VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

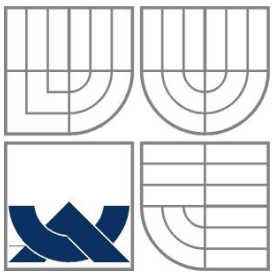
NÁSTROJ PRO PODPORU AGILNÍHO VÝVOJE SOFTWARU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

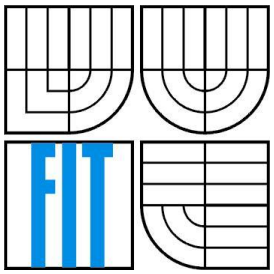
AUTOR PRÁCE
AUTHOR

Bc. PAVEL VEVERKA

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO PODPORU AGILNÍHO VÝVOJE SOFTWARU

AGILE DEVELOPMENT SOFTWARE SUPPORT TOOL

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE Bc. PAVEL VEVERKA
AUTHOR

VEDOUCÍ PRÁCE doc. RNDr. JITKA KRESLÍKOVÁ, CSc.
SUPERVISOR

BRNO 2014

Abstrakt

Cílem této práce je navrhnout systém pro podporu agilního vývoje softwaru ve virtuálním týmu. Jsou zde uvedeny metodiky agilního vývoje a některé dostupné nástroje pro jeho podporu. Aby mohl být systém použit v širokém spektru zařízení, je v práci uveden také teoretický rozbor multiplatformních možností vývoje aplikací. V dalších částech je popsán návrh a implementace pomocí webových technologií. Na závěr je systém demonstrován na modelovém příkladu a jsou navržená jeho rozšíření.

Abstract

Purpose of this thesis is to design system to support agile software development in virtual teams. Agile development methodology and some available tools to support it are listed here in this thesis, in order to use the system in a wide range of devices, theoretical analysis of multi-platform application development options are listed. The following sections describe the design and implementation witch using web technologies. At the end the system is demonstrated on model situation and extension its designed.

Klíčová slova

Project management, Agilní metodiky, Multiplatformní vývoj, Virtuální tým, Brainstorming, Mobilní platformy

Keywords

Project management, Agile methods, Multiplatform development, Virtual team, Brainstorming, Mobile platforms

Citace

Veverka Pavel: Nástroj pro podporu agilního vývoje softwaru, diplomová práce, Brno, FIT VUT v Brně, 2014

Nástroj pro podporu agilního vývoje softwaru

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Pavel Veverka
25. května 2014

Poděkování

Rád bych zde poděkoval doc. RNDr. Jitce Kreslíkové, CSc. za ochotu a čas, který mi věnovala při poskytování odborných rad a připomínek během tvorby této práce. Také bych chtěl poděkovat společnosti TokBox Inc. za to, že mi vyšla vstříc a poskytla mi aplikační klíč s prodlouženou zkušební dobou pro využívání jejich služeb.

Obsah

1 Úvod

Informační technologie jsou jedním z nejrychleji rozvíjejícím se odvětvím. Neustálým zlepšováním prochází nejenom hardwarová stránka, ale i postupy a metody vývoje softwaru. Na tvorbě informačních systémů pracuje mnoho týmů po celém světě a čím dál víc jich je virtuálních, to znamená, že jejich členové se mnohdy nikdy nepotkají osobně. Speciálně pro vývoj softwaru se začaly, počátkem devadesátých let dvacátého století, objevovat metodiky, které se později seskupily pod jeden název a to Agilní metodiky vývoje softwaru. Nyní se těší veliké oblibě a virtuální týmy je, pro jejich typ výsledného produktu, využívají také. Velký rozmach zažívají také mobilní zařízení. Jejich stále se zvyšující výkonnost společně s možností mít je stále u sebe, z nich dělají dobrého pomocníka nejen pro virtuální týmy.

Cílem diplomové práce je seznámit se s Agilními metodikami vývoje softwaru. Prozkoumat dostupné softwarové nástroje na podporu těchto metodik. Dále se seznámit s možnostmi multiplatformního vývoje softwaru a na základě zjištěných informací navrhnout nástroj pro podporu agilních metodik pro virtuální týmy. Na základě specifikovaných požadavků a návrhu systému poté vytvořit multiplatformní aplikaci pro podporu virtuálních týmů. Tuto aplikaci otestovat na modelové situaci a vyhodnotit výsledky testování.

Následující kapitola je věnována Agilním metodikám vývoje softwaru. Nejprve je stručně popsána jejich historie a následně jsou rozebrány principy rodiny metodik Crystal, podrobněji pak metodika Crystal Clear, poté metodika Scrum, Extrémní programování a Kanban.
Další kapitola se věnuje softwarovým nástrojům pro podporu Agilních metodik, jejich společným vlastnostem, rozdílům a nedostatkům, zvláště z pohledu virtuálních týmů.
Následné jsou rozebírány možnosti multiplatformního vývoje se zaměřením na mobilní platformy.
V páté kapitole je vypracován návrh nástroje pro podporu Agilních metodik v prostředí virtuálních týmů. Kapitola návrhu obsahuje specifikaci požadavků, návrhové diagramy a návrh uživatelského rozhraní. V šesté kapitole je dán prostor samotné implementaci aplikace, použitým technologiím, implementovaným částem, technickým požadavkům systémů a testování. V předposlední kapitole je popsané ovládání aplikace a provedena demonstrace použití na modelové situaci. V závěru jsou pak zhodnoceny výsledky a jsou diskutována možná vylepšení a rozšíření.
Diplomová práce vychází ze semestrálního projektu, který dal především teoretický základ pro tuto práci a kde byl vytvořen předběžný návrh aplikace.

2 Agilní metodiky vývoje softwaru

Tato kapitola se zabývá agilním přístupem k vývoji softwaru. Nejprve je zde popsána historie agilního vývoje, kde se čtenář mimo jiné doví o vzniku Manifestu agilního vývoje softwaru a jeho hlavních principů. Následně jsou uvedeny podstatné metody, které patří do množiny metodik agilního vývoje softwaru.

2.1 Historie

Agilní metody jsou reakcí na rychle se rozvíjející technologie v oblasti informačních technologií, zvláště pak na tradiční "heavyweight" metody vývoje softwaru. Tyto metody byly zejména založené na získávání všech požadavků na vyvíjený systém pouze na začátku vývoje. Dalším krokem pak byl obsáhlý a vysokoúrovňový návrh, následovaný implementací a testováním. Tento postup se na začátku poloviny devadesátých let dvacátého století začal jevit jako nedostačující a v některých případech i nepoužitelný. Výsledkem bylo, že začaly vznikat nové metody pro vývoj softwaru označované jako odlehčené, neboli "lightweigh".

2.1.1 Manifest Agilního vývoje softwaru

Vznik manifestu je datován na rok 2001 [1], avšak prvotní impuls vznikl o rok dříve. Na konferenci o "lightweight" metodách vývoje softwaru, kterou pořádal Kent Beck, tvůrce konceptu Extrémního programování, vzniklo uskupení s názvem Agile Alliance, spojující společné myšlenky těchto metod do jednoho konceptu. Pro ucelení a definování byl sepsán Manifet Agilního vývoje softwaru (Manifesto for Agile Software Development). Manifest se skládá ze čtyř základních hodnot, kde je vyzdvihována levá strana před tou pravou, která však není popírána, a dvanácti principů, které rozvíjejí základní hodnoty.

Hodnoty

- **Jednotlivci a interakce** před procesy a nástroji.
- **Fungující software** před vyčerpávající dokumentací.
- **Spolupráce se zákazníkem** před vyjednáváním o smlouvě.
- **Reagování na změny** před dodržováním plánu.

Principy

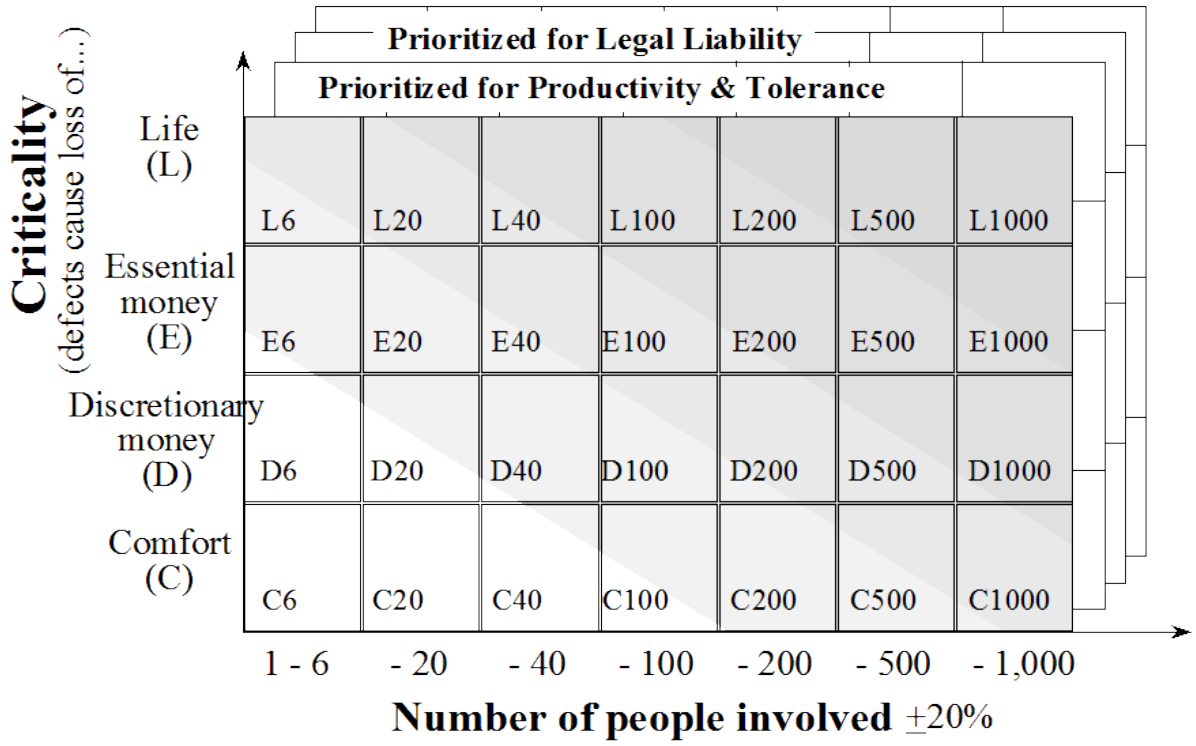
- Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
- Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
- Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
- Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.
- Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
- Nejúčinnějším a nejefektivnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
- Hlavním měřítkem pokroku je fungující software.
- Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
- Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
- Jednoduchost--umění maximalizovat množství nevykonané práce--je klíčová.
- Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.
- Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším, a následně koriguje a přizpůsobuje své chování a zvyklosti.

Z hodnot a principů manifestu jsme schopni určit charakteristiku agilního vývoje softwaru. Základem je malý efektivní tým, jehož součástí je i zákazník, který je aktivně zapojen do projektu. Členové týmu jsou dobře motivováni, respektují se a jsou ochotni se od sebe navzájem učit. Tým sám sebe organizuje, komunikuje formou osobní konverzace a neustále hodnotí svůj postup ve snaze zlepšování se. Vývoj vychází z kvalitního návrhu a postupuje po malých krocích, které jsou přístupné k libovolným změnám v požadavcích, a klade důraz na jednoduchost. Nejdůležitější pro agilní vývoj je kvalitní výsledek, který vyhovuje hlavně zákazníkovi a je mu dodáván průběžně.

2.2 Crystal

Rodina metod Crystal byla vytvořena v roce 1992. Hlavní zásluhu měl Alistar Cockburn a tím odstartoval evoluci metodik vývoje softwaru. Jak už bylo řečeno, Crystal není pouze jednou metodikou, ale rodinou více metodik. Každá metoda patřící do této rodiny sdílí stejné základy. Prvním je silná orientace na lidský faktor v projektech anglicky "Human-powered". Přestože některé jiné metodologie kladou také důraz na lidský faktor, často mohou být spíše procesně nebo architektonicky orientované. Naproti tomu Crystal se snaží dosáhnout úspěchu v projektu primárně pomocí posílení činností zúčastněných osob. Další základní vlastností je "ultralehkost" z anglického "ultralight". Což znamená, že ať je velikost a priorita projektu jakákoliv Crystal se snaží snížit režii projektu, jako je papírování a byrokracie, na takovou úroveň, aby byla pro projekt praktická a výhodná. Poslední vlastností je pružnost anglicky "Stretch-to-fit". Principem je, že pro projekt zvolíme metodiku o trochu menšího rozsahu, než si myslíme, jaká je třeba, a pak když je nutno ji rozšiřujeme. Obecně platí, že natahování je jednodušší, bezpečnější a efektivnější než odřezávání [5].

Nyní byly vysvětleny základní vlastnosti společné pro celou rodinu Crystal, avšak hlavní myšlenkou je, že různé projekty potřebují různé metodiky. Proto Crystal rozděluje projekty do třech hlavních dimenzí. Prvním faktorem dělení projektů je komunikační zatížení, respektive počet osob pracujících na projektu. Druhým je pak kritičnost systému, ukazující potencionální škody, které může projekt přinést. Posledním je pak priorita projektu. Tyto faktory jsou znázorněné v grafu Graf 2.1. Každá buňka znázorňuje třídu projektů s podobným komunikačním zatížením a bezpečím. To nám pomůže při výběru metodiky. Posouváním ve vertikálních pruzích vznikají jednotlivé metodiky rodiny Crystal dle počtu zúčastněných osob 2-6 pro Crystal Clear, 6-20 pro Crystal Yellow, 20-40 pro Crystal Orange a tak dále. Posun v horizontálním směru je označováno jako "tvrdnutí" metodiky.



Graf 2.1 - Rodina metodik Crystal [6]

2.2.1 Crystal Clear

Tato metoda je první metodikou z rodiny Crystal. Je určená pro malé týmy ve velikosti 2-6 osob. Jelikož mnoho metodik malé týmy zavrhnou pro jejich složitost, Crystal Clear se nesnaží být nejlepší metodikou, ale dostatečnou metodikou pro jakýkoliv tým. Princip metody se dá shrnout do jednoduchého popisu. Základem je hlavní vývojář s ostatními členy týmu, tabulemi a flipcharty v jedné místnosti. Mají přístup ke všem informacím, hlavním uživatelům a nejsou rušeni vnějším okolím. Pravidelně dodávají použitelný kód, testují a hodnotí svůj postup [7].

Vlastnosti metodiky

Ve spolupráci s týmy využívající Crystal Clear, Alistar Cockburn stanovil sedm klíčových vlastností anglicky "properties" a z nich byly vybrány tři hlavní, na které se metodika soustředí.

- Časté doručování

  Časté doručování je vlastnost označována jako nejdůležitější vlastnost každého projektu. Jádrem je předávání funkčního kódu, který je řádně otestován, každých pár měsíců. Když se tato vlastnost dodržuje přináší několik opravdu častých výhod. První výhoda je ta, že sponzor dostává kritickou zpětnou vazbu o postupu týmu. Další výhodou je poskytnutí šance uživateli

aby otestoval a zjistil, zda jeho původní požadavky jsou opravdu to, co potřebuje a může své požadavky upravit. Dále se pak tým udržuje soustředěný a pomocí postupných úspěchů i motivovaný.

- Reflektivní zlepšování

  Pro každý tým je velice důležité, aby pravidelně věnoval několik hodin pro sebereflexi a analýzu dosavadního postupu. Stejně tak jako sebereflexe, je pak důležité následné zlepšování.

- Osmotická komunikace

  Jedná se o podprahové vnímání komunikace mezi jednotlivými členy týmu. Jelikož je tým v jedné místnosti, každý člen na pozadí slyší komunikaci mezi kolegy, která se ho přímo netýká. Informace se tak šíří mezi všemi členy týmu, i když nejsou aktivními diskutéry.

- Osobní bezpečí

  Osobní bezpečí je možnost otevřeně mluvit o jakémkoliv problému, bez hrozby trestu. Například sdělení manažerovi, že je projekt špatně navržen. Osobní bezpečí je prvním krokem k důvěře. Bez možnosti otevřeně mluvit, by nebyl schopen odhalit a opravit slabiny.

- Zaměřenost

  Zaměřenost je znalost toho, co je cílem práce a možnost mít dostatek času na provedení práce.

- Jednoduchý přístup k expertním uživatelům

  Zajištění přístupu expertních uživatelů k týmu. Poskytnutí místa k testování, rychlé zpětné vazby o kvalitě produktu, návrhu a aktuálnost.

- Technické prostředí s automatickými testy, konfiguračním managementem a častou integrací

  Automatické testy umožňují rychle zrevidovat část kódu a odhalit v něm chybné části. Konfigurační management umožňuje lidem asynchronně zkontrolovat jejich práci. Zabalit ji do konfiguračního balíku a vrátit se k ní pokud nastane problém. Toto umožňuje tvorbu kódu jak samostatně, tak společně. Častá integrace pomáhá taktéž velmi brzo odhalit chyby. Čím častěji se systém integruje, tím se chyba rychleji odhalí.

