nutné jej doinstalovat a zvolit druhou možnost která nabídne stažení všeho potřebného a kliknutím na Finish operaci dokončit.



Obrázek 21 - Konfigurace ADT

3.7.5 Software Development Kit (SDK)

Balíček Software Development Kit (SDK) slouží k vytváření aplikací pro různé operační systémy. Instalací tohoto balíčku získá Eclipse přístup ke knihovnám API a Java a taktéž získá výkonný emulátor, na kterém lze testovat vytvořené aplikace.

3.7.5.1 Stažení a Instalace SDK

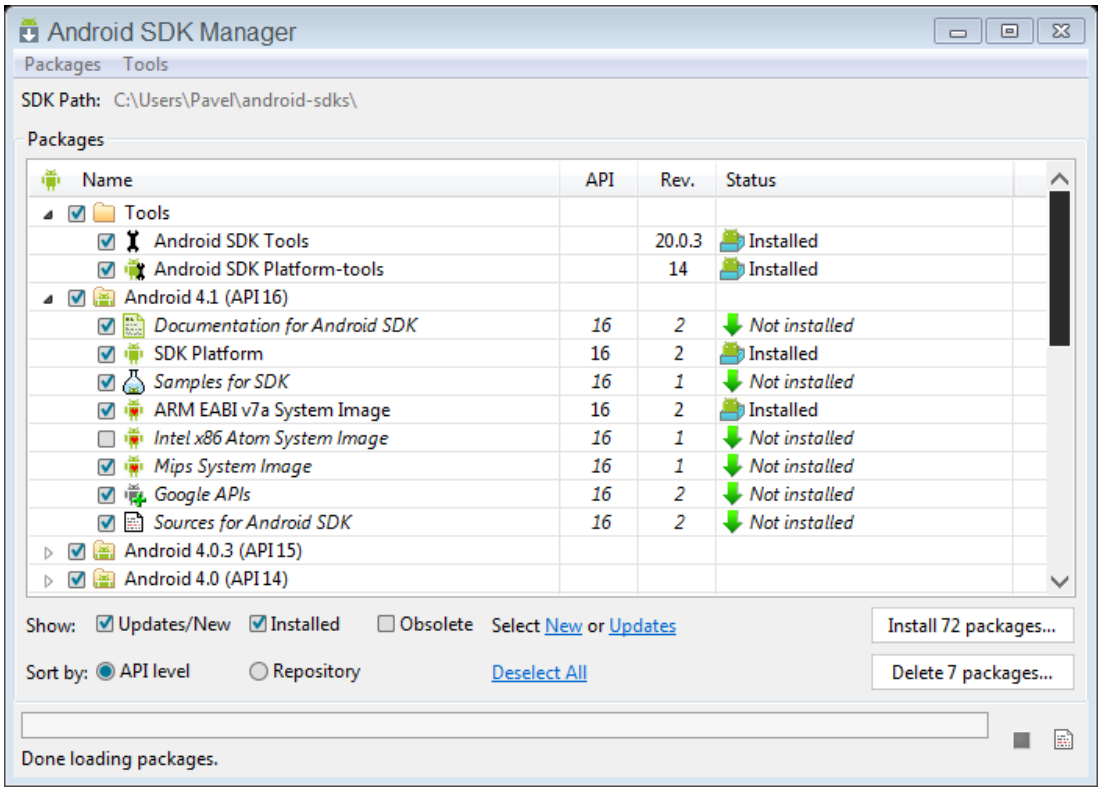
SDK je možné stáhnout na uvedené adrese.

http://developer.android.com/sdk/index.html

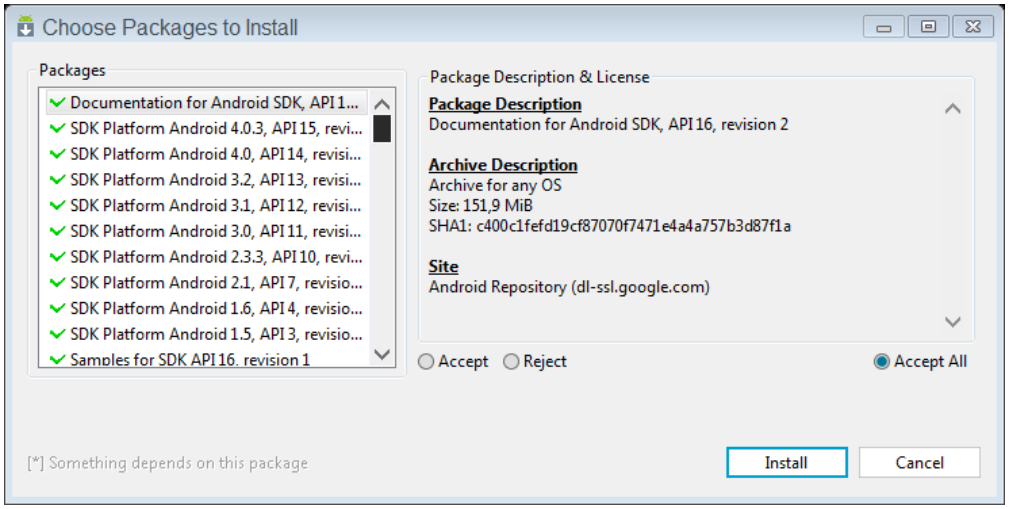
Nebo lze stáhnout a instalovat SDK přímo ve vývojovém prostředí Eclipse.

Po zobrazení dialogového okna Android SDK Tools Setup je třeba vybrat možnost Browse a zadat cestu k adresáři kam bude SDK instalováno. Následným kliknutím na Next, Accept, All, Finish dokončíte operaci.

Následně by se měla spustit úvodní obrazovka SDK Manager, pokud se tak nestane, Je možné ji vyvolat dvojklikem na SDK Manager.exe v adresáři který byl v předchozím kroku zvolen pro instalaci SDK.



Obrázek 22 - Stažení příslušných platforem



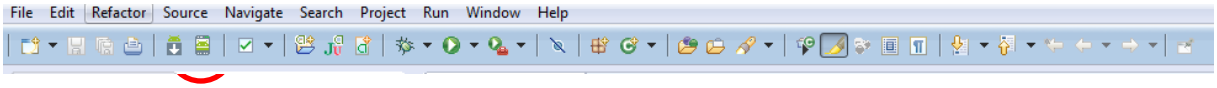
Obrázek 23 - Přijetí Licence

Na úvodní obrazovce bude zobrazen výběr platforem, které je možné použít. Výběrem všech se nedá nic pokazit, ovšem instalace bude zdlouhavá a přebytek doplňků vývojové prostředí dost zpomaluje. Pro účely této práce stačí defaultně zaškrtnuté možnosti. Následuje opětovné odsouhlasení licenčních podmítek volbou Accept All a stiskem Install.

Po úspěšné instalaci restartujte prostředí Eclipse a vytvořte Nové virtuální zařízení (AVD)

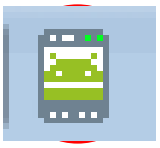
3.7.5.2 Přidání nového virtuálního zřízení

Nové virtuální zařízení je možné přidat přes vývojové prostředí Eclipse a to kliknutím na



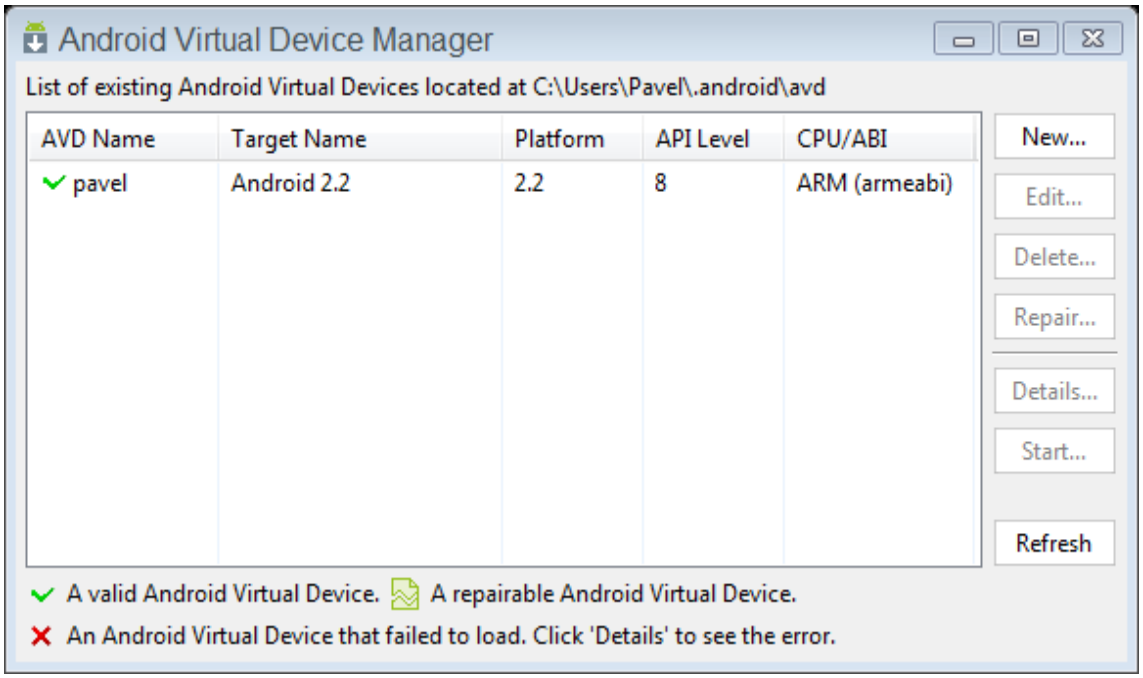
Obrázek 24 - Otevření emulátoru

tlačítko vyobrazené mobilním zařízením.