Role metodiky

Metodika Crystal Clear definuje osm rolí [7]. První čtyři role musejí být obsazené různými lidmi.

- Sponzor (Sponsor)
  Je zodpovědný za financování projektu, měl by udržovat dlouhodobý směr a vytvářet vnější pohled na projekt.

- Uživatel (Ambassador User)
  Osoba, která by měla mít povědomí o operačních procedurách a systémech, kde má být systém nasazen. Také by měl stanovit, jaké informace budou na jaké obrazovce zobrazené.

- Vedoucí návrhář (Lead Designer)
  Vedoucí osoba po technologické stránce. Předpokládá se, že má zkušenosti s vývojem i návrhem a je schopna činit rozhodnutí.

- Návrhář-programátor (Designer-Programmer)
  Má na starost vývoj produktu.

- Další role
  Byznys expert (Business Expert) je role podobná Uživateli. Avšak nemá znalosti technického zázemí zákazníka, ale poskytuje informace ohledně byznysu, strategií a nastavených obecných pravidel. Koordinátor (Coordinator) dělá poznámky o plánu projektu a stavu relace. Tyto informace zpracovává a prezentuje. Také má zodpovědnost za zpřístupnění informací sponzorům. Některé týmy mají stanoveného Testera a jako občasnou roli Písaře (Writer)

2.3 Scrum

Scrum, z anglického "skrumáž", byl poprvé oficiálně představen a prezentován na konferenci OOPSLA (konference o objektově orientovaném programování, systémech, programovacích jazycích a aplikacíh) v roce 1995 dvojicí Ken Schwaber a Jeff Sutherland. Scrum není přímo proces pro vývoj softwaru, ale procesní rámec ve kterém lze používat procesy a techniky zcela jiné. V současné době je hodně oblíbený pro jeho jednoduchost a srozumitelnost, avšak jeho dokonalé zvládnutí je obtížné. Je vystavěn na teorii empirismu, který říká, že rozhodování má být založeno na tom, co je známe a znalost vyplývá ze zkušenosti. Pro řízení empirického procesu je třeba třech základních prvků.

- Transparentnost
  Viditelnost všech hledisek projektu všem, kteří mají vliv na jeho výsledek.

- Kontrola
  Častá kontrola postupu je ve Scrum velmi důležitá, avšak nesmí být tak často, aby překážela vlastní práci na projektu.

- Adaptace
  Reakce na naleznuté odchýlení od prováděného procesu, které je vyhodnocené jako nepřijatelné. Aby se minimalizoval dopad, na odchýlení budoucí je třeba reagovat co nejrychleji.

Scrum se skládá ze čtyř základních částí. Jsou to Scrum tým, činnosti, artefakty a pravidla. Dále jsou jednotlivé části podrobněji rozepsány [8].

2.3.1 Scrum tým

Scrum tým je především samostatný, tedy není řízen z venčí a má k dispozici všechny schopnosti, aby nemusel být ani na nikom zvenčí závislý. Jelikož pracuje v iteracích a inkrementálně, je schopen velice rychle sebereflexe, zpětné vazby a produkt poskytuje postupně. Je rozdělován na tři role a to na vlastníka produktu, vývojový tým a Scrum master.

- Vlastník produktu
  Jedna osoba, která je zodpovědná za práci vývojového týmu a správu produktového katalogu (viz Artefakty). Je třeba, aby byl plně respektován a jedině on provádí veškerá rozhodnutí ohledně toho, na čem bude vývojový tým pracovat.

- Vývojový tým
  Je složen z profesionálů, kde každý má v týmu pouze titul vývojář bez ohledu na jeho funkci. Dále se nedělí na podtýmy, zodpovědnost za výsledek má jako celek. Jeho účelem je tvořit přírůstek do produktu a dodávat ho vždy na konci každého Sprintu (viz Činnosti). Obvyklá velikost vývojového týmu je 3-9 osob. Tento počet je stanoven tak, aby tým byl dostatečně flexibilní, ale zároveň měl dostatečné kompetence.

- Scrum master
  Vedoucí týmu, který je zodpovědný zejména za dodržování pravidel Scrumu, týmu spíše pomáhá, aby byl efektivnější a zlepšoval své metody a postupu v rámci Scrum.

2.3.2 Definice "hotovo"

Aby bylo zaručeno, že každý člen týmu má znalost o kvalitě poskytovaného produktu, zavádí se definice co je "hotovo". Tato definice, pokud je součástí konvencí, standardů nebo směrnic organizace vývojového týmu, je použita jako základ pro ukončení úlohy, pokud není, je nutné, aby si tým tuto definici vhodně definoval. Definice "hotovo" se postupně vyvíjí společně s týmem, aby pokryla nároky na zvyšující se kvalitu. Každý dodávaný přírůstek musí tuto definici splňovat.

2.3.3 Činnosti

Scrum má své předepsané činnosti, které zajišťují pravidelnost a snižují nutnost schůzek, které nejsou v činnostech definované. Každá činnost ve Scrum má svou dobu tvání a umožňuje kontrolu a adaptaci.

- Sprint

  Je to základní kámen Scrum. Cílem je poskytnutí přírůstku splňující definici "hotovo" k výslednému produktu. Jeho doba trvání je pevně stanovená a není možné ji v průběhu měnit. Jakmile Sprint začne, musí vždy uplynout celá jeho doba. Toto neplatí u ostatních činností, které mohou skončit po splnění cíle. Obvykle Sprint trvá jeden měsíc, nebo menší časový úsek. V rámci projektu mají většinou Sprinty stejnou dobu trvání. Sprint slouží pro dosáhnutí postupných cílů, které se na začátku Sprintu naplánují, a během něhož se nesnižuje jejich kvalita. Sprint může být zrušen v průběhu pouze Vlastníkem produktu, pokud dojde k zastarání cíle. Sprint se skládá z plánovací schůzky (Sprint Planning Meeting), denních schůzek (Daily Scrum), vlastních vývojových prací, vyhodnocení sprintu (Sprint Review), a retrospektivy (Sprint Retrospective).

- Plánovací schůzka (Sprint Planning Meeting)

  Na této schůzce dochází k plánování cílů Sprintu, velikosti dodaného přírůstku, který splňuje definici "Hotovo" a sní spojenými pracemi. Scrum master ji řídí a cíle stanovuje celý Scrum tým.

- Denní schůzka (Daily Scrum)

  Schůzka konaná na začátku každého dne sloužící k vytvoření plánu na následující den a zhodnocení prací provedených od schůzky minulé. Obvykle trvá 15minut a účastní se jí vývojový tým.

- Vyhodnocení sprintu (Sprint Review)
  Průběh schůzky řídí Scrum master a účastní se jí celý Scrum tým. Má za úkol vyhodnotit právě dokončený sprint, případně změnit produktový backlog a rozhodnout o dalším postupu.

- Retrospektiva (Sprint Retrospective).
  Slouží k sebereflexi Scrum týmu a jeho zlepšení do dalšího Sprintu. Schůzka je zaměřena na používané metody, nástroje, mezilidské vztahy a procesy. Nyní schůzku Scrum master pouze neřídí, ale zúčastní se jí jako rovnocenný člen a snaží se ostatní členy povzbuzovat k zlepšení.

2.3.4 Artefakty

Artefakty jsou další příležitostí ke kontrole a adaptaci projektu. Slouží zejména k reprezentaci práce, její hodnoty a maximální transparentnosti.

- Produktový backlog
  Je dynamický seznam všech věcí ohledně produktu. Na začátku obsahuje pouze dobře pochopené požadavky a v průběhu vývoje, ale i provozu neustále roste. Následně může obsahovat všechny vlastnosti, funkce, požadavky, opravy chyb, rozšíření a tak dále. Každá položka má popis, prioritu a odhad. Každý produkt má jeden backlog, i když na něm pracuje více Scrum týmů a položky v něm jsou stále upřesňovány a přehodnocovány.

- Backlog sprintu
  Je množina úkolů z produktového katalogu, které si vývojový tým vytyčil, aby splnil cíl Sprintu. Spravuje ho pouze vývojový tým a je ukazatelem práce během Sprintu.

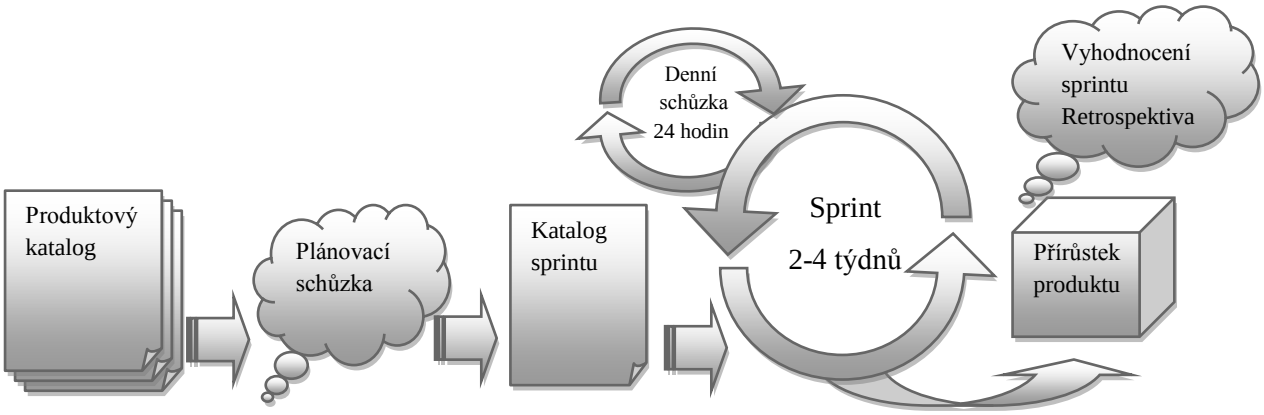
- Přírůstek
  Jsou všechny dokončené položky produktového backlogu v aktuálním sprintu. Přírůstek na konci každého Sprintu musí splňovat definici "hotovo", aby byl produkt potencionálně schopen nasazení.

2.3.5 Pravidla

Jsou určena vztahem a interakcí mezi výše popsanými rolemi, činnostmi a artefakty. Za jejich dodržování je, jak už bylo uvedeno, zodpovědný Scrum master.

2.3.6 Pracovní postup

Nyní známe všechny prvky, které Scrum obsahuje, a můžeme se podívat na vývojový cyklus pomocí Sprintů. Nejprve vlastník produktu vytvoří produktový katalog, který neustále doplňuje a spravuje. Následně proběhne Plánovací schůzka, kde je Sprint naplánován a jsou vybrány položky do katalogu sprintu. Poté probíhají práce na položkách katalogu sprintu a v rámci Sprintu se uskutečňují Denní schůzky. Na konci Sprintu vznikne přírůstek produktu, který je připravený na odevzdání zákazníkovi. Nakonec proběhne vyhodnocení sprintu a retrospektiva. Tento postup se opakuje pro každý Sprint. Celý cyklus je ilustrovaný na obrázku Obrázek 2.1.



Obrázek 2.1 - Pracovní postup SCRUM

2.4 Extrémní programování

Zakladatelem Extrémního programování (XP) je Kent Beck, který se i velmi přičinil o vznik Manifestu agilního vývoje softwaru. V roce 1999 vydal knihu s názvem Extreme Programming Explained [4], kde je poprvé Extrémní programování představeno. Byla tak vytvořena jedna z nejoblíbenějších, ale i nejkontroverznějších agilních metodik. Principy agilního vývoje tu jsou dovedeny opravdu až do extrému. Samotným tvůrcem je označována jako disciplína vedoucí k vývoji vysoce kvalitního softwaru rychle a plynule. Pro zachování jednoduchosti se vždy vytváří nejjednodušší verze produktu. Vyžaduje se velká zúčastněnost zákazníka na vývoji s rychlou zpětnou vazbou. Důraz je kladen na nepřetržité testování, plánování, zlepšování a integraci. Vše se děje ve velice krátkých pracovních cyklech, obvykle 1-3 týdny dlouhých. XP zavádí pojem "User Stories". Pomocí těchto "příběhů" zákazník definuje požadované chování systému [9].

2.4.1 Hodnoty

Extrémní programování je velmi zaměřené na lidský faktor a uvědomuje si, že právě ten může za většinu problému v rámci projektu. Proto je vystaven na čtyřech hodnotách pro udržení dlouhodobých sociálních cílů, před krátkodobými individuálními cíli jednotlivce.

- Komunikace

  XP staví komunikaci na první místo. Klade důraz na komunikaci mezi vývojáři, manažery a zákazníkem. Tvrdí, že špatná komunikace vyplývá z používání praktik, které komunikaci nepodporují. Proto využívá praktik, které dávají krátkodobý smysl jako jednotkové testování, párové programování atd. aby bylo zaručeno, že všechny strany musejí často komunikovat.

- Jednoduchost

  Při vývoji je dána přednost rychlé výrobě, co nejjednodušší podobu systému, který pokud je třeba, se v budoucnu upraví, před komplexním systémem, který není zcela využit. Jednoduchost spolu s komunikací vytváří dobré vztahy, protože o jednoduchých věcech se lépe mluví a jsou lépe pochopitelné.

- Zpětná vazba

  Zpětná vazba v XP funguje na několika úrovních pomocí neustálého testování (testování při vývoji, jednotkové testy, zákaznické testy, atd.). Čím víc se získá zpětné vazby, tím lépe. Zpětná vazba je úzce provázána s jednoduchostí a komunikací, jelikož s rostoucí jednoduchostí se zlepšuje možnost testování a většina informací se získá pomocí komunikace.

- Odvaha

  Poslední hodnotou je odvaha, která provází všechny předchozí hodnoty. Pokud nejsme schopni udržet naši nejvyšší rychlost, někdo jiný toho schopný bude a předběhne nás. Proto je dobré, abychom byli schopni nést riziko toho, že může nastat situace, kdy je nutné celou práci zahodit a začít znovu. Další důležitá funkce odvahy je v rámci komunikace, kde je třeba sdělovat i nepříjemné informace

  .

2.4.2 Praktiky

Při zohlednění výše zmíněných hodnot a základních aktivit, jako jsou psaní kódu, testování, naslouchání a návrh, XP využívá následujících 12 praktik.

- Plánovací hra (The Planning Game)

  Rychlé rozhodování o rozsahu příští verze produktu, který kombinuje obchodní priority a technické očekávání. Když aktuální stav překračuje plán, je třeba ho aktualizovat.

- Malé vydání (Small releases)

  Používání velmi rychlých cyklů k vydávání jednoduchých verzí systému.

- Metafora (Metaphor)

  Každý vývoj je dokumentován krátkým příběhem, jak má systém fungovat.

- Jednoduchý návrh (Simple design)

  Systém má být vždy navržen co nejjednodušeji. Jakákoliv komplexnost je odstraněna hned, jak je objevena.

- Testování (Testing)
  Programátor nepřetržitě píše jednotkové testy v průběhu vývoje. Zákazník píše demonstrační testy na hotové části funkčnosti systému

- Refaktorizace (Refactoring)
  Programátor neustále restrukturalizuje systém beze změny chování za účelem zjednodušení, odstranění duplicit, nebo přidání flexibility.

- Párové programování (Pair programming)
  Na jednom stroji pracují dva programátoři. Jeden kód píše, druhý ho hned kontroluje.

- Kolektivní vlastnictví (Collective ownership)
  Kód je majetek všech, tudíž každý může provést jakoukoliv změnu na kterémkoli místě systému kdykoliv chce.

- Nepřetržitá integrace (Continuous integration)
  Programátor má za úkol integrovat systém tolikrát denně, kolikrát dokončí úlohu.

- 40 hodin týdně (40 hour week)
  Nepracujte více jak 40 hodin týdně. Nikdy nepracujte přesčas dva týdny po sobě.

- Na straně zákazníka (On-site customer)
  Začlenění zákazníka do týmu. Vždy je vhodné věnovat plný čas na zodpovězení otázek zákazníka.

- Standardy pro psaní kódu (Coding standards)
  Programátoři dodržují dohodnuté stadardy pro psaní kódu, aby bylo udržené porozumění a komunikace.

2.5 Kanban

Původ této metodiky sahá do roku 1953, kdy ji vytvořil Taiiči Óno a zavedl ji v automobilce Toyota, jako metodu pro řízení výroby součástek. Agilní metodika Kanban využívá principů "Just-in-time", tedy řízení vývoje softwaru pomocí poptávky, který vychází z původního Kanbanu Kanban. Metodiku vývoje softwaru formuloval David J. Anderson jako inkrementální vývojový systém, který je proměnlivý pro jednotlivé organizace [10].

2.5.1 Praktiky

Kaban používá následujících šest praktik

- Vizualizace
  Pracovní tok je v podstatě neviditelný. Jeho vizualizace je jádrem pochopení pracovních postupů a aktuálního stavu práce. Tím je usnadněné provádění změn.