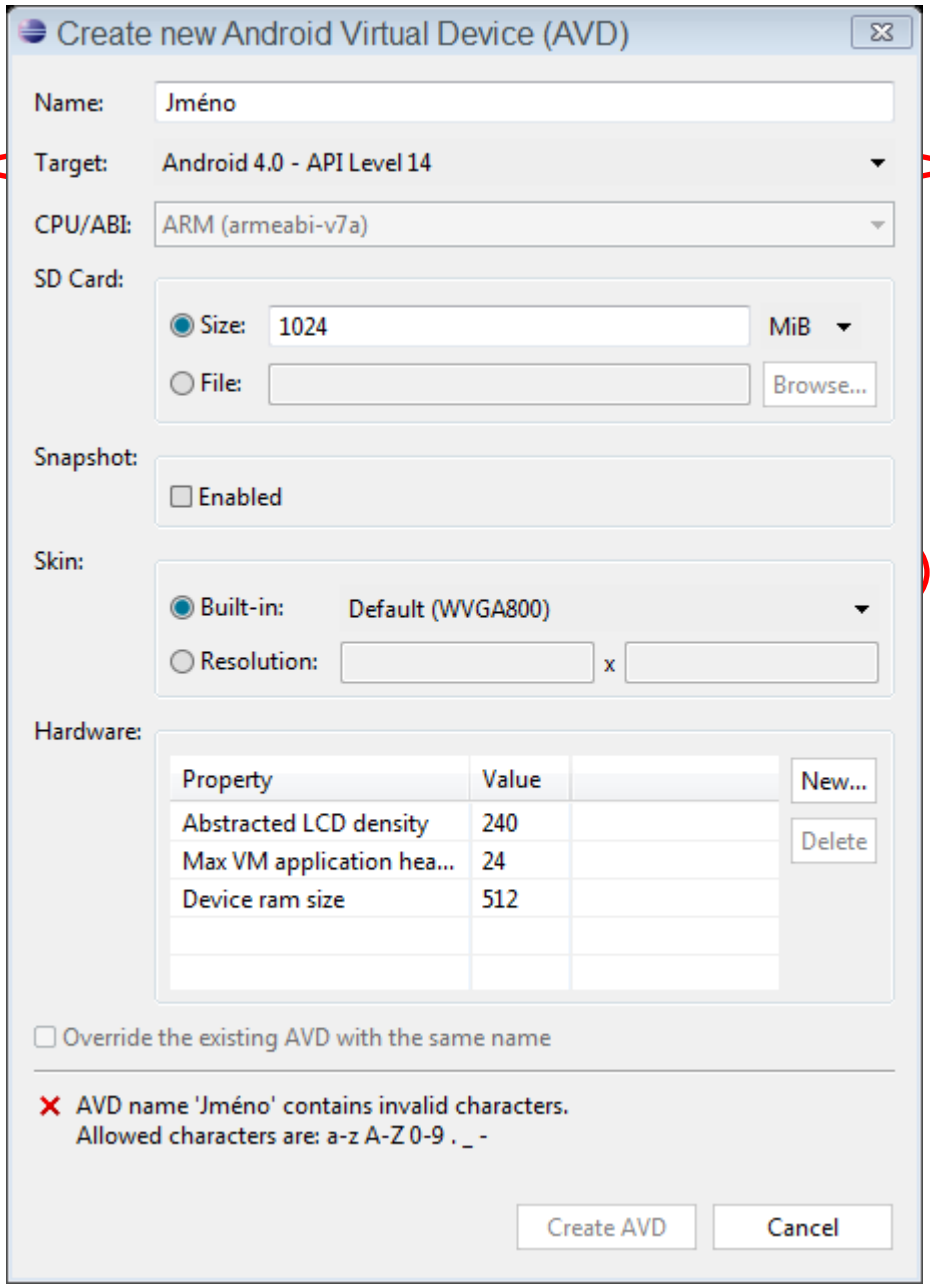
Obrázek 25 - Otevření emulátoru 2

Po zobrazení dialogového okna Android Virtual Device Manager se vytvoří nové virtuální zařízení zvolení možnosti New



Obrázek 26 - Tvorba virtuálního zařízení

Tím se otevře okno s nastavením jednotlivých parametrů nového virtuálního zařízení. Do pole Name se zadává název nového zařízení (například Nexus). V poli Target je potřeba zvolit příslušnou verzi OS Android, kterou bude zařízení využívat, a taktéž potřebnou knihovnu API. Do pole SD Card se zadává velikost simulované paměťové karty. Skin doporučuji nechat implicitně nastaven. V položce Hardware se přidává potřebný hardware. Pro jednoduché aplikace stačí jen malé množství přednastaveného hardware. Kliknutím na Create AVD dojde k vytvoření nového virtuálního zařízení.



Obrázek 27 - Nastavení parametrů

3.7.6 Další nutný software a aktualizace

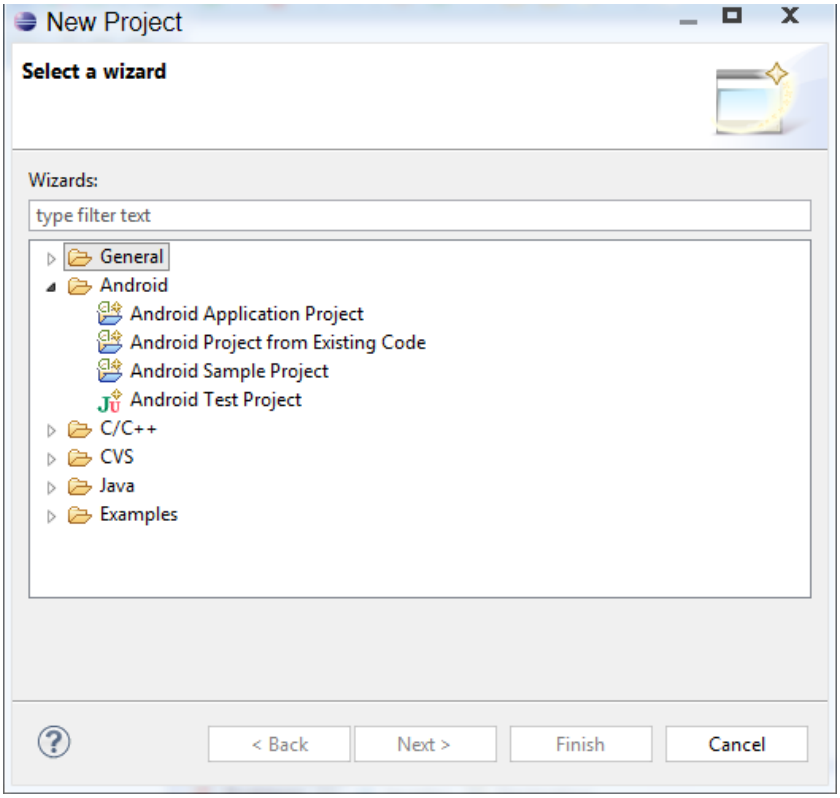
Vše potřebné by v tuto chvíli mělo být nainstalováno. Pokud by snad jenom prostředí Eclipse nabízelo při svém spuštění nebo pří práci v něm Stažení jistého software či aktualizaci stávajícího software, potvrďte tuto možnost a instalační balíček stáhněte tak jako v předchozích případech a možnostmi next se proklikejte k odsouhlasení licenčních podmínek k instalaci samotného software a následnému ukončení instalačního balíčkou kliknutím na finish.

3.8 První aplikace pro Os Android

První aplikaci pro Os Android vytvoříte velice snadno pomocí jednoduchých kroků popsaných níže.

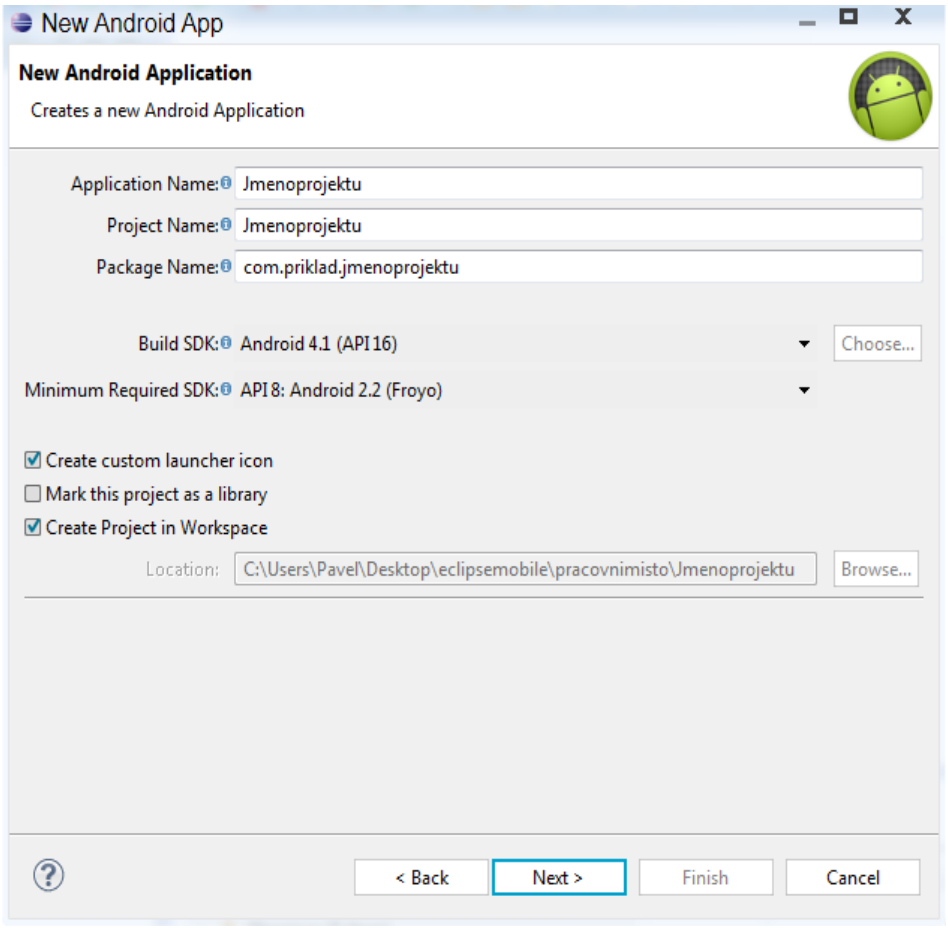
3.8.1 Vytvoření Android Projektu

V hlavním menu vývojového prostředí je třeba vytvořit nový projekt. To učiníme výběrem možnosti File a kliknutím na New Project. Tím se otevře dialogové okno a v něm zvolením složky Android výběrem Android Project a následným kliknutím na tlačítko Next dojde k otevření potřebného dialogového okna.



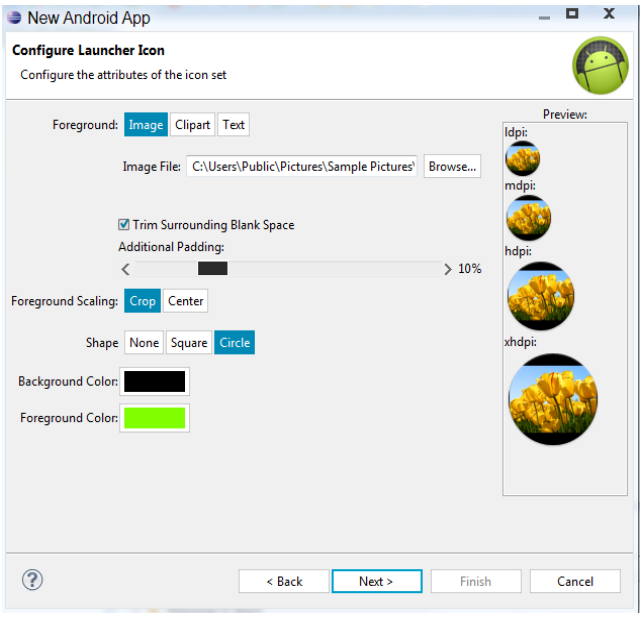
Obrázek 28 - Výběr typu projektu

V dalším dialogovém okně je potřeba zadat několik různých informací. Do pole Application Name zadat jméno projektu. Pole Project Name se vyplní automaticky tak jako Package name. U pole Packet Name vyskočí chyba (toto pole umožňuje identifikaci projektu na Android Market, a proto je vhodné přepsat jeho část. Nahradit slovo example třeba slovem příklad či jiným slovem a problém bude vyřešen. Dále je třeba nastavit, pro jaké verze OS Android bude Aplikace určena. Build SDK slouží pro zadání nejvyšší přípustné verze, Minimum Requier SDK slouží k nastavení naopak nejnižší verze OS android a jeho knihoven.[14] Defaultně je zaškrtnuta položka Create Project in Workspace. Pokud zůstane políčko zaškrtnuté, projekt se automaticky uloží na místo, které bylo nastaveno jako Workspace. V případě, že je třeba cestu k projektu změnit. Zrušením zatržení této položky a pomocí tlačítka Browse lze nastavit cestu k jinému adresáři. K dalšímu kroku se přejde kliknutím na Next



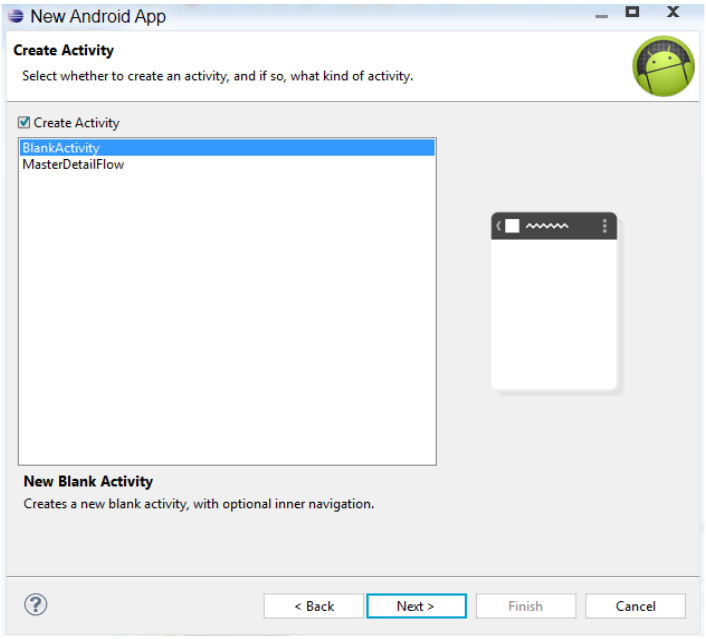
Obrázek 29 - Vytvoření nového Android projektu

V další obrazovce lze nastavit vzhled ikony, která se zobrazí v hlavním menu aplikací. V pravé části tohoto okna se nachází náhled. Uživatel si může vyzkoušet, co jednotlivé možnost s ikonou dělají. Rozhodně nic nemůže pokazit.



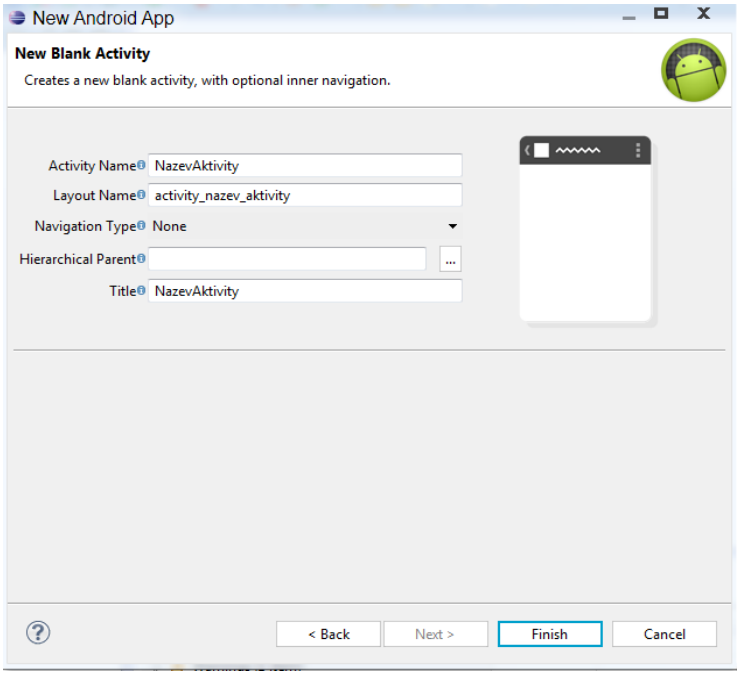
Obrázek 30 - Personalizace ikony

Další obrazovkou se není třeba ze začátku vůbec zabývat, jen je nutné zkontrolovat, jestli je vybrána možnost BlanckActivity a kliknutím na tlačítko Next se následně posunout dál.



Obrázek 31 -Vytvoření aktivity

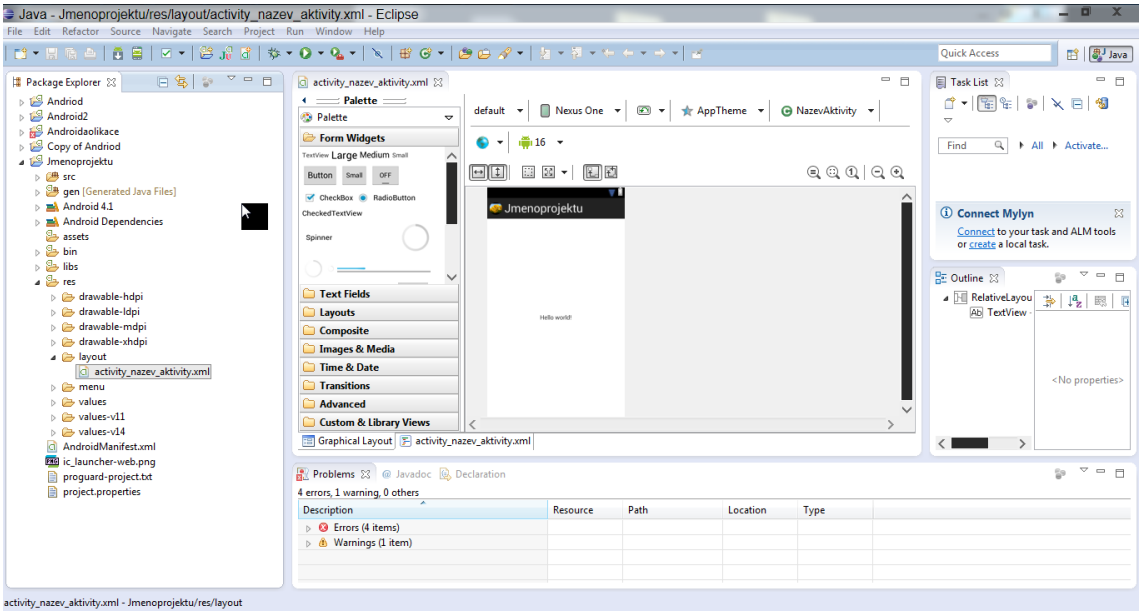
V dalším okně je nutné nastavit jméno první Aktivity. Do pole Activity Name. Pole s názvem Hierarchial Parent v tomto konkrétním případě zůstane nevyplněné (slouží pro nastavení dědičnost, která se bude využívat až při vytváření vícero aktivit). Nastavení bude dokončeno po kliknutí na tlačítko Finish. Zpracování chvíli trvá, tudíž je potřeba počkat na úplné dokončení tvorby našeho prvního Android projektu.