- Limitování rozdělané práce "work-in-progress"
  Limitování rozdělané práce se provádí na jednotlivé části, ale i na celý pracovní tok. Toto omezení pak představuje hlavní stimul k pokračující, inkrementální a evoluční změně systému.

- Spravuj pracovní tok
  Pracovní tok by měl být v průběhu kontrolován a spravován, aby bylo možné vyhodnotit, zda změny systému jsou pozitivní či negativní.

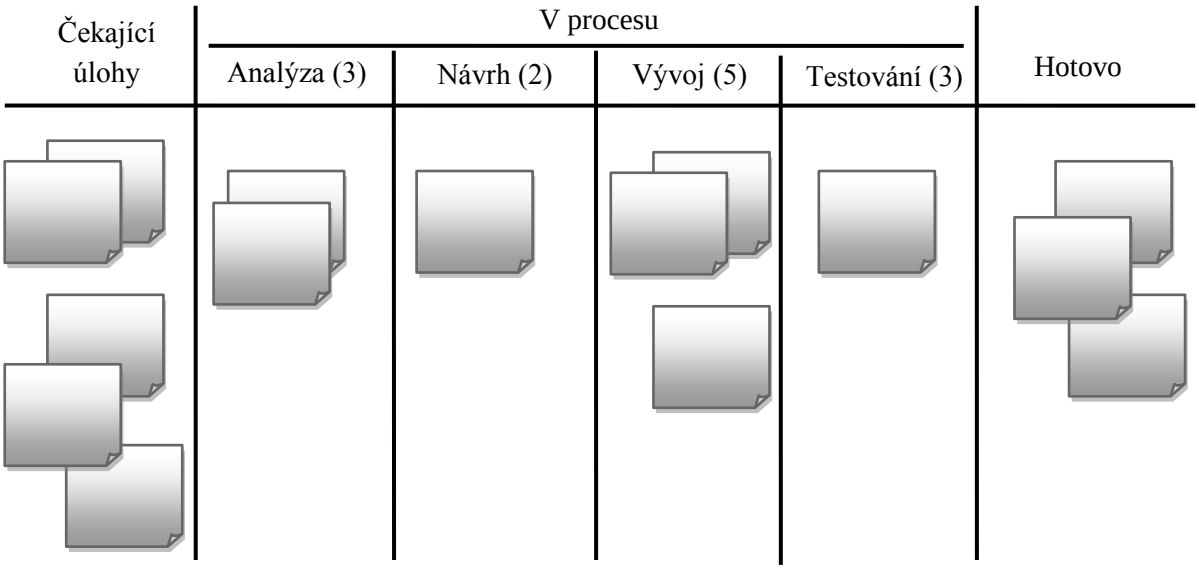
- Procesní politiku udělej explicitní
  Dokud procesy nejsou explicitní, je často těžké, nebo dokonce nemožné, diskutovat o jejich vylepšení. S explicitním pochopením procesů je možné upravovat je více racionálně a objektivně.

- Zaveď smyčky zpětné vazby
  Pro využití všech výhod Kanban systému je vhodné, aby organizace měla zavedenou druhou úroveň zpětné vazby. Tím se týmu dostane zpětné vazby z jiného úhlu pohledu a na ten je pak schopen lépe reagovat.

- Zlepšuj spolupráci a vyvíjej experimentálně
  Pokud tým dobře spolupracuje a sdílí informace, je schopen daleko lépe využívat různých modelových technik vývoje a minimalizuje riziko neúspěchu.

2.5.2 Kanban tabule

Velmi důležitým prvkem je Knaban tabule "Kanban Board". Slouží k vizualizaci pracovního toku, limitování rozdělané práce a kompletní správě pracovního toku. Jsou zde zobrazeny jednotlivé úlohy, reprezentované jako cedulky v aktuálních stavech rozpracování. Příklad tabule je uveden na obrázku Obrázek 2.2.



Obrázek 2.2 - Kanban tabule

3 Nástroje pro podporu agilních metodik

S rostoucí rozšířeností agilních metodik vzniká mnoho softwarových produktů určených pro usnadnění těchto metodik. Ve většině produktů je implementována podobná funkcionalita, která se liší podle metodiky, na kterou je produkt zaměřen. Nejvíce jsou implementovány nástroje pro podporu metodiky Scrum a Kanban. V této kapitole jsou přiblíženy vybrané softwarové nástroje, které sami sebe označují jako agilní softwarové nástroje, nebo nástroje pro agilní řízení.

3.1 VerisonOne

VerisonOne [11] je velice komplexní softwarový nástroj, který je na trhu už od roku 2002. Zaměřuje se převážně na malé týmy a řídí se heslem "Agile Made Easier", což v překladu znamená "agilní jednodušeji". Produkt obsahuje nástroje na podporu hned několika agilních metodik. Jsou to Scrum, Kanban, Lean, Extrémní programování a jejich kombinace. Jednou z výhod je, že poskytuje možnost integrace s jinými systémy. Pro virtuální týmy je zde výhodou možnost konverzací v reálném čase ať už ve fázi plánování, tak i v rámci celého projektu. Nedostatkem pak jsou chybějící nástroje pro podporu skupinové diskuse a brainstormingu, jako je například plánovací poker. Dále chybějící videokonference a mobilní verze. Nicméně VerisonOne je dobrým nástrojem pro správu projektů agilními metodikami a může být velkým přínosem i pro virtuální týmy.

3.2 OnTime Scrum

Dalším nástrojem pro správu agilních projektů je OnTime Scrum [12]. Implementována je metodika Scrum, ale k dispozici jsou i prvky jiných metodik, například Kanban tabule. Opět je nabízena možnost integrace s jinými systémy a navíc je produkt přizpůsoben i mobilním zařízením. Nevýhodou tohoto produktu je, že hodně funkcionality je řešeno pomocí přídavků, které uživatel musí doinstalovat.

3.3 ScrumDo

ScrumDo [13] má znovu v názvu Scrum z čehož je zřejmé na jakou metodiku se zaměřuje. Dále podporuje také metodiku Kanban. Je založen na jednoduchosti a umožňuje integraci do dalších systémů. Komunikace je řešena formou "chatu" využívá se pouze notifikací. Výhodou je, že jako jediný systém, který jsem nalezl, obsahuje funkci plánovacího pokeru.

3.4 Microsoft Project Profesional

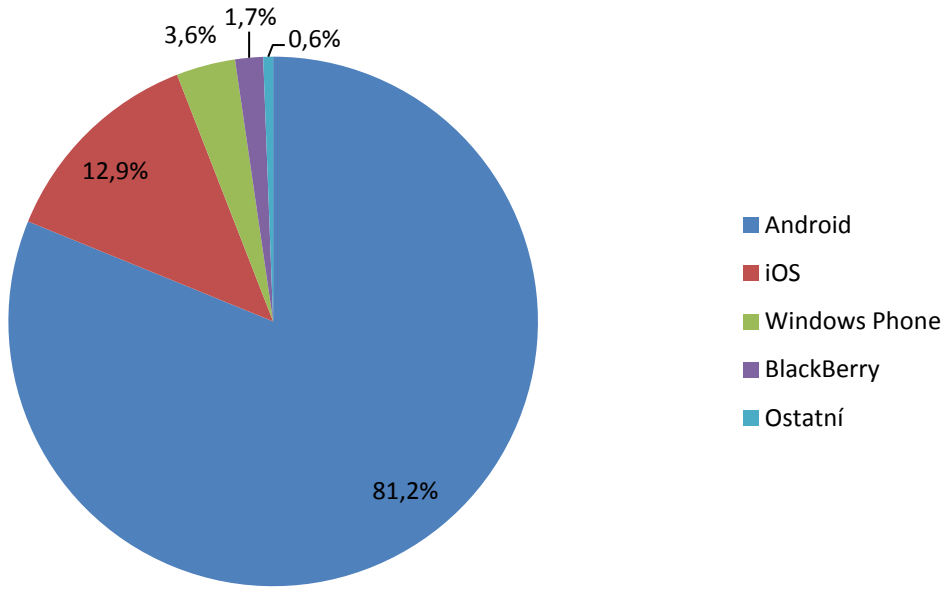
MS Project Profesional [14] je jedním z nejrozšířenějších nástrojů pro projektové řízení. Od ostatních produktů se liší zejména tím, že je velice rozsáhlý a je nutné ho před použitím instalovat na počítač. Všechny předchozí produkty byly dostupné z webového prohlížeče. I když není primárně určen pro řízení pomocí agilních metodik, je možné jej pro jejich podporu využít. Od nové verze 2013 byla přidána možnost interaktivní komunikace mezi uživateli pomocí Lync 2013. Proto je možné nyní jeho využití i virtuálními týmy.

4 Multiplatformní vývoj mobilních aplikací

Tato kapitola je zaměřena na vývoj aplikací pro mobilní zařízení. Nejprve se věnuje jednotlivým platformám operačních systémů. Zvláště třem nejrozšířenějším platformám Android, Windows Phone a iOS. Dále se v kapitole diskutuje o různých možnostech přístupu k vývoji aplikací na mobilní zařízení pro různé platformy. V poslední části jsou popsány nástroje pro multiplatformní vývoj mobilních aplikací, které jsou nejvíce používané. Jelikož je tato práce zaměřena na aplikace, nejsou zde rozebírány nástroje pro multiplatformní vývoj her.

4.1 Mobilní platformy

Každé mobilní zařízení ke svému provozu potřebuje operační systém. V průběhu vývoje mobilních zařízení se objevilo a zaniklo mnoho operačních systémů. V dnešní době mají největší tržní podíl tři platformy. Nejrozšířenější platformou pro mobilní telefony je Android od firmy Google, následuje iOS od firmy Apple a nakonec Windows Phone od Microsoftu. Ostatní platformy jsou na ústupu, například Sybian od firmy Nokia, nebo ještě v počátcích. Tržní podíl je znázorněn v grafu Graf 4.1.



Graf 4.1 - Podíl trhu ve třetím čtvrtletí 2013 [15]

4.1.1 Android

Android je open-source platforma založená na Linuxovém jádře. Její historie sahá do roku 2003, kdy ji začala vytvářet společnost Android, Inc. Následně tuto zatím malou firmu koupila společnost Google v roce 2005 a začalo se spekulovat, že chce vstoupit na trh s mobilními telefony [16]. V roce 2007 byla založena Open Handset Alliance, která sdružovala výrobce mobilních telefonů, poskytovatele mobilního spojení a další firmy se zájmem o mobilní technologie, a Android byl oficiálně prohlášen za open-source. O rok později bylo vydáno první SDK 1.0 (softwarový vývojový balík) a telefon G1 od společnosti HTC využívající právě Android. Od té doby počet mobilních telefonů s operačním systémem Android vzrostl a od roku 2010 má dominantní postavení na trhu.

Android, jak už bylo uvedeno, je open-source, to znamená, že vývojáři mají přístup ke zdrojovému kódu celé platformy a mohou ji jednoduše upravovat a vkládat na svůj hardware. Při vývoji byl kladen důraz na jednoduchou přenositelnost mezi zařízeními a zohledněny omezení související s mobilními zařízeními, jako je výdrž baterie. Výsledkem je systém, který je možný použít bez ohledu na rozlišení obrazovky a použitý chipset. Avšak tato variabilita se stává pro vývojáře určitou komplikací, protože musejí při návrhu aplikace počítat s velkým množstvím různorodých cílových zařízení. Android je nasazován mnoha výrobci, avšak Google vydává pro každou oficiální verzi "vzorový" přístroj s názvem Nexus od společnosti LG Electronics. Distribuce aplikací probíhá přes Google Play, který je nástupcem Android marketu. V roce 2013 nastal dramatický nárůst počtu aplikací, který nakonec překročil hranici jednoho miliónu aplikací [17]. Jelikož Android staví na veliké otevřenosti, uživatelé mohou aplikace získávat i z jiných zdrojů. Díky právě této otevřenosti kolem platformy vznikla široká komunita vývojářů i uživatelů a je možné vytvářet neoficiální upravené verze systému. Aktuální oficiální verze je 4.4.2 a nese název Kit Kat.

4.1.2 iOS

Operační systém pro mobilní zařízení společnosti Apple vychází z operačního systému pro osobní počítače Mac a nese od čtvrté verze název iOS, předchozí název byl iPhone OS, a vyšel pro mobilní telefony iPhone v roce 2007. Následně byl rozšířen do dalších zařízení společnosti Apple a to iPod Touch, iPad a Apple TV.

iOS je velice uzavřený systém a vývojářům, kteří chtějí vytvářet aplikace na tuto platformu, není umožněno zasahovat do systému. Distribuce aplikací probíhá pouze přes oficiální AppStore, který je řízen striktními pravidly. Apple je také jediný výrobce, který tento operační systém používá ve svých zařízeních. Malá variabilita dostupných zařízení a přísná pravidla vývoje přinášejí výhodu v podobě dobré přenositelnosti mezi zařízeními. Uživatel má téměř jistotu, že aplikace, kterou si stáhne, bude fungovat správně. Nejnovější verzí je iOS 7.0.4.

4.1.3 Windows Phone

Windows Phone, společnosti Microsoft, navazuje na operační systém Windows Mobile [18]. Vydán byl v roce 2010 ve verzi Windows Phone 7 a o dva roky později, tedy v roce 2012 vyšla nová verze s názvem Windows Phone 8 [23]. Každá verze je postavena na jiném jádře, proto systémy nejsou vzájemně kompatibilní. V roce 2011 oznámily společnosti Microsoft a Nokia společné plány na vytvoření nového mobilního ekosystému, skrze operační systém Windows Phone a mobilní zařízení od Nokie[19]. Úzká spolupráce nakonec vyústila v roce 2013 ve fúzi obou firem.

Windows Phone přichází s novým konceptem uživatelského rozhraní s názvem Modern UI. Úvodní obrazovka je složena z dlaždic, které si uživatel může poskládat podle sebe a které se dynamicky v čase aktualizují. Microsoft usiluje o sjednocení vzhledu všech jeho operačních systémů. Proto je toto uživatelské rozhraní v dnešní době nasazeno i v operačním systému pro stolní počítače Windows 8 a také v Xbox360. Mobilní zařízení s Windows Phone jsou vyráběny více výrobci, ale jsou stanoveny minimální požadavky, které přístroj musí splňovat. Aplikace jsou distribuovány pomocí Marketplace, na kterém je pro zveřejní aplikace nutné absolvovat kontrolu pracovníkem Microsoftu. Windows Phone tvoří kompromis mezi uzavřeností iOS a otevřenosti platformy Android.

4.1.4 Ostatní

Další platformou, která nemá takový podíl na trhu, je BlackBerry od firmy Research In Motion . Tato platforma se zaměřuje především na firemní sféru a vyniká zejména integrovaným systémem zpráv a upozorněním s vysokou mírou zabezpečení. Operační systém BlackBerry OS je založen na jazyku Java a je výhradně dodáván do zařízení stejného jména BlackBerry. Aktuální verze je 10.2.0.1791. Stále používaným, ale už nepodporovaným operačním systémem je Symbian OS. Symbian bylo sdružení založené firmami Motorola, Erricson, Psion a Nokia v roce 1998 pro podporu operačního systému EPOC, ze kterého vznikl Symbian OS [20]. Symbian se následně těšil veliké oblibě a také většinovému podílu na trhu chytrých telefonů. V roce 2008 Nokia skoupila téměř všechny akcie a výrobu svých zařízení zaměřila výhradně na něj. Avšak v konkurenci operačních systémů Android a iOS, Symbian neobstál a Nokia na tomto systému začala ztrácet a v roce 2011 oznámila ukončení vývoje [24].

Velké množství "menších" operačních systémů je založeno na Linuxu. Například relativně nový a zajímavý systém FireFox OS. Původně se jedná o projekt z roku 2011 s názvem Boot to Gecko, který byl v roce 2014 přejmenován na FireFox OS. Operační systém je vyvíjen společností Mozzila tak, aby aplikace vytvořeny technologiemi pro tvorbu webu, mohly přímo komunikovat s hardwarem. Na konferenci CES 2014 byla představeny nové zařízení využívající tento operační systém. Jsou to například televize od společnosti Panasonic nebo mobilní telefon od ZTE [21]. Dalším zástupcem této

skupiny operačních systémů je HP WebOS společnosti Hewlett-Packard. Operační systém byl představen v roce 2009 původně firmou Palm, která díky ekonomické situaci byla odkoupena firmou HP. Základem je jádro Linux a webové technologie. Systém využívá takzvaných karet pro přepínání mezi aplikacemi. Zajímavým operačním systémem od společnosti Samsung je Bada. Jako jádro je možné využít Linux, nebo jiný RTOS (operační systém reálného času). Bada byl představen v roce 2009 a o rok později vydán na trh. Nicméně v roce 2012 bylo ohlášeno ukončení vývoje a postupná integrace do systému OS Tizen, který je vyvíjen ve spolupráci firem Samsung a Intel.