Obrázek 32 - Nastavení aktivity

3.8.2 Správa a spuštění projektu

Po dokončení předchozí kroku se zobrazí následující obrazovka.



Obrázek 33 - Ukázka vývojového prostředí

Dřív, než se pustíme do vývoje složitějších programů, vyzkoušíme nejprve funkčnost všech komponentů kliknutím na zelenou šipečku a volbou možnosti Start as Android Application, spustí se tak přednastavený projekt Hello World, který vývojové prostředí Eclipse automaticky vygenerovalo. Po chvíli by se mělo zobrazit okno, z něhož je potřeba vybrat vytvořený Emulátor a kliknout na start. Samotné spuštění emulátoru trvá strašně dlouho (obzvlášť při použití staršího PC nebo jeden z méně výkonnějších notebooků. Po několika minutách se spustí operační systém android a na úvodní obrazovce je potřeba odemknout virtuální mobilní přístroj. Následně se spustí, aplikace Hello World obrazovka nejdříve zbělá a následně vypíše textové pole se slovy Hello World.

3.9 Aplikace v reálném zařízení

Když vytvoříme program a chceme jej otestovat, doladit atd. na reálném zařízení, první, co je potřeba, nějaké takové vlastnit. Android je sice ve vývoji spolehlivější než iOS, ale i tak se může stát, že aplikace běžící na emulátoru v reálu fungovat nebude nebo naopak. Fyzické zařízení tedy již vlastníme, následný postup je zcela triviální

3.9.1 Nahrání aplikace do reálného zařízení

Začneme tedy se samotným nastavením mobilu. Pokud aplikaci nenahrajeme přímo na Android market (dnes Google Play), budeme muset v nastavení telefonu někde najít položku Aplikace a v ní zaškrtnout políčko Neznámé zdroje (povoluje instalaci aplikací, které nepocházejí ze služby Market)

Dále je potřeba do smartphonu aplikaci nahrát. Na výběr je několik možností.

- Pomocí Google Play
- Nahráním na určitou webovou, či wapovou stránku a následovným stažením
- Posláním aplikace přes e-mailovou schránku jako přílohu a následovným otevřením v mobilu
- Pomocí bezdrátového připojení k PC (Bluetooth Wi-Fi)
- Pomocí USB kabelu

Osobně doporučuji posílat aplikaci do mobilu přes Bluetooth

Tak tedy zapneme Bluetooth jak na PC, tak i na Mobilu (nezapomeňte zapnout viditelnost zařízení), obě zařízení spárujeme a pošleme soubor, který najdeme ve zvoleném (či defaultně) nastaveném adresáři Workspace, ve složce bin. V této složce je soubor nesoucí název projektu s příponou ".apk" například ahojsvete.apk. Pokud soubor chybí, s největší pravděpodobností není zkompilován projekt, což by se ovšem nemělo stát, jelikož Eclipse automaticky kompiluje celou aplikaci při každém pokusu o její spuštění v emulátoru.

3.9.2 Spuštění aplikace v reálném zařízení

Po přijetí souboru v mobilu jej stačí následně jen otevřít a zvolit možnost instalovat aplikaci. Počkat chvíli, než se dokončí instalace. Následně otevřít hlavní menu (v mobilním zařízení nikoliv v PC) a najít ikonu aplikace, kterou jsme vytvořili, otevřete ji a čekejte, zda vytvořená aplikace udělá totéž, co probíhalo na emulátoru. Pokud ano, blahopřeji, máte za sebou první krok do světa vývojáře mobilních aplikací.

3.10 Doslov

I když aplikace běžící pod OS Android jsou vyvíjeny na jiném systému, než jsou následně spouštěny, neměl by mít s programováním základních aplikací problém ani úplný laik. Instalace a konfigurace všech balíčků a aplikací je intuitivní a na internetu je spousta návodů a rad, které poslouží nezkušeným uživatelům jako vodítko, kterého se lze držet. Samotné prostředí Eclipse je velmi přehledně zpracované a má relativně rychlou odezvu. Vývojář může využít širokou škálu možností, jež tento program nabízí, jako je například korekce a upozorňování na chyby v zápisu i logické chyby. Tak i možnost spustit aplikaci na vlastnoručně nakonfigurovaném emulátor či více emulovaných zařízení najednou. Co se týče distribuce vytvořené aplikace, dá se říct, že je na špičkové úrovni tak jako podpora pro developery, která je sice v angličtině, ale je doplněna spoustou obrázků a screenshotů zpracovaných velmi dopodrobna. Mobilní aplikace mají jistě před sebou velkou budoucnost a společnost Google výborně nastartovala projekt, který téměř každému umožňuje se zapojit a být součástí dění světa jménem Android.

4 Tvorba výsledných programů

4.1 Úvod do problematiky

Díky nově nabytým zkušenostem při tvorbě programu „Hello World“ jsme se mohli pustit do vytváření výstupu naší práce. Abychom vyhověli požadavkům, bylo nutné vytvořit dva na sobě nezávislé programy (server a klient), které byly ještě následně rozděleny na dva módy (části). První podmínka byla nutná z bezpečnostních důvodů, zatímco ta druhá sloužila jinému účelu – demonstrování rozdílů mezi dřívějšími a našim systémem.

Pojmenováním server máme označenu aplikaci běžící na operačním sytému Linux nebo Mac OS X. Tato aplikace obsahuje data o každém uživateli, umožňuje vytváření nových uživatelů a jejich správu. Jejím hlavním cílem je starat se o autentizaci uživatele a následnou autorizaci. Může tak sloužit jako například elektronický vrátný nebo přístupový bod k uloženým, chráněným datům.

Klientem se pak rozumí aplikace běžící na smarthphonech, které jsme si pro jejich skvělé vlastnosti (snadná přenositelnost, předmět každodenní potřeby, výpočetně silné zařízení, možnost dohledání v případě odcizení) zvolili. Aplikace je pak vyvíjena na platformě Android. Jejím cílem je zprostředkovat každému jednotlivému uživateli přihlášení do systému a umožnit tak autentizaci daného uživatele.

4.2 Základní struktura systému

Vše se zakládá na naprosto jednoduché myšlence. Uživatel se autentizuje pomocí klientské aplikace. Tedy potvrdí, že je skutečně ten, za koho se vydává. Serverová aplikace následně vyhodnocuje, zda má daný uživatel příslušná práva (vstup do budovy, přístup ke střeženým datům). Pokud serverová aplikace vyhodnotí, že uživatel se prokázal správným autentizačním klíčem a současně má práva přístupu tam, kam žádá, bude následně autorizován, čímž dojde například k otevření dveří, či zpřístupnění databáze údajů.

4.2.1 Autentizační klíč

Jedná se o určitý předmět, heslo, cokoliv, co prokazuje uživatelovu totožnost. Právě použití dostatečně silného, unikátního, ale taky eticky přijatelného autentzikačního klíče je jedna z největších slabin většiny systémů.

Většina systémů používá klíč velmi slabý. Vůbec nejčastěji k jeho prolomení (získání třetí stranou) dochází u systémů zajišťující přístup k digitálním datům. Hesla uživatelů se sice zpracovávají ve formě hashů, dle potřeby se solí a provádí se s nimi operace, které mají prolomení hesla ztížit, ovšem ani tato ochrana není dostatečná a to především pro uživatele, kteří používají velmi slabá hesla, jako jsou 12345, qwertz, pasword a podobně. Útočníkovi pak stačí i hash hesla, který porovnává s takzvanými duhovými tabulkami, což jsou shromážděné otisky nejčastěji používaných hesel. V případě shody je heslo prozrazeno. Velká nevýhoda těchto systému taktéž je, že v reálném čase můžeme posílat tisíce požadavků na server a heslo uhádnou. Řešením může být, a často bývá, solení hashů a odmítnutí dalších požadavků na autentizaci pro překročení určitého množství neplatných pokusů.

U systémů typu elektronických vrátných, či registru příchodů a odchodů zaměstnanců, bývá zabezpečení zpravidla nejhorší. K pokusu o jejich prolomení dochází zřídkakdy, ale vyskytuje se zde jiný problém. Zaměstnanci jednotlivých firem si často půjčují své čipy, sdělují si své piny a tím obcházejí systém. Tento přístup zaměstnanců nutí firmy používat biometrické klíče, jako jsou otisky prstů či rozpoznávání tváře, duhovky. Tyto systémy jsou finančně nákladné, dochází k problémům (v případě poranění) a mezi největší komplikace pak patří etická otázka užívání lidského těla jako klíče. Zaměstnanci tedy často odmítají například svůj otisk prstu poskytnout a staví sebe i firmy to vypjaté situace.

Volbu správného autentizační klíče, jenž by byl využitelný, jak pro čistě digitální systémy (přihlášení k emailu), tak i pro registr zaměstnanců, či jako elektronického vrátného, jsme si vzali jako prioritu.

4.2.2 Volba zprostředkovatele

Aby bylo možno naplnit cíle bezpečné, etické, relativně rychlé (dostatečně pomalé) autentizace co nejvíce spjaté s daným uživatelem, bylo potřeba najít vhodné médium, které by vývoj něčeho takového vůbec umožnilo. Klasicky používané čipy a karty nepřipadaly v úvahu, z důvodu nemožnosti solit přenášený klíč a nutnosti používat pro každou komunikaci klíč stejný. Z toho plynou nevýhody, hlavně v případě vytvoření kopie tohoto klíče na základně jednoho jediného odposlechu přenosu třetí stranou. Biometrii jsme označili za bezpečnou, ale drahou a neetickou.

Inspirovali jsme se tedy minulostí a rozhodli se pro systém, který se snad už nikde nepoužívá, vylepšit. Dříve se ve školách k výdeji obědů používaly kartičky s čárovými kódy. My jsme se pro předání autentizačního klíče rozhodli použít QR kódy. A pro jejich uchování a přenos se rozhodli použít chytré telefony jakožto silné výpočetní zařízení, které nám umožní každou komunikaci učinit unikátní.

Zároveň nám smartphony umožní použít takových metod, které dokáží svázat klíč s daným zařízením. Při odcizení smarthphonu můžeme jednoduše (v případě, že uživatel používá antitheft) zařízení lokalizovat, zablokovat, popřípadě na něj zavolat. Samotná aplikace generující klíče může být chráněna heslem pro zvýšení bezpečnosti. Nebo dokonce můžeme naše zařízení používat jako správce účtů, kterého nosíme neustále s sebou. Možností je tolik, že si snad ani nedokážeme všechny výhody zatím uvědomit a plně jich využít.

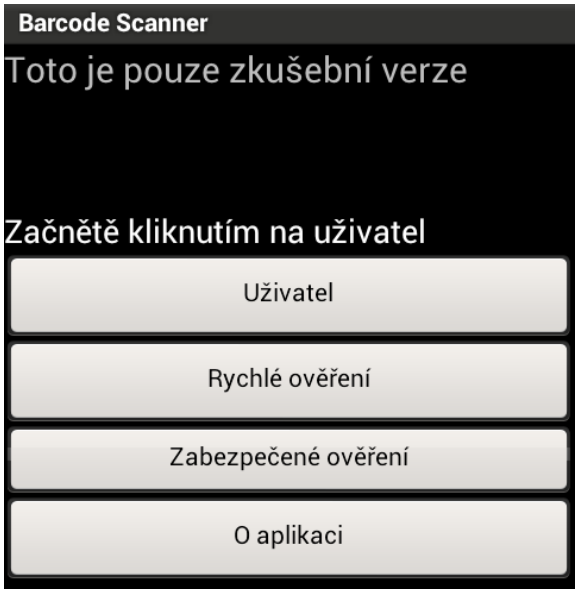
Jako každý systém má i ten námi navrhovaný slabiny, ale cílem naší práce je poukázat na nový způsob, který může být dostatečně bezpečný a ne vytvořit něco co bude okamžitě dokonalé. Na to je potřeba, aby systém byl nasazen do praxe a chyby spolu s nedostatky se vyladily v průběhu testovacího provozu, který už v průběhu psaní této práce probíhá v jedné firmě.

4.3 Klientská část

- Platforma Os Android
- Licence Apache II
- Verze 1.2.1 Demo
- Kompatibilita: Android api 15 a vyšší (Android 4.0 a vyšší)
- Oprávnění:
    - Nutnost povolit instalaci z neznámých zdrojů

4.3.1 Uživatelské rozhraní klienta

GUI se skládá z několika aktivit. Při prvním spuštění aplikace se zobrazí obrazovka s informacemi o aplikaci. Při dalších spuštěních se otevírá rovnou hlavní menu aplikace, ve kterém najdete tlačítka Uživatel, Rychlé ověření, Bezpečné ověření, O aplikaci



Obrázek 34 – Uživatelské rozhraní

- **Uživatel**

Ve verzi 1.2.1 Demo je možno vyzkoušet si funkci aplikace zadáním uživatelského jména 12345 a hesla 12345. To učiníte kliknutím na „Uživatel“ a následným vybráním položky

Přihlásit. Funkce Změna hesla a získat účet je v této verzi zakázána. Po přihlášení uživatele se načte aktivity a hlavním menu. Zde se zobrazí jméno přihlášeného uživatele a je možno si vyzkoušet ověření.

- **Rychlé ověření**

Po kliknutí na rychlé ověření se načte aktivita s náhledem QR kódu, ve kterém je uložen autentizační řetězec.

- **Zabezpečené ověření**

Nejdříve dojde k otevření CaptureActivity, jež má na starosti sejmutí QR kódu vygenerovaného serverem. Po sejmutí se automaticky otevře aktivita podobná té z rychlého ověření. Pouze vygenerovaný QR kód se liší.

- **O aplikaci**

Pod touto položkou se nachází aktivita s informacemi o aplikaci a nápovědou.

4.3.2 Programové řešení

Většina zdrojového kódu je psána v jazyce java. Programování pro OS Android se vyznačuje jistými odlišnostmi od klasické syntaxe, tyto rozdíly však není potřeba zmiňovat. Pro tvorbu GUI jsme využívali vestavěného GUI editoru, jenž je součástí Android pluginu pro vývojové prostředí Eclipse v kombinaci s úpravou generovaného kódu v xml. Pro neměnné řetězce a hodnoty jsme taktéž upřednostňovali zápis v podobě xml. Pro zprovoznění základních funkcí či například využívání vlastností projektu pro obsluhu kamery, jež je společností Google distribuován pod licencí Apache II bylo nutné provádět i zápisy do manifestu aplikace. Obrázky jsme pak ukládali podle standart pro Android resources. Pro ukládání informací jakou jsou například informace o uživateli, jsme použili pouhé ukládání do souborů, jež je na rozdíl od klasických java aplikací na Androidu dostatečně zabezpečeno proti čtení třetí stranou.

4.3.2.1 Práce s aktivitami

Jedna aktivita vždy znázorňuje jednu obrazovku aplikace. Tvoří ji uživatelské rozhraní (GUI) ukládané ve formě xml souboru a logika jazyka java využívající knihovny dle příslušné verze OS Android (v naší aplikaci se setkáte i s importem knihoven, které nejsou přítomny v Android API).

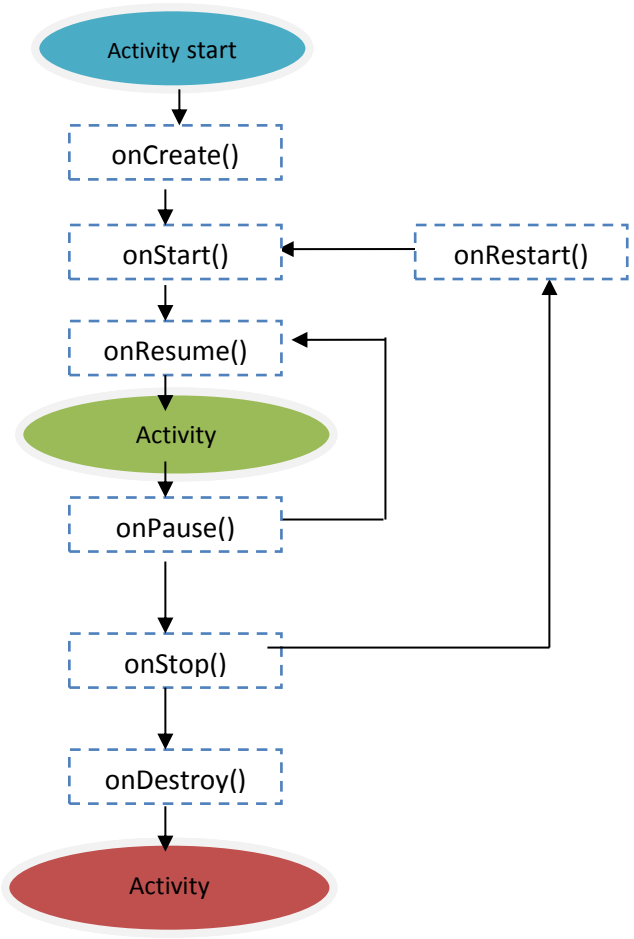
Každá aktivita má svůj vlastní životní cyklus, může být například na popředí a mít uživatelský vstup, může být viditelná třeba jen částečně nebo může být celá skryta na pozadí. Pro přechod mezi jednotlivými aktivitami se používají takzvané callbacky, které určují, v jaké části životního cyklu se daná aktivita nachází.

Základní metody:

- onCreate() – Tato metoda je volána okamžitě při vytváření aktivity. Používali jsme ji například k odkazování a definování GUI, nastavování hodnot, jenž mají být nastaveny ihned při vytváření aktivity, či například k volání metod, které prováděly potřebné operace po vytvoření aktivity. Jedná se o primární prostředek na práci s aktivitami viz obrázek 35.
- onStart() – Tato metoda je volána, když se aktivita stane viditelnou pro uživatele. onResume() - Volána, když aktivita začne dostávat uživatelský vstup. V tento moment je tedy na vrcholu zásobníku a není nijak překryta jinou aktivitou.
- onPause() – Je volána těsně před zamrznutím aktivity.
- onStop() - Zavolána, když aktivita přestává být viditelná pro uživatele.
- onDestroy() - Volána před zrušením samotné instance aktivity.
- onRestart() - Tato metoda je volána těsně předtím než aktivita, která byla zastavena, je znovu vykreslena.

```
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    setContentView(R.layout.activity_login);
    name=(EditText)findViewById(R.id.name);
    name2=(EditText)findViewById(R.id.name2);

    heslo_nahled=(TextView)findViewById(R.id.heslo_nahled);
    potvrdit=(Button)findViewById(R.id.potvrdit);

    setDefVal();

    Bundle extras = getIntent().getExtras();
    if (extras == null) {

        return;

    }
```

Obrázek 35 – Metoda onCreate()



Obrázek 36 – Cyklus aktivity

Každou aktivitu je nutno deklarovat v manifestu. Vstupní aktivita (zobrazující se startem aplikace) musí mít ve své deklaraci subtagy action a category v předepsaném tvaru. Na obrázku 37 můžete vidět deklaraci vstupní aktivity naší aplikace.

```
<activity
    android:name=".ActivityFirstStart"
    android:label="QR-Key" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Obrázek 37 – Deklarace vstupní aktivity

Volání aktivit jsme realizovali prostřednictvím intentů (záměrů). Pro pouhé zobrazení některé aktivity jsme používali metodu startActivity(Intent intent); pro komunikaci, vložení určitých hodnot do aktivity jsme používali metodu startActivityForResult (Intent intent String string);

```
Intent intent = new Intent(this,ActivityStart.class);
startActivity(intent);
Intent i = new Intent(this, ActivityFastMode.class);
i.putExtra("Value1", "Value2");
startActivityForResult(i, pom);
```

Obrázek 38 – Užití intentů

Jelikož jsme ve svém projektu využívali opensource projektu dodávaného společností Google, bylo nutné vyřešit komunikaci mezi naším projektem a volně distribuovaným projektem, který jsme nastavili jako knihovnu. V případě aktivit si to vyžádalo zvláštní ošetření v manifestu a úpravu hlavní aktivity (CaptureActivity) v distribuovaném projektu tak, aby po provedení příslušných operací došlo k jejímu ukončení a zobrazení aktivity z našeho projektu.