4.2 Možnosti vývoje aplikací

Vývoj aplikací pro mobilní zařízení lze v zásadě rozdělit do dvou kategorií. První je nativní vývoj aplikací vždy pro jednu platformu. Druhým způsobem je takzvaný multiplatformní, neboli v angličtině "cross-platform", vývoj. Každý způsob nese pro vývojáře určité výhody i nevýhody.

4.2.1 Nativní vývoj aplikací

Základní způsob tvorby aplikací. Každá platforma využívá jiných technologií a programovacích jazyků. Díky této rozdílnosti nastává problém, jestliže chceme vytvořit jednu aplikace na více platforem. Taková aplikace se tvoří pro každou platformu zvlášť. Vznikají tedy různé varianty aplikace se stejným nebo podobným chováním pro každou platformu zvlášť. Proto je zapotřebí mít znalost o technologiích pro dané platformy nebo spolupracovat s vývojáři, kteří tyto znalosti mají. Vývojář má plnou kontrolu nad psaným kódem, který je uzpůsoben na danou platformu a má přímý přístup k hardwarovým možnostem přístroje. Avšak údržba těchto aplikací bývá komplikovaná a jakákoliv změna se provádí ve všech verzích. Aplikace jsou distribuovány skrze obchody s aplikacemi dané platformy, kde se dají lehce dohledat.

Android

Hlavním jazykem pro vývoj aplikací pro platformu Android je Java [16]. Primární vývojové prostředí je Eclipse, do kterého je možné doinstalovat nástroje pro podporu vývoje pro Android, tedy ADT (Android Development Tools) plugin. Je možné využívat i jiná vývojová prostředí. Společnost Google v roce 2013 vydala specializované prostředí s názvem Android Studio, které je poskytované zdarma. Základem tvorby aplikací je SDK (Software Development Kit), kde jsou obsaženy knihovny, dále pak emulátor pro testování a debugger pro ladění aplikací. SDK je dostupné pro operační systémy Linux, Windows a také Mac OS. Pro optimalizaci kódu napsaného v jazyce Java je využito virtuálního stroje Dalvik. Pro psaní aplikací pro platformu je možné využít také jazyka C/C++. Kód je však třeba přeložit do nativního kódu pomocí NDK (Native Development Kit).

Základní prvky aplikace:

- Activity
  Odpovídá jedné obrazovce a obstarává interakci s uživatelem. Jelikož aplikace většinou obsahují více activit, je v systému Activity Manager, který je spravuje.

- Service
  Proces běžící na pozadí. Je využíván pro úkoly, které trvají delší dobu nebo k přístupu ke vzdáleným zdrojům

- Content provider
  Prostředek pro oddělení dat od uživatelského rozhraní. Zajišťuje sdílení dat v rámci aplikace, ale i mezi aplikacemi.

- Broadcast reciever
  Tato část je zodpovědná za "odchytávání" upozornění a dle jejich typu a nastavení na ně reaguje. Využívá se například k upozornění aplikace na příchozí data nebo zprávu, o kapacitě baterie, atd.

Prvky, až na Content provider, mezi sebou komunikují pomocí zpráv (intents), které si mezi sebou posílají. V kořenovém adresáři aplikace musí obsahovat soubor AndroidManifest.xml, kde jsou všechny tyto prvky definovány a jsou zde uvedeny požadavky na využití služeb a hardwarových zdrojů

iOS

K programování pro platformu iOS je možno využít jazyka C nebo jeho nástavby, objektově orientovaného jazyka Objective-C [22]. Vývojové prostředí, XCode, je dostupné pouze v operačním systému MacOS a nativní vývoj na jiných systémech není možný. iOS je velmi uzavřený systém a i pro vývojáře je do značné míry limitován. K distribuci aplikací slouží výhradně oficiální App Store, který obsahuje schvalovací proces od firmy Apple. Omezenost systému a aplikací je důvodem vzniku modifikačního procesu systému, který je označován jako jailbreak, tedy prolomení vězení. Tento proces umožní nahrávání neautorizovaných aplikací do systému, ale nese s sebou riziko napadení zařízení a snížení jeho výkonu. Společnost Apple se snaží tomuto procesu soudně bránit. Zatím neúspěšně.

Windows Phone

Vývoj na platformu Windows Phone je v dnešní době rozdělován dle použitého SDK. Ve starší verzi 7.1 je možné využít technologii Silverlight nebo framework XNA, využívající technologii DiretctX pro tvorbu her. Verze SDK 8 už XNA nepodporuje. Aplikace napsané pro jednotlivé verze SDK nejsou navzájem kompatibilní. Základem pro obě možnosti je framevork .NET a vývoj probíhá v jazycích C# nebo C++. Pro oddělení dat aplikace a uživatelského rozhraní je hojně využíváno návrhového vzoru MVVM (model-view-viewmodel). Uživatelské rozhrání se tvoří pomocí značkovacího jazyku XAML. Díky těmto technologiím je možné kompletně oddělit tvorbu uživatelského rozhraní, které může vytvářet grafik neznalý jazyka C#. Pro vývoj v aktuálním SDK8 je potřeba mít nainstalované Visual Studio 2012 a Windows 8. Pokud chce vývojář využívat emulátor Windows Phone 8, je třeba mít Windows 8 Enterpise nebo Pro, aktivovanou hardwarovou virtualizaci Hyper-V a procesor podporující technologii SLAT.

Přehled platforem

Platforma	Jazyk	Vývojové Prostředí
Android	Java/C++/C	Eclipse,Android Studio, ..
iOS	Objective-C/C	XCode
Windows Phone	C#/C++/XAML	Visual Studio
BlackBerry OS	BlackBerry Widgets (HTML + Java)	Eclipse
Symbian	C++ Qt	Carbide C++
FireFox OS	HTML5,JavaScript,	Visual Studio

Tabulka 4.1 - Přehled vybraných platforem

4.2.2 Multiplatformní vývoj aplikací

Jak je vidět z tabulky Tabulka 4.1 ve vývoji aplikací pro mobilní zařízení vzniká velká fragmentace už na úrovni platforem a je obtížné obsáhnout všechny dostupné nástroje a technologie, aby mohl jeden programátor tvořit aplikaci běžící na více platformách zároveň. Stále častěji je poptávka po aplikacích, které jsou přenositelné i mezi fyzickými platformami, jako jsou stolní počítače, televize, tablety, mobilní telefony nebo herní kozole. Právě proto vznikla myšlenka multiplatformího vývoje, tedy vytvoření jednoho kódu a následné nasazení na více platforem najednou. Značnou výhodou je tento přístup, jak už bylo zmíněno výše, pro programátory jednotlivce, ale i pro malá vývojová studia. Ta mohou ušetřit na marginálních nákladech při vývoji aplikace pro větší množství potencionálních zákazníků. Nemusejí mít zaměstnané odborníky na jednotlivé platformy, ale mohou se zaměřit pouze na jeden nástroj. Nevýhodou může být nevhodné zvolení nástroje pro tvorbu. V dnešní době je

k dispozici mnoho nástrojů pro multiplatformní vývoj aplikací, ale ne všechny podporují alespoň základní tři platformy (Android, iOS, Windows Phone). Některé nástroje mají také problémy s integrací nativních funkcí přístroje, což ve výsledku může znamenat, že náklady na integraci aplikace na různé platformy překročí náklady na vývoj nativních aplikací pro tyto platformy.
Tvorbu multiplatformních aplikací lze rozdělit do základních tří fází.

- Tvorba aplikace

  Vlastní tvorba aplikace. K dispozici je mnoho programovacích jazyků. K vývoji je možné využít prostředků pro webové stránky jako HTML, JavaScript a CSS, dále objektově orientované jazyky Java a C# a v neposlední řadě jazyky C++ nebo LiveCode.

- Integrace na platformu

  Další částí je integrace nativních funkcí mobilního zařízení a testování aplikace na zvolené platformě. Každý nástroj řeší integraci trochu jinak. Některé mají vlastní jednotný systém volání systémových funkcí, u jiných musí programátor použít rozdílné funkce dle konkrétní platformy.

- Zveřejnění

  Finální fáze vývoje mobilní aplikace je zveřejnění. Liší se jednak dle způsobu vývoje aplikace, tak i dle použitého nástroje. Pokud je aplikace vytvářena pouze jako webová, tak šíření lze zajistit pouze sdílením odkazům na stránku, kde aplikace běží. Dále je možné vytvářet aplikace, které se tváří jako nativní aplikace a lze je publikovat v obchodech určených pro konkrétní platformu (AppStore,Google Play, atd.). Některé nástroje poskytují vlastní obchody pro sdílení aplikací jím vytvořených.

Spojením nástrojů dle použitých technologií vznikají tři základní přístupy k tvorbě multiplatformních aplikací.

- Webové aplikace

  Programátor má možnost vytvořit čistě webovou aplikaci, která na mobilním zařízení běží ve webovém prohlížeči. Prakticky se jedná o internetovou stránku, která je přizpůsobená pro běh a ovládání na mobilních zařízeních. K tvorbě se používají klasické webové technologie, jako jsou HTML a CSS. Pro zařazení prvků ovládání dotykového displeje anebo vložení grafických prvků, jsou využívané JavaScript aplikační rámce. Nástroje založené na JavaScriptu, pro tento typ vývoje, jsou například Sencha Touch nebo jQuery Mobile. Webové aplikace ke svému běhu potřebují webový prohlížeč a jejich distribuce probíhá výhradně pomocí sdílení internetových odkazů. Jedná se většinou o jednoduché aplikace, které nevyužívají nativních funkcí zařízení, jako je kamera, GPS, atd.

- Nativní aplikace

  Druhou možností je vytvoření nativní aplikace. Výhodou je možnost využít distribuční sítě platformy a dostupnost nativních funkcí zařízení. Multiplatformní způsob vývoje nativních aplikací se dělí do dvou skupin. První jsou nástroje, které jeden napsaný kód překládají do nativních jazyků různých platforem (Java,C++,Objective-C). Zástupcem je například Xamarin. Druhou skupinou jsou meziplatformní runtime nástroje, která vytvářejí komplexní meziplatformní vrstvu, která je nad nativním operačním systémem a tím stírá rozdíly jednotlivých platforem. Zástupcem jsou například Titanium, Corona nebo herní Unity.

- Nativně-webové aplikace

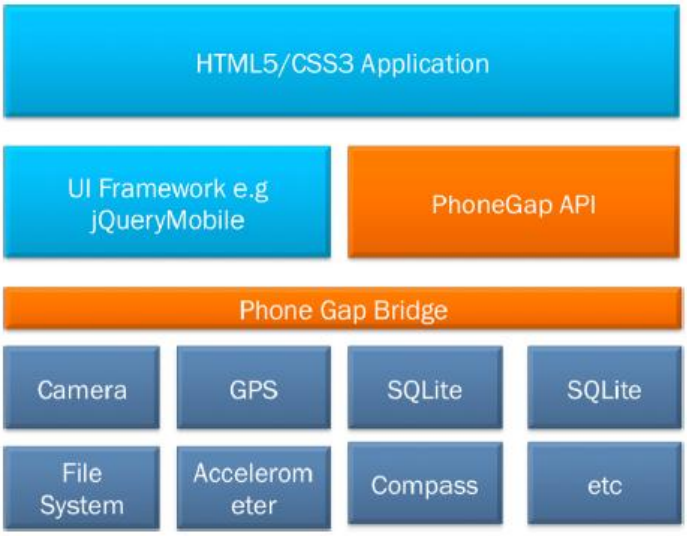
  Kombinací předchozích dvou způsobů vytváření aplikací, je nativně-webové aplikace. Programátor využívá webových technologií, jako u webových aplikací, a poté je vytvořená aplikace, kterou zapouzdřuje "wrapper", čímž z ní udělá aplikaci nativní. Wrapper je nástroj vytvářející instanci webového prohlížeče, ve kterém webová aplikace běží. Poskytuje přístup k nativním funkcím přístroje pomocí JavaScriptu. Výsledná aplikace je tedy nativní a je možné ji distribuovat skrze obchody poskytované jednotlivými platformami. Výhodou také je, že vývojářem může být tvůrce webových stránek, který je tak schopen do aplikací vložit nativní prvky. Nejoblíbenějším nástrojem tohoto typu je PhoneGap.

4.3 Nástroje

Rozdělení jednotlivých typů nástrojů bylo popsáno výše. Nyní se blíže podíváme na nejpoužívanější nástroje multiplatformního vývoje aplikací pro mobilní zařízení.

4.3.1 PhoneGap

Nástroj PhoneGap[25] je framework pro tvorbu aplikací na mobilní zařízení pomocí standardizovaných API (aplikační programovací rozhraní). Jádro bylo poprvé naprogramováno v roce 2008 na iPhoneDevCamp a následující rok vyhrál cenu People's Choice Award na O'Reilly Media's 2009 Web 2.0 Conference. Společnost Nitobi, která stála u zrodu tohoto nástroje, byla v roce 2011 odkoupena společností Adobe. Nyní je poskytován zdarma a licencován jako open-source projekt pod názvem Apache Cordova zařazený do neziskové organizace Apache Software Foundation.
PhoneGap patří do skupiny "wraperů", tedy aplikace jsou nativně-webové. Pro vývoj je využíváno primárně HTML5, které je stylizováno pomocí CSS3. Přístup k nativním funkcím přístroje je pomocí JavaScript knihoven. Struktura aplikace je znázorněna na obrázku. Obrázek 4.1.



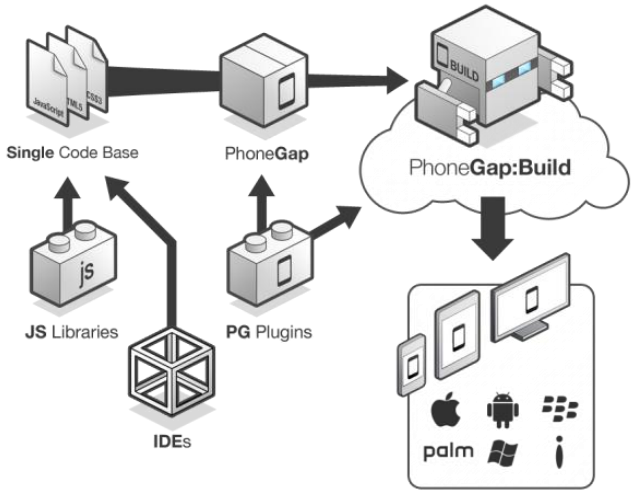
Obrázek 4.1 - Struktura aplikace PhoneGap [26]

PhoneGap je velmi komplexní nástroj a je dostupný pro mnoho platforem, jak je vidět v tabulce Tabulka 4.2 - Přehled podporovaných funkcí a operačních systémů. Nicméně je pouze framework a neposkytuje tedy žádné vlastní vývojové prostředí. Pro vývoj aplikací na různé platformy je nutné použít vývojové prostředí a SDK dané platformy. Nicméně v září roku 2012 PhoneGap spustil službu jménem PhoneGap Build, která vývojářům poskytuje možnost využít "cloud compiler", Obrázek 4.2, program vytvořený pouze v HTML, CSS a JavaScriptu zkompiluje na síti společnosti PhoneGap do všech poskytovaných platforem. Programátor tedy není nucen instalovat všechna SDK platforem, pro které chce vyvíjet. Distribuovat aplikace lze pomocí obchodu poskytovaných jednotlivými platformami a také interním skladištěm v rámci komunity PhoneGap.

	iPhone/ iPhone3G	iPhone3Gs and newer	Android	BlackBerry OS 60+	BlackBerry 10	WebOS	Windows Phone 7,8	Symbian	Bada
Accelerometr	✓	✓	✓	✓	✓	✓	✗	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compas	✗	✓	✓	✗	✓	✓	✓	✗	✓
Contacts	✓	✓	✓	✓	✓	✗	✓	✓	✓
File	✓	✓	✓	✓	✓	✗	✓	✗	✗
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	✗	✓	✗	✓	✗	✗
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification	✓	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✗	✗

✓ – podporovaná funkce ✗ –nepodporovaná funkce

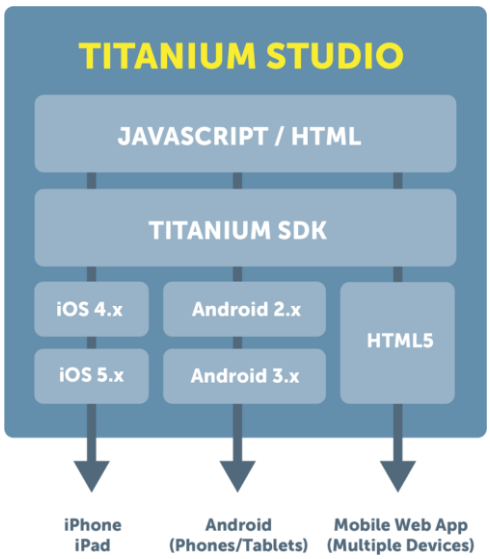
Tabulka 4.2 - Přehled podporovaných funkcí a operačních systémů [25]



Obrázek 4.2 - Princip fungování PhoneGap Build [25]

4.3.2 Titanium

Platforma pro vývoj mobilních aplikací od společnosti Appcelerator [27]. Tato platforma byla v roce 2008 představena pro multiplatformní vývoj aplikací mezi operačními systémy pro osobní počítače Linux a Mac. Následně byla přidána nativní podpora pro iOS a Android, později BlackBerry a Tizen. V roce 2011 bylo vydáno Titanium Studio. Vlastní Titanium SDK umožňuje tvorbu nativních, webových, ale i nativně-webových aplikací. Představuje most mezi kódem napsaného v JavaScriptu a operačním systémem. Poskytuje také možnost tvořit projekty pomocí modulů, které je možné sdílet mezi vývojáři skrz speciální obchod Open Mobile Marketplace. Tím je možné tvořit aplikace s minimálním psaním kódu. SDK je poskytované jako open-source pod licencí Apache 2. Poskytována je také placená licence pro podnikatele, která umožňuje integraci do systémů třetích stran.