```
<activity
    android:name="com.google.zxing.client.android.CaptureActivity"
    android:configChanges="orientation|keyboardHidden"
    android:screenOrientation="landscape"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
    android:windowSoftInputMode="stateAlwaysHidden" >

    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>

    <intent-filter>
        <action android:name="com.google.zxing.client.android.SCAN" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Obrázek 39 – Ošetření manifestu

```
//získá sekvenci znaků načtenou z oskenováného QR kódu
CharSequence data;
data=resultHandler.getDisplayContents();

//získá řetězec ze sekvence znaků
String zpracuj;
zpracuj=data.toString();

//zobrazí zprávu o úspěšném načtení
Toast.makeText(CaptureActivity.this, "QR kó byl načten. Můžete pokračovat", Toast.LENGTH_LONG).show();

//Ulozi vysledek do Intentu
Intent intent = new Intent();
intent.putExtra("RESULT_STRING", zpracuj);
setResult(1, intent);

// ukončí CaptureActivity
finish();
```

Obrázek 40 – Upravená hlavní aktivita

4.3.2.2 Přihlášení se k aplikaci

Pro využití funkcí naši aplikace je potřeba se nejdříve přihlásit. Ve verzi 1.2.1 Demo je vytvořen na serveru testovací uživatel (jméno 12345; heslo 12345). Pokud se na klientské části přihlásíte jako tento uživatel, proběhne u serverové části autorizace úspěšně. V opačném případě bude přístup zamítnut.

Po zadání jména a hesla proběhne uložení uživatelského jména a MD5 hashe hesla do soubru. K ukládání a načítání souboru využíváme vlastností knihovny java.io. Konkrétně jsme použili následující importy.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
```

Pro generování MD5 hashe jsme používali metodu na obrázku 41. Aplikace umožňuje při drobném přenastavení používat i pokročilejší hashovací algoritmy jako je například SHA1, ale pro prvotní vývoj a demonstraci bohatě stačí MD5 hash, který je méně výpočetně náročný.

```
private String ziskejMd5Hash(String s) {
    MessageDigest m = null;

    try {
            m = MessageDigest.getInstance("MD5");
    }

    catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
    }

    m.update(s.getBytes(),0,s.length());
    String hash = new BigInteger(1, m.digest()).toString(16);

    return hash;
}
```

Obrázek 41 – Generování hashe

4.3.2.3 Generování QR kódu

Pro generování QR kódu používáme knihovny zXing . Hlavní metodu pro využití této knihovny máte na následujícím obrázku.

```
public void genQRcode (){
    Bitmap bmp;
    int bitmapWidth = 300;
    int bitmapHeight = 300;

    String context;
    String userid;
    String username;
    String userpassword;
    String usernamefilename;
    String userpasswordfilename;
    String md5fmchash;

    usernamefilename="jmeno";
    username =readFromFile(usernamefilename);
    userpasswordfilename="heslo";
    userpassword=readFromFile(userpasswordfilename);
    md5fmchash=ziskejMd5Hash(username+userpassword);
    userid=getUserId(username);

    context="IDFMC"+userid+md5fmchash;
    writeToFile(md5fmchash, "authstring");
    writeToFile(context,"qrstring");


    String data=context;

    com.google.zxing.Writer writer = new QRCodeWriter();
    String finaldata = Uri.encode(data, "UTF-8");
    BitMatrix bm;
    try {
        bm = writer.encode(finaldata, BarcodeFormat.QR_CODE, bitmapWidth,bitmapHeight);
                bmp= Bitmap.createBitmap(bitmapWidth, bitmapHeight,Config.ARGB_8888);

    for (int i = 0; i < bitmapWidth; i++) {
        for (int j = 0; j < bitmapHeight; j++) {
            bmp.setPixel(i, j, bm.get(i, j) ? Color.BLACK: Color.WHITE);
        }
    }
    iv.setImageBitmap(bmp);
    } catch (WriterException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

Obrázek 42 – Generování QR kódu

4.3.3 Logika aplikace

V následující části naleznete popis nejdůležitějších částí programu či základních postupů.

4.3.3.1 Rychlé ověření

Jak už bylo zmíněno, jedná se o demonstraci dnes používaných systémů, která užívá jen o něco jiných prostředků k dosažení téhož cíle za pomoci velmi podobných metod.

Nejdůležitější částí je samotné generování autentizačního klíče. Tento klíč je generován na základě uživatelského jména a otisku hesla ve formě MD5 hashe. Autentizační klíč je řetězec skládající se z několika částí. Pro použitou MD5 funkci je jeho délka 42 znaků (pro jiné funkce se pak délka řetězce liší). První až pátý znak řetězce označuje identifikátor akce. Pro rychlé ověření je tento identifikátor IDFMC (pro bezpečné ověření je identifikátor roven znakům IDSMC nebo například pro změnu hesla najdete identifikátor ve tvaru IDCPC).

Obrázek 43 - Identifikátor

Tento identifikátor serveru říká, jaké operace se na něm budou provádět, respektive o jaký mód se jedná.

IDFMC= IDentificator Fast Mode Client.

První dva znaky jsou u všech identifikátorů totožné. Jedná se pouze o označení, že jde o identifikátor. Následující dva znaky říkají, o jakou akci se jedná, a poslední znak určuje, jestli řetězec odesílá klientská část či část serverová (bíle na obrázku 43)

Následujících pět znaků je věnováno uživatelskému identifikátoru (červeně na obrázku 43). Jedná se o čísla v šestnáctkové soustavě. V přikládané verzi programu jsou rovny prvním pěti znakům MD5 hashe uživatelského jména. Avšak jejich tvar může být i jiný v závislosti na použitém systému udělování uživatelských id. Pod tímto id je uživatel veden na serveru.

Zbylých 32 znaků autentizačního klíče je věnováno autentizačnímu kódu (žlutě na obr. 43). Zjednodušeně řečeno prvních 10 znaků říká, kde hledat a jaký způsob ověřování použít. Ale nejpodstatnější část je ukryta právě v posledních třiceti dvou hexadecimálních číslicích řetězce. Ty jsou v případě rychlého ověření tvořeny MD5 hashem sloučeného řetězce skládajícího se z uživatelského jména a MD5 otisku uživatelského hesla.

K vytváření autentizačního klíče dochází přímo v aktivitě rychlého ověření na základě uložených hodnot o uživateli. Tento autentizační klíč se pak následně pomocí metody pro tvorbu QR kódů a knihovny zXing zobrazí jako QR kód, který je následně poskytnut serveru po přiložení ke čtečce serveru.

4.3.3.2 Bezpečné ověření

Bezpečné ověření je demonstrací námi navrhovaného systému. Logika je velmi podobná jako u rychlého ověření. Následující popis tedy obsahuje hlavní rozdíly.

Pro generování autentizačního klíče byl použit následující proces. Při spuštění zabezpečeného ověření dojde k otevření CaptureActivity a na obrazovce se objeví vstup z kamery přístroje.

Následně musí uživatel sejmout QR kód který mu poskytne serverová část. Tento sejmutý QR kód obsahující náhodný řetězec, slouží, jako sůl kterou zahrnujeme do výpočtů. Po sejmutí dojde ihned k přechodu na aktivitu, která zobrazí QR kód s autentizačním klíčem.

Tvar autentizačního klíče je následovný. Prvním pět znaků je rovno identifikátoru akce tedy IDSMC následuje pět znaků uživatelského identifikátoru. Hlavní odlišnost od rychlého ověření je tedy v autentizačním kódu.

Autentizační kód se skládá z a autentizačního kódu tvořeného podle stejného pravidla jako u rychlého ověření, ke kterému je přidáno náhodně generované, hexadecimální, 160bitů dlouhé číslo získané oskenováním QR kódu ze serverové aplikace. Z této hodnoty je pak následně vytvořen md5 otisk.

4.4 Serverová část

- Optimalizováno pro platformu Linux Ubuntu
- Licence Apache II
- Verze 1.2.1 Demo
- Kompatibilita: Testováno na Xubuntu a Mac OS X
- Oprávnění: Přístup k webkameře, zápis na disk
- Nutnost mít nainstalován balíček java

4.4.1 Uživatelské rozhraní

Vývoj celé serverové části programu probíhal ve vývojovém prostředí Netbeans. Pro samotnou tvorbu grafického uživatelského rozhraní jsme používali pluginu dodávaného spolu s distribucí Netbeans postaveného na knihovnách swing, která umožňuje GUI tvořit velmi příjemným způsobem.