Obrázek 4.3 - Struktura aplikací pomocí Titanium SDK [28]

4.3.3 Xamarin

V roce 2001 byl spuštěn open-source projekt pro mutiplatformní vývoj. Byl založen na .NET a programovacím jazyku CLI (Common Language Infrastructure), projekt byl nazván jako Mono. Byl určen jako nástroj na vývoj aplikací pro osobní počítače a herní konzole. Na základě Mono byl vytvořen MonoTouch pro mobilní platformu iOS a Mono for Android pro aplikace na platformu Android. Všechny tři systémy jsou založeny na programovacím jazyku C# (lze požít i CLI) ve spojení s .NET. Díky tomu je kód použitelný i pro Windows. Aplikace obsahuje jádro napsané v C# a .NET a je použitelné pro ostatní platformy. Uživatelské rozhraní je však třeba vytvořit pro každou platformu zvlášť, proto je vždy celé nativní aplikační rozhraní kopírované do aplikace a uživatelské rozhraní tak pro uživatele vypadá jako nativní aplikace. V roce 2011 byly licence pro Mono, MonoTouch a Mono for Android sdruženy pod název Xamarin [29]. O dva roky později byla vydána druhá verze Xamarin 2.0, která přinesla sdružení nástrojů pro vývoj na Android a iOS do Xamarin Studia. Také byla přidána možnost integrace do Viual Studia, kde je možné vyvíjet i aplikace pomocí Xamarin do Windows.

Obrázek 4.4 - Architektura Xamarin [30]

4.3.4 Sencha Touch

Sencha Touch [31] je zástupce nástrojů webových aplikací od společnosti Sencha. V roce 2010 byly touto společností zkombinovány oblíbené knihovny JavaScriptu a byla vydána první beta-verze Sencha Touch pro Andorid a iOS. Postupně byly přidávány další platformy a nyní jsou podporovány také BlackBerry, Windows Phone, Kindle a Tizen. Použit je jazyk JavaScript ve spojení HTML5 a CSS3. Sám o sobě tento nástroj nepodporuje přístup k nativním prvkům přístrojů, ale nabízí grafické uživatelské rozhraní a ovládání dotykových displejů. Pro tyto vlastnosti je používán mnohými jinými nástroji, jako jsou PhoneGap nebo Cordova, které poskytují právě nativní funkce. Sencha nyní poskytuje také vlastní Sencha SDK Tools, které je schopné webovou aplikaci zabalit do nativní podoby.

5 Návrh systému

V této kapitole bude popsán návrh systému pro podporu agilního vývoje softwaru virtuálními týmy, která bude zpracovávána v druhé části diplomové práce. Nejprve bude uvedena neformální specifikace obsahující závěry, které vyplývají z analýzy agilních metodik a nástrojů pro jejich podporu. Dále výčet a představení metod, které by bylo vhodné v systému implementovat. Poté jsou uvedeny použité technologie pro vývoj systému. Na závěr kapitoly je umístěn návrh grafického uživatelského rozhraní GUI.

5.1 Neformální specifikace

Ze zkoumání agilních metodik vývoje softwaru, Kapitola 2, vyplývá, že všechny agilní metodiky kladou důraz na komunikaci v týmu a předávání informací. Speciálně v metodice SCRUM je základním stavebním kamenem každodenní pořádání krátkých porad. V prostředí virtuálních týmů, kde se členové většinou za celé období spolupráce ani jednou nesetkají osobně, jsou tyto schůzky velkým problémem. Poskytované nástroje, Kapitola 3, sice poskytují možnosti zanechávání vzkazů nebo komunikace v reálném čase, ale pouze v textové podobě. Zároveň však ve většině případů neposkytují nástroje pro podporu brainstormingu. Proto bude systém zaměřen hlavně na přímou komunikaci. Cílem je, aby mohl systém vytvořit jakousi simulaci přímé interakce s podporou nástroji pro týmové plánování a brainstormingu. Dále jsou popsány nástroje vhodné pro implementaci.

Plánovací poker [2]

Jak už z názvu vyplývá, jedná se o rychlý a kvalitní nástroj pro podporu plánování. Je vhodný pro týmy, které mají 3 až 10 členů. Pokud je v týmu více členů, je vhodné je rozdělit do menších skupin. Plánovacího pokeru se účastní celý vývojový bez ohledu na funkce jeho členů a postupně ohodnocuje náročnost provádění úkolů. Každý z hodnotitelů má pro hodnocení karty v hodnotách 0, 1, 2, 3, 5, 8, 13, 20, 40 a 100. Vlastník produktu se také účastní, ale nehodnotí. Jeden z účastníků se zvolí jako moderátor, který však nemá zvláštní výsady, ale průběh plánovacího pokeru pouze řídí. Vlastní "hra" probíhá tak, že moderátor zvolí úkol pro hodnocení. Následně vlastník produktu poskytne komentář k úkolu a odpoví na všechny otázky. Další fází je hodnocení, kdy každý hodnotitel zvolí jednu kartu, ale neodhalí své hodnocení. Karty se obrátí všechny najednou a účastník s nejnižším a nejvyšším hodnocením poskytnou názory, proč se takto rozhodli. Následuje opět diskuse a nové hodnocení. Tento cyklus se opakuje do té doby, dokud se všichni neshodnou na jednom odhadu. Pro diskuse je vhodné stanovit časový limit, například 2 minuty, aby nedocházelo do zbytečného zabřednutí do problému. Plánovací poker je používán hlavně na začátku projektu, jako odhad pro velké množství úkolů, a při každém nově vzniklém úkolu. Moderátor také tvoří poznámky k jednotlivým úkolům.

SWOT analýza

Analýza silných a slabých stránek, jak je někdy nazývána, je metoda pro nalezení 4 faktorů, které ovlivňují projekt, projektový tým, produkt, atd. Svůj název dostala podle prvních písmen těchto faktorů:

Strengths - silné stránky

Weaknesses - slabé stránky

Opportunities - příležitosti

Threats - hrozby

První dva faktory se týkají vnitřních vlastností zkoumaného objektu a druhé dvě ho externě ovlivňují. Na základě rozpoznání a ohodnocení těchto faktorů lze stanovit strategie postupu.

Strategie S-O - Rozvoj silných stránek pro získání příležitosti

Strategie W-O - Využitím příležitosti překonat (odstranit) slabé stránky

Strategie S-T - Využitím silných stránek odvrátit hrozby

Strategie W-T - Omezit hrozby využívající slabé stránky

	Přínosné	**Škodlivé**
Vnitřní	S - silné stránky 1. 2. ..	W - slabé stránky 1. 2. ..
Vnější	O - příležitosti 1. 2. ..	T - hrozby 1. 2. ..

Tabulka 5.1 - SWOT tabulka

Je vhodné aby se analýzy účastnil celý tým i s vlastníkem produktu a nad silnými a slabými stránkami diskutovali společně. Analýza může probíhat dvěma způsoby. První je takový, že existuje pouze jedna tabulka slabých a silných stránek a účastníci ji společně doplňují pomocí moderátora. Druhým, který více podněcuje aktivitu účastníku, je takový, kdy každý účastník vyplní vlastní tabulku a poté se všechny tabulky spojí v jednu. Pro vyplňování tabulky je vhodné mít zvolen časový limit. U obou případů po sestavení tabulky následuje diskuze nad jejím obsahem a její úprava.

Agile Brainstorming

Tato metoda je určená k řešení problémů, které se vyskytnou v průběhu projektu. Nejprve se zvolí moderátor sezení, většinou vedoucí týmu, který ho řídí, ale také je aktivní účastník. Moderátor popíše problém a jasně stanoví cíl, ke kterému se má dojít. Každý z účastníků napíše návrh řešení problému. Jakmile má každý návrh připraven, tak ho přednese a jednotlivé návrhy se shromáždí. Následně probíhá diskuse a návrhy jsou ohodnoceny. Každý má v diskusi stejný hlas a váhu hodnocení. Tímto způsobem se kolektivně rozhodne řešení daného problému.

Skórovací metoda s mapou rizik [32]

Metoda sloužící k identifikaci a ohodnocení rizik. Je složena ze tří fází: identifikace rizika, ohodnocení rizika a návrhy na opatření ke snížení rizika. Pro identifikaci rizik [3] se tým zaměřuje na ohodnocení rizik ze čtyř nejdůležitějších oblastí: technická, finanční, personální a obchodní oblast. Rizika může tým, podobně jako provádět SWOT analýzu, provádět dohromady nebo každý zvlášť a výsledky spojit. Každé riziko se hodnotí dle možnosti výskytu a také dle jeho dopadu. Pro ohodnocení se používá metody, kde každý člen týmu ohodnotí nezávisle na ostatních riziko v rozmezí od 0 do 10. Jednotlivé známky se zprůměrují a tím se stanoví skóre pro výskyt a dopad. Celkové ocenění rizika je pak součinem těchto dvou skóre. Výsledky se zapíší do grafu a pro významná rizika tým zpracuje návrhy protiopatření. Doporučuje se zpracování rizik v kvadrantu kritických a významných rizik.

Kvantifikace rizik členy analytického týmu	1.	2.	3.	4.	5.	6.	7.	8.	Skóre (průměrné hodnoty)	
Možnost výskytu (1 min. až 10 max.)										X
Dopad (1 min. až 10 max.)										X
ocenění rizika = skóre pravděpodobnosti × skóre dopadu										

Tabulka 5.2 - Příklad tabulky pro zápis hodnocení rizika [32]

Graf 5.1 - Mapa rizik [32]

Risk Burn-down Chart [33]

Tento nástroj slouží primárně k vizualizaci rizik v průběhu projektu. Nejdříve jsou opět identifikována rizika a následně ohodnocena. V této metodě se hodnotí procentní pravděpodobnost výskytu a počet dnů, které je nutné vyhradit pro nápravu rizika, jestliže nastane. Pro stanovení hodnocení lze využít stejné metody jako u ohodnocování ve skórovací metodě, jen s rozdílem vstupných proměnných. Míra rizika se zde stanoví jako součin průměru pravděpodobností výskytu s průměrem počtu dnů ztráty. Na závěr se míry rizika sečtou a vznikne celková míra rizika pro danou iteraci. Tento postup se provádí v každé iteraci a hodnoty celkové míry rizik se vynášejí do grafu. V grafu by měl být zřetelný lineární pokles v celkové míře rizika. Pokud tomu tak není, tým by měl věnovat čas na přímé snížení rizik.

Videokonference

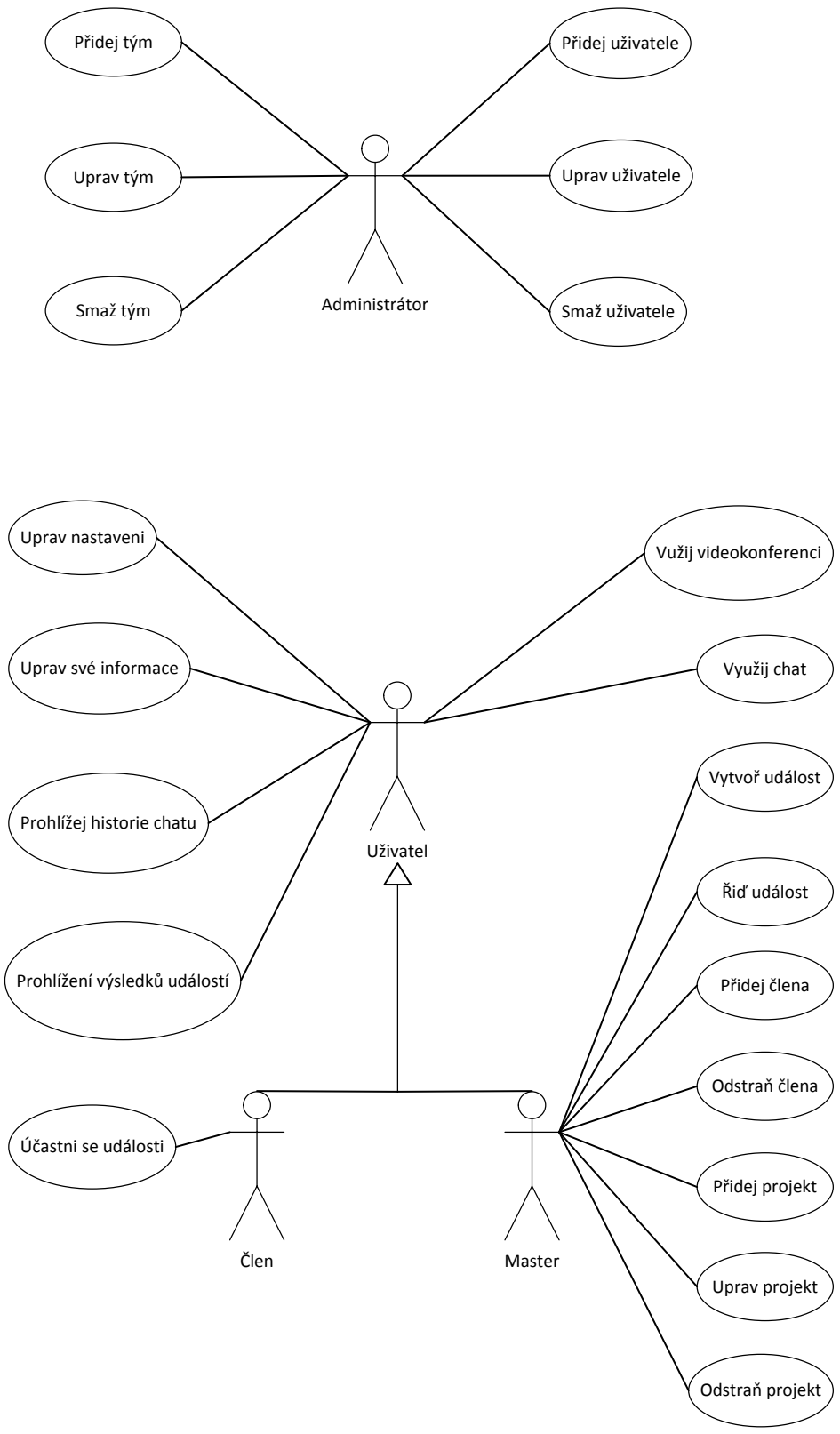
Hlavním komunikačním prvkem bude v systému videokonference. Ta bude zajišťovat interakci mezi uživateli a podporovat integraci všech členů týmu. Pokud nebude díky technickým problémům možné navázat přímou komunikaci skrze videokonferenci, bude možné komunikovat v textovém režimu.

5.2 Diagram použití

Na základě neformální specifikace a rozboru očekávaných funkcí byly identifikovány tři role uživatelů systému. Všichni aktéři se budou do systému přihlašovat pomocí vlastních přihlašovacích údajů, které budou odeslány na server a zkontrolovány. Dle role má uživatel rozdílná práva a možnosti pro používání systému. Jednotlivé role a jejich možnosti jsou popsány dále.

- Administrátor - je hlavním správcem systému. Má za úkol zadávat do systému nové týmy a jejich vedoucí, kteří jsou označovány rolí "Master". Požadavky na vložení nového týmu do systému přicházejí administrátorovi elektronickou poštou společně s objednávkou na využívání služeb systému. Jelikož videokonference je řešena pomocí externí služby, administrátor tým dle objednávky zavede do systému. Po té odešle vedoucímu týmu jeho přihlašovací údaje.
- Master - je vedoucím týmu. Jeho hlavním cílem je spravovat a řídit události. Při každé nově vytvořené události by se měl odeslat informační e-mail všem členům týmu. Dále má možnost spravovat členy týmu a jednotlivé projekty. Stejně pak jako člen má možnost využívat videokonferenci a chat. Dále pak může spravovat své vlastní údaje a měnit nastavení aplikace. V neposlední řadě bude moci prohlížet historii textové komunikace, výsledky již vykonaných událostí a termíny naplánovaných událostí.
- Člen - má nejmenší oprávnění ve využívání systému. Hlavní činností je účast na naplánovaných událostech. Může také využívat společných možností jako Master, což je správa vlastních dat, nastavení systému, využití videokonference, chat, prohlížení historie, výsledků vykonaných událostí a termínů naplánovaných událostí.

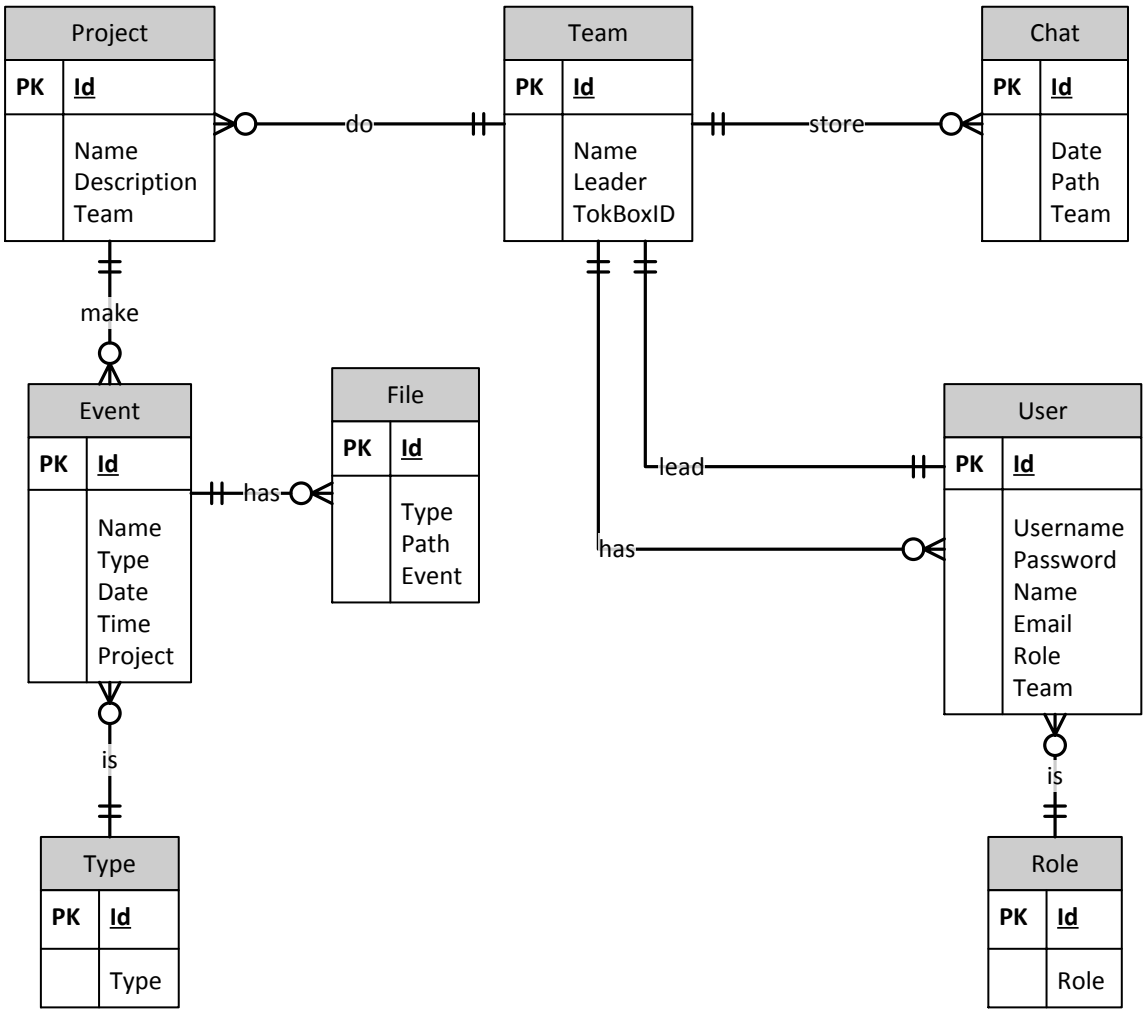
Pro vytvoření přehledného diagramu použití byl vytvořen abstraktní uživatel, který slučuje případy použití uživatelů Master a Člen. Výsledný diagram použití demonstruje Obrázek 5.1.



Obrázek 5.1 - Diagram použití

5.3 Návrh databáze

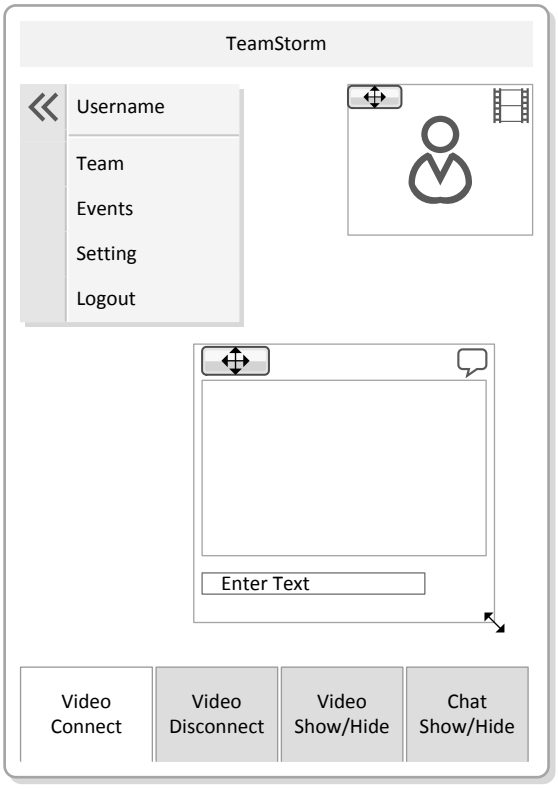
Informační systémy jsou založeny na práci s informacemi a ty se musejí někde ukládat. Pro potřeby aplikace této diplomové práce byla navrhnuta databázová struktura, kterou popisuje Obrázek 5.2 - Návrh databáze Hlavní tabulkou je *Team,* na který ostatní navazují. Každý tým musí mít minimálně jednoho uživatele s rolí *Master* a vlastní klíč pro přístup k službě TokBox, přes kterou je řešena videokonference. Pro vedení uživatelů v systému slouží tabulka *User*. Tabulka *Chat* slouží k udržení historie textové komunikace. Obsahuje datum a cestu k souboru, kde je komunikace uložena. Další tabulkou, která navazuje na *Team,* je *Poject*. Ta obsahuje název projektu, jeho popis a jsou na ni odkázány jednotlivé události aplikace, uložené v tabulce *Event*. Typ události může nabývat hodnot, z tabulky *Type***.** Události mají na sebe navázané soubory s výsledky.



Obrázek 5.2 - Návrh databáze

5.4 Uživatelské rozhraní

Vzhled uživatelského rozhraní u informačních systémů je důležitou a nedílnou součástí. Ač je systém určen pro projektové týmy vyvíjející softwarové nástroje, ne všichni budou informatici, kteří povětšinou vzhled upozaďují, je nutné, aby byl vzhled líbivý a příjemný. Proto je vhodné vytvořit alespoň základní uživatelskou customizaci vzhledu. Pro tento účel je vytvořeno dvoje barevné provedení aplikace a to světlé a tmavé doplněné o barevné dekorační prvky. Jelikož má být aplikace multiplatformní a má být spustitelná na více typů zařízení, je nutné podle toho volit i rozložení komponent v uživatelském rozhraní. Proto je navigační panel aplikace skrytý na pravé straně tak aby při práci na menším zařízení nepřekážel, a je možné ho vyvolat tlačítkem, nebo gestem. Videokonference i chat jsou taktéž skryté a pomocí tlačítkové lišty na spodní straně aplikace je možnost jejich vyvolání. Po vyvolání je možné obě okna umístit pomocí "Drag-and-Drop" na uživatelem definované místo. Oknu Chat lze také libovolně měnit velikost. Návrh základního rozložení ilustruje Obrázek 5.3 - Návrh uživatelského rozhraní.



Obrázek 5.3 - Návrh uživatelského rozhraní

5.5 Vstupní a výstupní data

Jak bylo uvedeno v kapitole 3, programů pro podporu agilních metodik je více a proto je vhodné implementovat podporu importu a exportu dat, aby bylo možné data sdílet mezi jednotlivými programy. Pro import souborů jsem vycházel ze struktury uložení dat formátu XML programem MS Project [14]. Úlohy se v MS Project exportují v tagu <*Tasks*> následují jednotlivé úlohy v tagách <*Task*>. Pro import dat do aplikace je důležitá pouze položka <*Name*>, jelikož je umožněn import pouze seznamu úloh jako vstupů do jednotlivých implementovaných nástrojů, a položka <*WBS*>, která určuje stupeň rozložení cílů. Systém pracuje až do 2. úrovně zanoření. Import je umožněn pro všechny typy nástrojů kromě SWOT analýzy, jelikož ta pojednává o celém projektu.

Výstupem aplikace jsou výsledky výše popsaných metod. Tyto výsledky jsou k dispozici v datovém formátu XML a z každé metody je vytvořen report ve formě PDF dokumentu. Při exportu dat do souboru formátu XML jsem taktéž vycházel z notace souborů MS Project, která byla dle potřeb upravena. V Analýze rizik, byl pro stanovení ceny rizika použit tag <*Cost*>, pro odhad trvání byl v nástroji Scrum Pocker použit tag <*Estimated*>. Pro nástroj SWOT analýza a Agile brainstorming byly zavedeny tagy nové a to <*Strenghts*>, <*Weaknesses*>, <*Opportunities*>, <*Threats*> pro SWOT analýzu a tag <*Solution*> pro Agile brainstorming.

Report v dokumentu ve formátu PDF obsahuje vždy na začátku identifikační údaje o projektu a události, následuje tabulka výsledků a u nástroje Scrum pocker a Analýzy rizik bude součástí výstupu vygenerovaný graf výsledků.

```
<?xml version="1.0" encoding="UTF-8"?>
<Project>
  <Name>Proj 1</Name>
  <Author>Pavel Veverka</Author>
  <CreationDate>2014-05-04</CreationDate>
  <Team>Prvni tym</Team>
  <Event>Scrum Pocker</Event>
  <Tasks>
      <Task>
        <Name>Phase #1</Name>
        <WBS>1</WBS>
        <Estimated>7</Estimated>
      </Task>
      <Task>
        <Name>Ukol 1</Name>
        <WBS>1.1</WBS>
        <Estimated>2</Estimated>
      </Task>
      <Task>
        <Name>Ukol 2</Name>
        <WBS>1.2</WBS>
        <Estimated>5</Estimated>
      </Task>
  </Tasks>
</Project>
```

Obrázek 5.4 - Ukázka exportovaného XML souboru

6 Implementace

6.1 Použité technologie

Virtuální týmy mohou být složené ze zaměstnanců jedné společnosti, nebo mohou být z několika různých společností. Tím je pravděpodobnost používání nejednotných systémů značně vysoká. Ať už se jedná o operační systémy, komunikační technologie nebo i používané zařízení, každý člen týmu může pracovat na zcela jiném zařízení. Proto výsledný systém této práce má být multiplatformní. Dále je žádoucí, aby byl schopen importovat data ze systémů třetích stran a umožňoval výsledky exportovat ve vhodných formátech. V této části jsou popsány technologie, které jsou rozděleny do několika kategorií.

Platformy

Jak už bylo zmíněno výše, cílem je postihnout co nejvíce variant použití. V prvé řadě je důležité, aby systém mohl být používán na osobních počítačích. Z nabízených operačních systémů je pro jeho rozšířenost hlavní důraz kladen na MS Windows. Dalším cílovým zařízením jsou tablety a chytré mobilní telefony. Tyto dvě hardwarové platformy nejvíce pokrývá Android.

Vývojový nástroj

Pro vývoj byl zvolen nástroj PhoneGap. Tento nástroj je zvolen pro jeho nejširší rozsah použití na různých platformách ze všech zkoumaných nástrojů. Díky této vlastnosti je možné vyvíjený systém v budoucnu dále rozšiřovat za rámec zvolených platforem. Další výhodou tohoto nástroje je možnost tvorby aplikací na mobilní zařízení v nativním vzhledu. Na osobním počítači bude systém spustitelný ve webovém prohlížeči a na mobilním zařízení bude webový kód "zabalen" pomocí PhoneGap.

Programovací jazyk

PhoneGap je framework, který zpracovává aplikace vyvíjené pomocí webových technologií. Proto pro sestavení aplikace bylo použito HTML5, jako jazyk vytvářející webové stránky, a CSS3, které je použito na stylování jednotlivých elementů stránky. Chování aplikace obstarává knihovna jQuery a její nástavba upravená pro mobilní zařízení jQuery mobile. Obě knihovny jsou založené na jazyku JavaScript. Pro uchování dat bylo použito databáze vytvořené v jazyku MySql, která běží na webovém serveru společně se serverovou částí systému vytvořenou v jazyku PHP.

Další použité technologie a knihovny

Videokonference bude možná skrze službu TokBox [34]. Tato služba je dostupná na takzvaném "cloud" tedy na externím serveru. TokBox byl vybrán, protože poskytuje aplikační rozhraní pro PhoneGap, který zprostředkovává hardwarový přístup k videokameře a mikrofonu mobilního zařízení. Služba TokBox je zpoplatněna, ale pro školní účely vytvoření této práce, po vzájemné domluvě, mi byl poskytnut aplikační klíč s prodlouženou zkušební dobou.

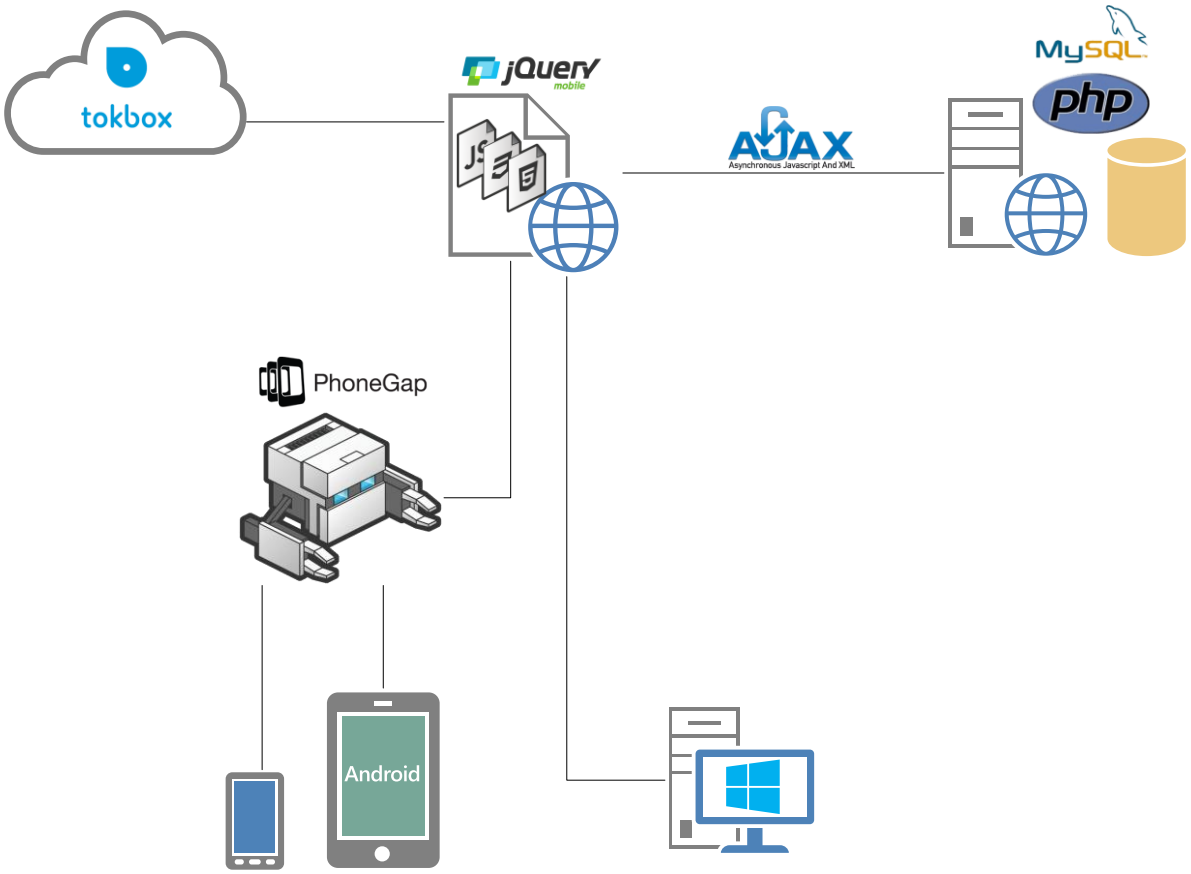
Pro komunikaci a předávání dat mezi webovou stránkou a PHP serverem jsem zvolil technologii AJAX (*Asynchronous JavaScript and XML*). Jak už název napovídá, jedná se o asynchronní přenos dat ve formátu XML pomocí JavaScript knihoven.

Aplikace je primárně určena pro mobilní operační systém Android a proto pro úpravu vzhledu je využito nativeDroid a jQuery Mobile Theme, který byl upraven pro specifické potřeby aplikace.