4.4.1.1 Hlavní menu

V hlavním menu máte na výběr z několika možností. Po kliknutí na příslušnou volbu se otevře další okno, které vykonává potřebné služby. Některé butony jsou v dodávané verzi zakázány. To z důvodu především snadné distribuce (pro demonstraci, zbytečné funkce jsme zakázali a raději se soustředili na plnou funkčnost nezbytného).

4.4.1.2 Rychlé ověření

Jak už bylo několikrát zmíněno v předchozích částech práce, jedná se o standardně používaný model poskytující velmi slabé zabezpečení. Po otevření tohoto módu dojde zobrazení nového okna, ve kterém se načte, po kliknutí na button čti, stream videa z webkamery.

Samotné zprovoznění webové kamery a vykreslování snímaného obrazu do jpanelu bylo asi nejobtížnější částí našeho projektu.

Toto okno běží do doby, než je před objektivem kamery rozpoznán QR kód přiložený uživatelem pomocí klientské části.

Jakmile dojde k rozpoznání QR kódu na snímaném obrazu webkamery nastane uzavření okna, a následně se otevře nové okno v závislosti na tom, zda proběhla autorizace úspěšně, či byl uživateli přístup zamítnut.

- Odmítnutí uživatele

  V případě, že dojde k vyhodnocení autorizace jako neplatné, zobrazení se dialog přístup zamítnut.

- ✓ Přijetí uživatele

V případě vyhodnocení, že přístup je oprávněný, proběhne autorizace úspěšně a dojde k zobrazení okna s ukázkovými daty, ke kterým jsme omezili přístupová práva

4.4.1.3 Bezpečné ověření

Po zvolení bezpečného modu dojde k zobrazení okna, ve kterém se v imagepanelu vykreslí QR code generovaný za použití random funkce, sloužící jako následný soil, ve výpočtech probíhajících na serveru i klientovi, zvyšující zabezpečení celého systému.

Vedle QR kódu, po následném kliknutí na button pro čtení, dojde stejně tak, jako u rychlého ověření ke spuštění kamery a streamování jejího obrazu.

Celý proces probíhá obdobně stejně jako rychlého ověření. Pro uživatele je jen jediná odlišnost, a to v mezikroku, při kterém klientskou částí sejme výše zmínění QR kód, aby mohlo dojít ke konkrétním výpočtům.

4.4.2 Programové řešení

Naše ukázková aplikace, sloužící jako server, byla primárně vyvíjena pro OS Linux Ubuntu. Platforma Linux vyplynula z komplikací, které nastávaly v případě použití OS Windows, a to především při práci s prostředky umožňující implementaci webové kamery.

4.4.2.1 Formát informací uložených na serveru

Na serveru je potřeba ukládat informace o uživateli, jako je například jeho otisk hesla, uživatelské jméno a identifikátor, popřípadě je možné přidat další informace. V přiložené verzi je vytvořen pouze jeden ukázkový uživatel, na kterém probíhá demonstrace základních funkcí systému. Tento uživatel je deklarován přímo ve zdrojovém kódu systému.

Při testování našeho systému jsme používali ukládání těchto dat do souborů. Mohli jsme tedy vytvořit, v administrační části, nespočet uživatelů. Tato logika se již pak neliší od bezpečnějšího a rychlejšího zápisu do komplexních databází.

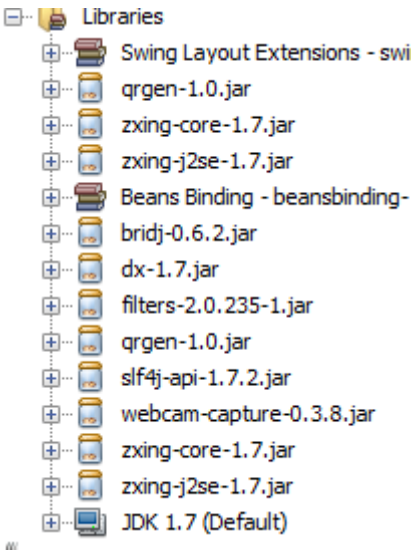
Pokud by se měla aplikace běžně používat, bylo by nutné neukládat tyto údaje do souborů, nýbrž použít lepších vlastností databází, tak jako je to u většiny dnešních systémů. Pro demonstraci nám tato varianta přišla jako zbytečná komplikace, která by stěžovala její distribuci.

Ať už je forma, jíž je samotná informace na serveru uložena, jakákoliv, musí vždy splňovat námi stanovené kritéria. Všechna data musí být uložena pod identifikátorem uživatele. Heslo musí být uchováváno ve tvaru MD5 hashe, u kterého bylo jako soil užito jeho uživatelské heslo. Tento tvar je totiž mnohem odolnější proti kryptoanalýze.

4.4.2.2 Čtení QR kódu

Čtení QR kódu nebo spíše práce s webovou kamerou byla asi nejnáročnější částí našeho projektu. Z důvodu nefunkčnosti webové kamery na OS Windows jsme se přešli na Linux Ubuntu, kde vše fungovalo bez větších obtíží.

Pro správu kamery a testování jejího streamu na přítomnost barcodu bylo nutné importovat do našeho projektu celou škálu externích knihoven, které nejsou standardně součástí distribuce java. Na obrázku 44 můžete vidět všechny importované knihovny, z nichž jedna jen slouží pro generování QR kódu. Zbytek byl nutný pro čtení.



Obrázek 44 – Knihovny

Jelikož probíhá na serveru více operací paralelně bylo potřeba vytvořit nové vlákno programu, které spouštíme pomocí jeho instance a metody start().

```
229         //Spusti vlakno skenování scanning
230         Scanning scan = new Scanning();
231         scan.start();
```

Obrázek 45 – Spuštění scanování

Hlavní část vlákna je na obrázku 46. Můžete vidět, že po spuštění vlákna dojde ke startu webové kamery a k vykreslování jejího streamu přímo v GUI. Také v tomto vláknu běží smyčka, která prohledává snímaný obraz na přítomnost barcodu.

```
236      class Scanning extends Thread {
         public void run() {
238              //Implementace skenování
239
240          Webcam webcam = Webcam.getDefault();
241
242          webcam.open();
243
244          do{
245           BufferedImage image = webcam.getImage();
246           ImageIcon icon = new ImageIcon(image);
247           jLabel4.setIcon(icon);
248
249           LuminanceSource source = new BufferedImageLuminanceSource(image);
250           BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));
251           MultiFormatReader barcodeReader = new MultiFormatReader();
252
253           Result result;
254
255           try {
256               result = barcodeReader.decode(bitmap);
257               finalResult = String.valueOf(result.getText());
258           } catch (NotFoundException ex) {
259               }
260           }while("Chyba převodu".equals(finalResult));
261
262          webcam.close();
```

Obrázek 46 – Hlavní část vlákna

4.4.2.3 Generování QR kódu

Samotné generování QR kódu vyžadovalo zakomponovat do projektu knihovnu, která tuto operaci umožňuje (konkrétně knihovnu qrgen-1.0.jar). A provést importy:

- import net.glxn.qrgen.QRCode;
- import net.glxn.qrgen.image.ImageType;

Konkrétní zápis zdojového kódu pak můžete vidět na obrázku 47.

```
        data=(retezec1);

        ByteArrayOutputStream out = QRCode.from(data).to(ImageType.PNG).stream();
        byte[] bytes = out.toByteArray();
        ByteArrayInputStream bis = new ByteArrayInputStream(bytes);

        Iterator<?> readers = ImageIO.getImageReadersByFormatName("png");
        ImageReader reader = (ImageReader) readers.next();
        Object source = bis;

        ImageInputStream iis = ImageIO.createImageInputStream(source);
        reader.setInput(iis, true);

        ImageReadParam param = reader.getDefaultReadParam();
        Image image = reader.read(0, param);

        ImageIcon icon = new ImageIcon(image);
        jLabel2.setIcon(icon);

    } catch (IOException ex) {
        Logger.getLogger(SMode.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Obrázek 47 – Generování QR kódu – server

4.4.2.4 Generování hodnoty soil

Pro generování dostatečně velkého sub-náhodného čísla jsme použili random funkci knihovny java.util.Random, která je součástí standardní java distribuce. Provedení můžete vidět na obrázku 48

```
try {
   String data;

   int cislo=0;
   String retezec1="0", pom="0";

  for(int i=0; i < 32; i++){

    cislo = (int)Math.round(Math.random()*15 +0 );
    retezec1 = Integer.toHexString(cislo);

  if (i==0){
        pom=retezec1;
    }
    else{
        retezec1=retezec1+pom;
        pom=retezec1;
    }
}
```

Obrázek 48 – Generování náhodného čísla

Získaný String následně posíláme třídě obsluhující autentizaci a autorizaci, která jej zahrne do svých výpočtů (viz obr. 49).

```
Authorization_1 smode = new Authorization_1();
accessenable=smode.setAccessValues(finalResult, randomStr);
```

Obrázek 49 – Odeslání stringu třídě

4.4.3 Logika programu

4.4.3.1 Rychlé ověření

Při rychlém ověření se spustí webkamera a čeká, dokud nerozpozná QR kód. Jakmile dojde k rozpoznání, zpracováváme hodnoty v něm uložené jako řetězec (string). Tento string je brán jako autentizační klíč.

Nejdříve dochází k testování, zda příslušný kód pochází skutečně z klientské aplikace. Testujeme správnou délku příslušného stringu. V případě, že string má správnou délku, rozložíme jej na tři substringy a testujeme, zda první substring (prvních pět znaků autentizačního klíče) odpovídá identifikátoru akce, a to konkrétně IDFMC (Identifikator Fast Mode Client- identifikátor rychlého ověření klient).

V případě, že jsou předchozí podmínky splněny, dochází k testování existence uživatele na základě jeho uživatelského id, které je z autentizačního klíče získáno jako druhý substring.

Pokud byl uživatel rozpoznán, dojde k použití aktivní části autentizačního klíče, (třetí substring-32 hexadecimálních číslic rovných MD5 hashi uživatelského jména sloučeného s hashem uživatelského hesla) který jsme nazvali autentizační kód. Tento autentizační kód si na základě již ověřeného uživatelského id vyžádá serverová aplikace z její databáze.

Autentizační kód z klienstské části se porovná s autentizačním kódem uloženým na serveru. Když dojde ke shodě, systém potvrdí autentizaci uživatele jako úspěšnou a provede autorizaci.

V přiložené demonstrační aplikaci dojde, v případě úspěšného autorizovaného pokusu, k otevření okna s jistými informacemi (jde o generovaný text Lorem ipisum). Snadno si lze představit, že tento systém by mohl být například přístupovým bodem k emailové schránce nebo by například sloužil k otevření dveří budovy.

Tento systém má obrovskou slabinu zakomponovanou v samotných kořenech tohoto rychlého, dnes běžně používaného ověření. QR kód, a v něm uložený autentizační klíč, je statický a neměnný. Pro každou jednotlivou komunikaci je užit stejný klíč, který v případě odposlechnutí, zkopírování nebo zcizení znamená narušení bezpečnosti.

Jednoduše si toto tvrzení lze ověřit vytvořením kopie QR kódu generovaného klientskou částí. Po přiložení ke čtečce serveru, pokud nedošlo k poškození kopie, bude úspěšně autorizována i tato kopie.

4.4.3.2 Bezpečné ověření

Bezpečné ověření je demonstrací námi navrhovaného systému, který vnáší do oblasti bezpečnostních systémů jisté novum. Naší přidanou hodnotou je využití výpočetně silného zařízení schopného užívat soil. Tímto můžeme zajistit unikátnost každého přenosu.

Aby bylo možné používat náš systém k vyššímu zabezpečení, bylo nutné vytvořit postup, který by umožňoval výměnu informací mezi klientem a serverem. Předávanou informací je právě soil, který musí být stejný pro server i klienta. K přenosu hodnoty sloužící jako soil jsme použili obdobný způsob jako u samotného procesu ověřování, a to QR kódů a jejich čtečky na straně klienta.

4.5 Alternativy

Zvolený způsob přihlášení k aplikaci umožňuje jednomu uživateli autorizaci i s jinými zařízeními. Vždy se stačí na jakémkoliv smarthphonu přihlásit do naší aplikace pod vlastním uživatelským jménem a heslem.

Navrhli jsme ještě jednu alternativu, která však není součástí přikládaného programu, ale přesto bychom ji rádi zmínili. Při generování autentizačního klíče dochází k vytváření uživatelského id. Toto id je v přikládané verzi rovno prvním pěti znakům z MD5 hashe uživatelského jména. Toto id je však možno nahradit id, které nebude počítáno na základě uživatelského jména, ale bude generováno náhodnou funkcí na serveru. V klientské části pak nebude stačit pouhé zadání hesla a jména, ale bude třeba získat ještě správný tvar id. Toho lze docílit tak, že při vytváření nového účtu v serverové části, server vygeneruje QR kód obsahující uživatelské id, uživatelské jméno a heslo. Po sejmutí tohoto QR kódu se id uloží do paměti zařízení. Tehdy bude možné se pod daným účtem autorizovat pouze z jednoho zařízení, které bylo použito při samotném vytváření účtu. Pokud by se id sdělovalo přímo, bylo by jej možné zadat klasicky jako uživatelské jméno a heslo, postrádalo by toto zabezpečení smysl, jelikož by mohlo docházet k přenášení id na další zařízení. Proto je důležité střežit tvar id.

Přiložená verze má taktéž zakázanou změnu hesla uživatele. Ovšem postup je obdobný jako při autorizaci. Po kliknutí v klientské části v hlavním menu na Uživatel a vybráním možnosti Změnit heslo dojde k výzvě pro zadání stávajícího uživatelského jména, hesla a k zadání hesla nového. Následuje vygenerování QR kódu s instrukcemi pro server, který po rozpoznání požadavku na změnu hesla, za předpokladu správně zadaného uživatelského jména a hesla stávajícího, heslo změní.

Systém je navrhnut tak, aby jej bylo snadné rozšiřovat a obměňovat. Například identifikátor akce (viz následující část) předem počítá s možností budoucího provozování serverové aplikaci na smarthphonech či tabletech s platformou Android.

Na serverové aplikaci je i možnost manipulovat s úložištěm. Soubory s informacemi mohou být ukládány pomocí databáze, například na webový server, což otvírá nové možnosti.

Ověření může probíhat také bezkontaktně, například pomocí bluetooth či wifi, což dokáže snížit náklady na naprosté minimum.

Variant je zkrátka hodně a variabilita programu je až neskutečná, což se nám potvrzuje při spolupráci s firmou, která by ráda obdobný systém používala pro registr příchodů a odchodů svých zaměstnanců. S touto firmou pracujeme na verzi, jejíž pořizovací a udržovací náklady jsou téměř nulové.

Závěr

Zadání naší práce jsme bezezbytku naplnili, neboť všechny cíle, jež byly vytyčené, se nám podařilo dosáhnout. Navrhli jsme a vytvořili funkční program, který lze použít jako zabezpečovací systém a který, po několika málo dodatečných úpravách, může být uveden na trh, kde si jistě vydobude místo mezi svými konkurenty, pro něž bude více než zdatným soupeřem.

Důkazem pro toto naše tvrzení je fakt, že již ve fázi testování našeho programu jsme navázali spolupráci s Půjčovnou lodí Sudoměřice, kde jsme dostali zpětnou vazbu a zajistili si reklamu. Ta se nakonec ukázala být důležitější, než jsme původně mysleli, protože vedla k nabídce ze strany Pneuservisu Hodějovice, na vytvoření obdobného systému, registru příchodů a odchodů. Tuto příležitost nadále vylepšovat náš program jsme nemohli odmítnout.

K našim úkolům jsme si také rovněž zakomponovali i něco navíc. V aplikaci jsme vytvořili celkem dva módy umožňující autentizaci. Jedním z nich je rychlé ověření, které jak již je z názvu patrné, funguje velmi rychle, ovšem za cenu ztráty téměř veškeré bezpečnosti. Naproti tomu ověření bezpečné přidává jeden nutný krok autentizace navíc, který ale celé zabezpečení posouvá na novou úroveň. Díky tomuto rozdělení činíme z našeho programu nástroj, na němž můžeme upozornit na ne-úplně ideální stávající zabezpečení, a zároveň nastiňujeme způsob, jakým lze některá bezpečnostní rizika minimalizovat, či dokonce odstranit.

Z celkového hlediska musíme hodnotit celou naši středoškolskou činnost jako úspěšnou. Nabyli jsme mnoho nových poznatků o oblasti, která je centrem našeho zájmu, neboť v budoucnosti nás informatika bude živit. Získali jsme zkušeností s vypracováváním podobných prací a nahlédli jsme do komerčního světa a, ze všeho nejdůležitější, navázali jsme kontakt s VUT Brno prostřednictvím Ing. Jana Hajného, z jehož strany přišla cenná nabídka na pokračování ve spolupráci i po dokončení tohoto projektu.

Seznam obrázků

a) Použitá literatura a zdroje

[1] SINGH, Simon. *Kniha kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii*. Praha: Dokořán, 2003, 382 s. ISBN 80-865-6918-7.

[2] PIPER, Fred a Sean MURPHY. *Kryptografie. 1. vyd. v českém jazyce*. Překlad Pavel Mondschein. Praha: Dokořán, 2006, 157 s. ISBN 80-736-3074-5.

[3] Cryptography. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-03-12]. Dostupné z: http://en.wikipedia.org/wiki/Cryptography

[4] STALLINGS, W. *Cryptography and network security*. Vyd. 1. New Jersey: Prentice-Hall, 1999, 569 s. ISBN 01-386-9017-0.

[5] DOSTÁLEK, Libor, Marta VOHNOUTOVÁ a Miroslav KNOTEK. Velký *průvodce infrastrukturou PKI a technologií elektronického podpisu*. 2., aktualiz. vyd. Brno: Computer Press, 2009, 542 s. ISBN 978-80-251-2619-6.

[6] HANÁČEK, Petr a Jan STAUDEK. *Bezpečnost informačních systémů: metodická příručka zabezpečování produktů a systémů budovaných na bázi informačních technologií*. Vyd. 1. Praha: Úřad pro státní informační systém, 2000, 127 s. ISBN 80-238-5400-3.

[7] BURDA, K. *Bezpečnost informačních systémů*. 1. Brno: FEKT VUT Brno, 2005. s. 1-100

[8] QR Kódy [online]. 2009 [cit. 2013-03-24]. Dostupné z: http://www.qr-kody.cz

[9] What Is A QR Code [online]. 2013 [cit. 2013-03-12]. Dostupné z: http://www.whatisaqrcode.co.uk/

[10] QR Code. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-03-08]. Dostupné z: http://en.wikipedia.org/wiki/QR_code

[11] Smartphone. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-03-12]. Dostupné z: http://en.wikipedia.org/wiki/Smartphone

[12] UJBÁNYAI, Miroslav. *Programujeme pro Android*. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.

[13] Android (operating system). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-03-11]. Dostupné z: http://en.wikipedia.org/wiki/Android_(operating_system)

[14] MURPHY, Mark L. Android 2: průvodce programováním mobilních aplikací. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.

b) Loga k poděkování