K vytvoření reportů ve formě PDF dokumentu byla použita PHP knihovna *fpdf* [35]. Součástí reportů jsou generované grafy vytvořené opět PHP knihovnou a to *phplot* [36].

6.2 Struktura systému

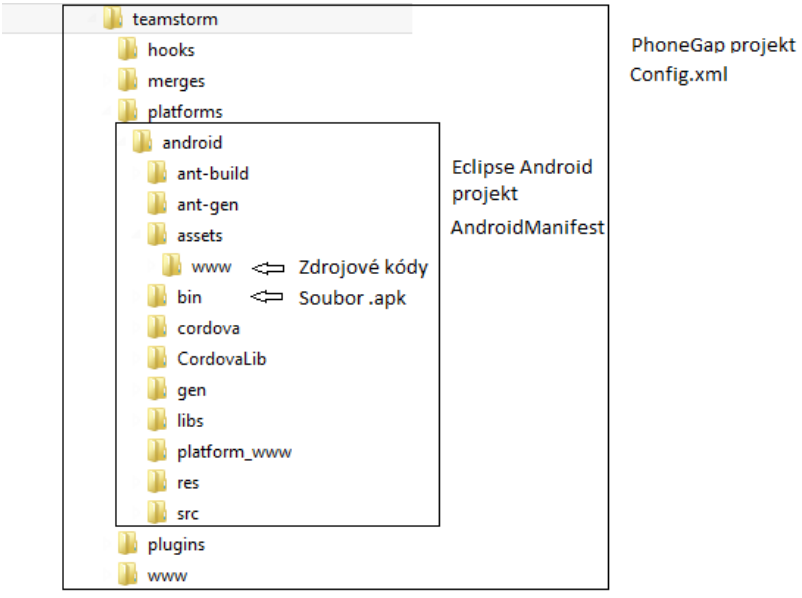
Strukturu systému s použitými technologiemi ilustruje Obrázek 6.1. Je zde zřejmé rozložení celého systému, kde je jádrem webová stránka vytvořená pomocí jQuery mobile, HTML5 a CSS3, která při využívání videokonference komunikuje se službou TokBox. Na druhé straně komunikuje pomocí technologie AJAX s webovým serverem s přístupem do databáze. Jak je vidět celý systém vychází z jednoho kódu, který je zpracován pomocí PhoneGap pro mobilní zařízení a ke kterému desktopové stanice přistupují skrze webový prohlížeč.



Obrázek 6.1 - Struktura systému s použitými technologiemi

6.3 Nastavení vývojového prostředí

Proto, aby výsledná aplikace mohla být multiplatformní, bylo využito nástroje PhoneGap. Jako hlavní mobilní platformou, na kterou je aplikace zaměřena, je systém Android. Jako vývojové prostředí lze využít jakýkoliv textový editor. Já jsem si vybral MS Visual Studio 2012, protože jsem na práci v tomto prostředí zvyklý. Ladit aplikaci lze pomocí jakéhokoliv nástroje poskytovaného pro vývoj webových aplikací, například Developers Tool ve webovém prohlížeči Google Chrome. V prvé řadě stáhneme a nainstalujeme samotný nástroj PhoneGap. Jelikož chceme vytvářet aplikace pro Android, mobilní zařízení je nutné nejprve nainstalovat pomocí Android SDK (software development kit). Cílová verze Android SDK byla zvolena nejnovější 4.4 KitKat ovšem minimální verze musí být 4.0 Ice Cream Sandwich. Následuje vytvoření aplikace, to lze provést pomocí příkazové řádky spuštěním příkazu *phonegap create jméno_aplikace*. Nyní je možné vytvořit aplikaci pomocí PhoneGap. Testovat aplikaci lze na fyzickém zařízení nebo na emulátoru poskytovaném Android SDK. Pro spuštění aplikace lze využít příkazové řádky a příkazů nástroje PhoneGap nebo lze využít prostředí Eclipse, který je hojně využíván k vývoji aplikací pro Android. Po nastavení prostředí Eclipse, pro práci s PhoneGap [37], je testování a práce s emulátorem pohodlnější a proto jsem se rozhodl využít pro testování prostředí Eclipse. Spuštění a překlad aplikace poté probíhá stejně, jako kdyby se jednalo o nativní aplikaci pro Android. Díky tomu je při každém překladu vytvořen instalační soubor formátu APK, který je možné nahrát do fyzického zařízení, nebo publikovat v Google Play marketu. Těmito kroky je vytvořena stromová struktura složek projektu, kterou ilustruje Obrázek 6.2. Mimo pozice uložení zdrojových kódů aplikace a instalačního souboru APK je také důležité umístění souboru *Config.xml*, který nastavuje chování PhoneGap a souboru *AndroidManifest.xml*, který upravuje nastavení nativní Android aplikace.



Obrázek 6.2 - Stromová struktura složek projektu

6.4 Serverová část

Jak už bylo zmíněno, serverová část je napsaná v jazyce PHP. Tento jazyk funguje na bázi dotaz odpověď, kdy server čeká na dotazy ze strany klienta a pak mu na ně zasílá odpovědi. Tento způsob není pro systémy komunikující v reálném čase nejvhodnější, a proto se nabízela možnost využít jiných programovacích jazyků pro tvorbu aktivního serveru například NodeJS, což je knihovna jazyka JavaScript. Servery napsané v jazyce python nebo JavaScript obvykle potřebují vlastní fyzický hardwarový server, nebo pronajatý virtuální server. Při průzkumu a výběru vhodné technologie jsem narazil na velké problémy s nastavením a provozem serveru takovéhoto druhu. Právě proto jsem nakonec zvolil jazyk PHP. Spustit serverovou část lze na kterémkoli webovém serveru, který podporuje PHP5 a MySQL, což je v dnešní době skoro každý. Proto je nasazení serveru napsaném v jazyku PHP daleko rozsáhlejší, než by tomu bylo u aktivního serveru napsaného v jiném jazyku.

Serverová část systému byla rozdělena do dvou hlavních souborů. Jelikož PHP pracuje na principu dotaz odpověď, byl pro řízení stavů a komunikace v systému zřízen komunikační mechanismus na bázi dotazů a příkazů, který obstarává soubor *process.php*. Zde jsou zpracovávány zprávy nesoucí textovou komunikaci, či řídící povely. Pomocí technologie AJAX serveru přicházejí zprávy od klientů a na základě položky *function* se rozhoduje, jaký se má vykonat příkaz. Je zde obsluhován chat, zaznamenáváni a vyhledáváni online uživatelé a vytvářejí se zde soubory s výsledky událostí ve formátu XML. Tyto soubory jsou identifikovány v jejich názvu podle klíče *typ_události-číslo_události-datum.xml*. Jako odpověď je vrácen příznak úspěchu či neúspěchu a nové položky v textové komunikaci.

Druhým souborem je *server.php*, který obstarává zbytek funkcí serveru. Hlavním účelem je komunikace s databází. Na začátku souboru jsou definované přístupové informace do databáze MySQL. Dále je struktura podobná jako v předchozím souboru. Na server jsou zasílané zprávy pomocí technologie AJAX a dle položky *function* se rozhoduje o provedeném příkazu. Jsou zde obstarávány příkazy pro přihlášení, vyčtení informací o uživateli z databáze, úprava, přidávání a mazání dat v databázi. Další důležitou funkcí je vytváření zpráv ve formě PDF dokumentu. Zde jsou využity knihovny PHPlot a fpdf pro vytvoření grafů a zpráv s výsledky.

6.5 Klientská část

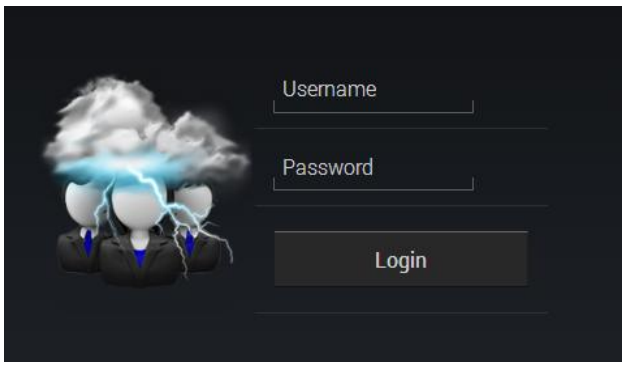
Použitím PhoneGap aplikace vychází z jednoho kódu a proto není nutné pro jednotlivá zařízení vytvářet rozdílnou klientskou část systému. Pouze u funkcí, které vyžadují nativní přístup k zařízení, například videokonference požaduje přístup ke kameře a mikrofonu, bylo nutné rozpoznat, zda se jedná o mobilní zařízení a poté použít příslušnou funkci.

Jádrem aplikace je webová stránka vytvořena pomocí jazyka HTML5. Byla použita struktura "Multi-page template", kterou využívá jQuery Mobile. Jednotlivé stránky aplikace jsou uloženy v jednom souboru *index.html*. Každá stránka je identifikována v rámci dokumentu definovanou strukturou, kterou zobrazuje Kód 6.1.

```
<div data-role="page" data-theme='b' id="loginPage">

        <div data-role="header" data-position="fixed" data-tap-toggle="false" data-theme='b'>
        </div> <!-- hlavička stránky -->

        <div data-role="content">
        </div> <!-- obsah stránky -->

        <div data-role="footer" data-position="fixed" data-tap-toggle="false" data-theme='b'>
        </div> <!-- patička stránky -->
</div>
```

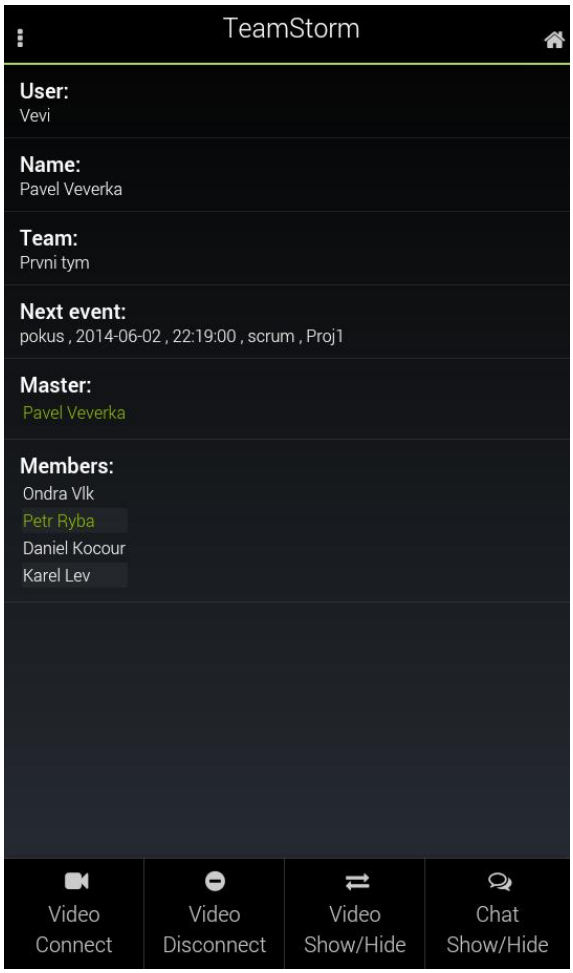
Kód 6.1 - Struktura stánek v dokumentu

Celá aplikace je tímto způsobem rozdělena do několika stránek, mezi kterými je navigováno. Hlavička je pro všechny stránky stejná a obsahuje pouze název aplikace "TeamStorm", tlačítko "domů" odkazující na hlavní stránku a tlačítko pro otevření navigačního menu. Patička je také stejná pro všechny stránky a jsou v ní umístěna tlačítka pro ovládání komunikace. Jediná stránka, která má odlišné tyto dvě části je první stránka sloužící pro přihlášení uživatele. Obsahuje logo aplikace, které bylo vytvořeno z podkladů podléhajícím licenci Creative Commons [38], a formulář pro přihlášení uživatele.



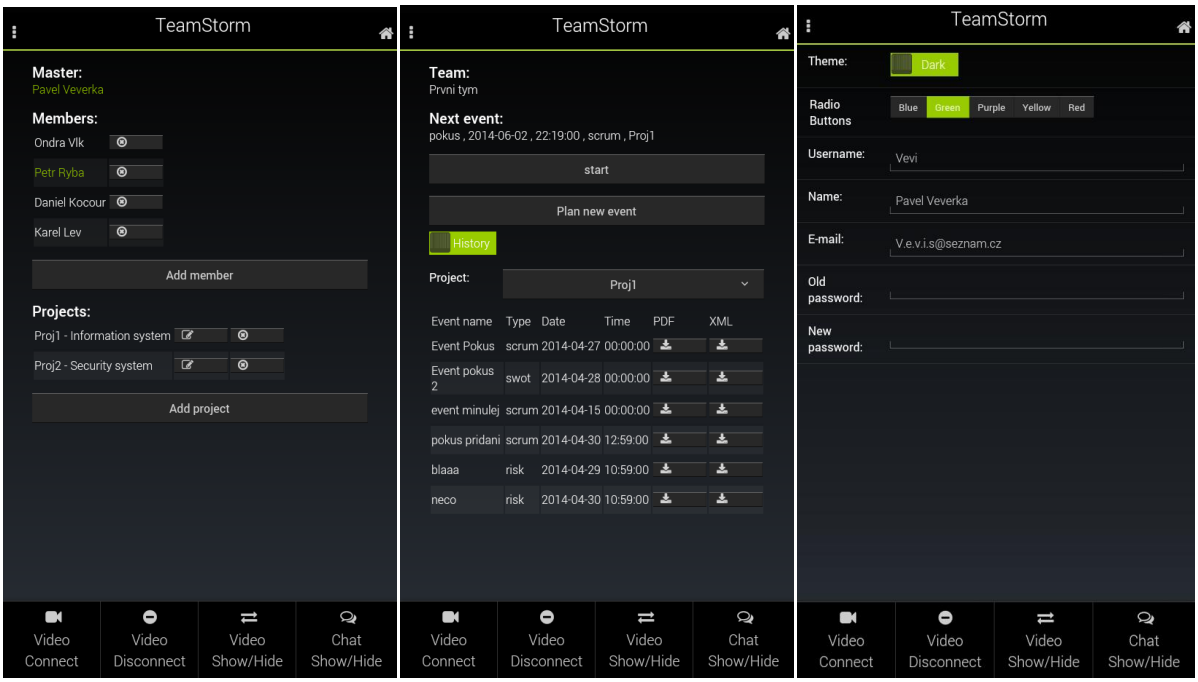
Obrázek 6.3 - Logo aplikace a přihlašovací formulář

Po přihlášení je uživatel přesměrován na hlavní stránku aplikace. Zde jsou zobrazeny hlavní údaje o přihlášeném uživateli, jeho týmu a následující události. Ve spodní části je zobrazen vedoucí týmu pod položkou *Master* a členové týmu pod položkou *Members*. Systém průběžně kontroluje uživatele, a pokud jsou přihlášeni, změní barvu jejich jména. Pro vedoucího týmu je důležité vědět, kteří členové týmu jsou přihlášeni, aby mohl řádně řídit naplánované události. Je pak v kompetenci týmu, jaký počet přítomných lidí je nutný pro zahájení události. Tento ukazatel je pak umístěn ve všech stránkách, kde je vhodné mít okamžitý přehled o jednotlivých uživatelích.



Obrázek 6.4 - Hlavní stránka

Přístup k dalším stránkám závisí na roli přihlášeného uživatele. Přistupuje se na ně pomocí zasouvacího navigačního menu na levé straně aplikace. Menu lze vyvolat tlačítkem třech teček v levém horním rohu, gestem "swipe right", neboli tahem doprava nebo pomocí hardwarového tlačítka na mobilním zařízení. Vedoucí týmu, tedy role "master", má přístup ke všem stránkám, kdežto člen týmu nemá přístup ke správě týmu a nemůže zakládat nové události. Administrátor má přístup pouze na hlavní stránku, kterou má upravenou pro účely správy týmů. Na stránce pro správu týmu je vedoucímu týmu umožněno přidávat či mazat členy týmu, dále má možnost přidávat, upravovat a mazat projekty. Nejdůležitější stránkou z pohledu funkčnosti aplikace je stránka s událostmi. Zde je horní části zobrazena následující událost. Vedoucí týmu může naplánovat novou událost, pomocí tlačítka *Plane new event*, nebo událost zahájit. Jakmile je událost zahájena, všem členům se objeví tlačítko *Enter*, které slouží ke vstupu do události. Jednotlivé typy událostí podrobně popisuje kapitola 6.6. Ve spodní části se nachází tabulka dalších naplánovaných událostí, nebo historie již uskutečněných událostí s možností stáhnutí souborů s výsledky. Další stránkou je pak historie diskuze, kde je možné nalézt podle zadaného data historii textové komunikace zadaného dne. Poslední stránkou je *Settings*, tedy nastavení vzhledu aplikace a nastavení účtu přihlášeného uživatel. Odhlášení lze provést položkou *Logout* v menu aplikace.



Obrázek 6.5 - Stránky zleva: Team, Events, Settings

Fungování jednotlivých elementů obstarávají dva JavaScript soubory. Hlavní funkce jsou obsaženy v souboru *index.js*. Po vytvoření elementů webové stránky je zavolána funkce *init()*, ve které proběhne kontrola používaného zařízení. Tato kontrola je nutná ze dvou důvodů. Prvním je import knihovny *opentok.js* potřebné pro provoz videokonference TokBox, kde je nutné použít rozdílnou variantu pro mobilní zařízení nebo webový prohlížeč, a knihovny *cordova.js*, která je nutná pro používání PhoneGap, ale nemůže být importována při využití desktopového webového prohlížeče. Druhý důvod je kvůli rozdílu indikaci připravenosti všech elementů stránky. Mobilní zařízení posílá po událost "deviceready", kterou je nutné odchytit. Webový prohlížeč využívá funkci *ready*.

```
function init() {

        if (CheckDevice()) {
                console.log('You are using a mobile device!');
                loadScript('./cordova.js');
                loadScript('./opentok.js');
                document.addEventListener("deviceready", onDeviceReady, false);
        } else {
                console.log('You are not using a mobile device!');
                loadScript('js/opentok.min.js');
                $(document).ready(function () { onDeviceReady(); });
        }
//Pridame EventListener ktery v priade, ze je app offline spusti v callbacku fci onOffline
document.addEventListener("offline", onOffline, false);
}
```

Kód 6.2 - Funkce Init()

Jakmile je dokument plně připraven spustí se funkce *onDeviceReady()*, která obsahuje veškeré chování elementů, jako jsou tlačítka, formuláře atd. Také je tu implementováno chování jednotlivých stránek při jejich vytvoření ve struktuře DOM nebo při přechodu na ně. Funkce uvedené v souboru *index.js* komunikují skrze zprávy technologie AJAX se serverovým souborem *server.php*. Účel komunikace je v tomto případě hlavně komunikace se serverovou databází, tedy přihlášení, správa uživatelů, projektů a událostí. Veškerá komunikace probíhá metodou POST protokolu HTTP, která data nepřidává do url adresy, ale zasílá je skrytě v HTTP požadavku. Komunikace pro řízení systému je implementována v souboru *communication.js*. Zde je vytvořen objekt *Control*, který obsahuje funkce pro řízení stavu systému. Nejdůležitější a nejobsáhlejší funkcí je *updateControl(),* sloužící pro udržování aktuálního stavu a rozklíčování přijaté zprávy. Opět je pro zasílání zpráv na server použito technologie AJAX a metody POST. Zprávy s řízením stavu a komunikací se zasílají na server do souboru *process.php*, který je zpracuje a zašle na ně odpověď.

6.6 Implementované nástroje

V této kapitole jsou popsány nástroje, které se podařilo implementovat do systému. Všechny vycházejí z požadavků a teoretického rozboru v kapitole 5.1. U každého nástroje jsou uvedené jednotlivé kroky, které jse nutné vykonávat v rámci jeho používání. Výsledkem je pak datový soubor formátu XML a PDF zpráva s výsledky. Pro spuštění je třeba, aby vedoucí týmu nejprve naplánoval událost. Ta nese informaci o datu, času konaní a o typu nástroje, který bude využíván. Jakmile je událost naplánovaná, automaticky se rozešle informační e-mail všem členům týmu a zaznamená se do budoucích událostí. Ve stanovený čas je možno začít s prací. Jakmile vedoucí týmu rozhodne, že je přítomno dostatečné množství členů, zahájí událost a členům je umožněno se do ní připojit. Další kroky jsou rozdílné pro jednotlivé typy nástrojů. Během práce s nástroji jsou všem uživatelům průběžně zobrazovány instrukce v horní liště stránky pod hlavičkou.

Plánovací poker

Po zahájení je vedoucímu týmu umožněno nahrát soubor s daty uložené ve formátu XML, které odpovídají notaci MS Project. Další možností je manuální zadávání jednotlivých fází projektu a jim odpovídajícím úkolům. Dále lze postupovat dle níže uvedených kroků.

Krok 1. Zadání fáze projektu

Fázi projektu vybere vedoucí týmu z položky *Phases*, pokud je použit import souboru, nebo ji do této položky vepíše ručně a potvrdí tlačítkem *Enter*. Pokud chce vedoucí týmu práci ukončit, může tak učinit tlačítkem *End Scrum* přejde na Krok 6.

Krok 2. Zadání úkolu

Úkol, podobně jako fáze, vedoucí týmu vybere z položky *Stories* u importu, nebo jej vepíše do položky *Enter story for play* a potvrdí tlačítkem. Pokud jsou všechny úkoly v rámci fáze zpracované a vedoucí týmu nechce žádné přidat, vrátí se tlačítkem *End Phase* na Krok 1.

Krok 3. Odhad

Po vložení je úkol zobrazen všem členům a ti odhadnou jeho míru náročnosti na stupnici označenou jako *Cards* a odešlou vedoucímu týmu. Hodnocení ostatních je jednotlivým členům týmu utajeno. Vedoucí týmu odhad neprovádí.

Krok 4. Odhalení

Příchozí odhad od člena týmu se vedoucímu zobrazí u seznamu *Members* a jakmile obdrží odhady od všech přítomných, odhalí všem celkové hodnocení.

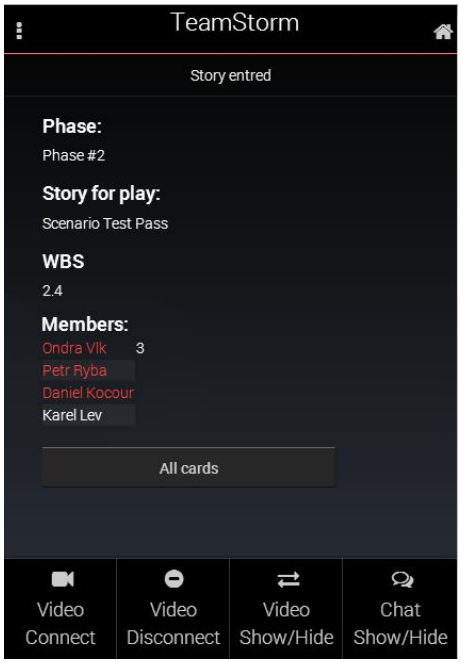
Krok 5. Vyhodnocení, diskuse

Pokud se jednotlivé odhady liší, je zahájena diskuse ohledně řešeného úkolu. Po uplynutí dvou minut, je hodnocení smazáno a tým se vrací na Krok 3. Jakmile se všechny odhady shodují, výsledek je odeslán na server a zapsán do datového souboru. Vedoucí týmu následně přejde tlačítkem *Next Story* na Krok 2.

Krok 6. Ukončení

Všechna zpracovaná data jsou uložena na serveru v XML souboru a je vygenerována zpráva v PDF dokumentu. Všem uživatelům se zobrazí vyskakovací okno, které jim oznámí konec události a přesměruje je na stránku *Events*.

Výstupem jsou data uložená ve formátu XML a zpráva v PDF dokumentu, která obsahuje tabulku s řešenými úkoly a jejich odhady. Odhady jednotlivých fází projektu jsou spočteny jako suma odhadů úkolů, které do dané fáze spadají. Tyto výsledky jsou ve zprávě ilustrovány sloupcovými grafy.



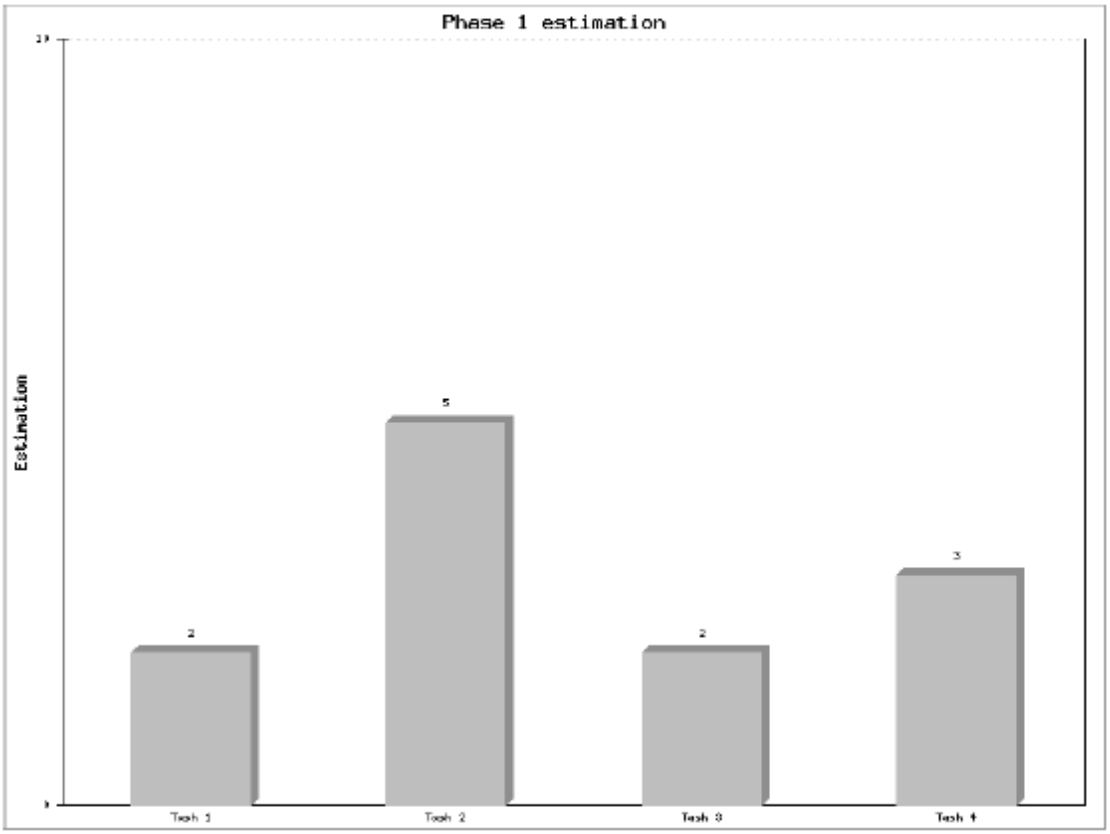
Obrázek 6.6 - Plánovací poker stránka

```
<Task>
  <Name>Phase #1</Name>
  <WBS>1</WBS>
  <Estimated>12</Estimated>
</Task>
<Task>
  <Name>Task 1</Name>
  <WBS>1.1</WBS>
  <Estimated>2</Estimated>
</Task>
```

Obrázek 6.7 - Ukázka uložení dat fáze a úkolu Plánovacího pokeru v XML

Task	Estimation
Phase #1	12
Task 1	2
Task 2	5
Task 3	2
Task 4	3

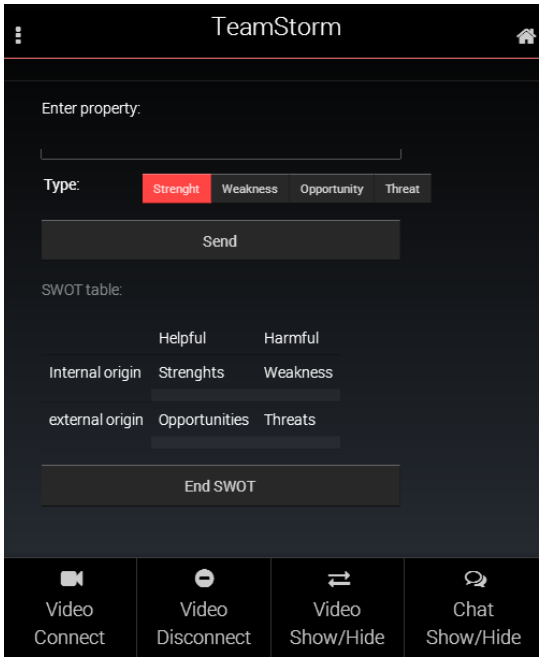
Obrázek 6.8 - Ukázka výstupu ve zprávě Plánovacího pokeru



Graf 6.1 - Ukázka generovaných grafů ve zprávě Plánovacího pokeru

SWOT Analysis

Analýza silných a slabých stránek je prováděna pro celý projekt a všemi členy týmu, tedy také vedoucím. Po zahájení má každý člen možnost zadat vlastnost projektu. Vedoucí týmů disponuje možností na základě diskuse s týmem odstranit vlastnost z tabulky. Analýza je ukončena vedoucím týmu tlačítkem *End SWOT*. Výsledná data jsou odeslána na server, kde jsou uložena v XML souboru, a je vygenerována PDF zpráva.



Obrázek 6.9 - Obrazovka SWOT Analýza

```
<Strenghts>
  <Strenght>Am if number no up period regard sudden better.</Strenght>
  <Strenght>Rather number can and set praise.</Strenght>
  <Strenght>Distrusts an it contented perceived attending oh.</Strenght>
  <Strenght>Indeed on people do merits to.</Strenght>
</Strenghts>
```

Obrázek 6.10 - Ukázka uložení SWOT dat v XML souboru

Agile Brainstorming

Po spuštění nástroje je umožněno vedoucímu týmu importovat soubor s daty. Oproti plánovacímu pokeru není rozlišována WBS struktura, tedy členění na fáze. Důvodem je to, že Agile Brainstorming je určený pro stanovení řešení konkrétního úkolu bez ohledu na fázi projektu.

Krok 1. Zadání úkolu

Vedoucí týmu vybere úkol k řešení z položky *Stories*, pokud je použit import souboru, jinak úkol vypíše do položky *Enter story for brainstroming*. Pokud chce vedoucí týmu práci ukončit může tak učinit tlačítkem *End Agile brainstorming,* přejde na Krok 7.

Krok 2. Návrh řešení

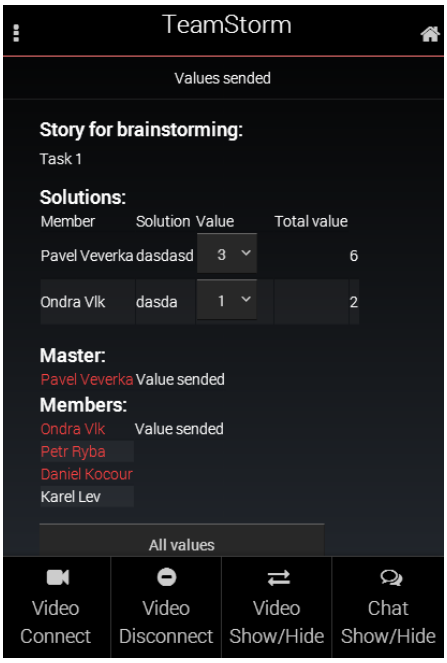
Všichni členové týmu, včetně vedoucího, napíší svůj návrh řešení a odešlou ho vedoucímu týmu. Jakmile všichni účastnící odešlou svá řešení, vedoucí týmu pomocí tlačítka *All solutions* přejde do dalšího kroku.

Krok 3. Hodnocení

Každý účastník ohodnotí jednotlivé návrhy řešení na stupnici od 1 do 5 a hodnocení opět zašle vedoucímu týmu, kterému se postupně zobrazuje, kdo své hodnocení už zaslal. Průběžně se hodnocení sčítají. Když jsou výsledky kompletní, vedoucí je tlačítkem *All values* vyhodnotí. Návrhy se tímto krokem smažou a zůstane pouze ten, který má největší hodnocení. Tlačítkem *Next task* jsou data odeslána na server a následuje Krok 1.

Krok 4. Ukončení

Všechna zpracovaná data jsou uložena na serveru v XML souboru a je vygenerována zpráva v PDF dokumentu. Všem uživatelům se zobrazí vyskakovací okno, které jim oznámí konec události a přesměruje je na stránku *Events*.



Obrázek 6.11 - Obrazovka Agile Brainstorming

```
<Task>
  <Name>Task 1</Name>
  <Solution>Am if number no up period regard sudden better.</Solution>
</Task>
<Task>
  <Name>Task 2</Name>
  <Solution>Certainty listening no no behaviour existence assurance situation is.</Solution>
</Task>
```

Obrázek 6.12 - Ukázka uložení řešení úkolu v XML souboru

Risk Analysis

Posledním nástroj spojuje Skórovací metodu s mapou rizik a Risk burn-down chart. Po zahájení je možné importovat soubor totožným způsobem jako u Plánovacího pokeru. Je tedy možné identifikovaná rizika importovat ze souboru, nebo je v rámci systému identifikovat pro každou fázi či iteraci projektu manuálně. Dále lze postupovat dle níže uvedených kroků.

Krok 1. Zadání fáze/iterace projektu

Fázi projektu vybere vedoucí týmu z položky *Phases*, pokud je použit import souboru, nebo ji do této položky vepíše ručně a potvrdí tlačítkem *Enter*. Pokud chce vedoucí týmu práci ukončit, může tak učinit tlačítkem *End Risc analysis* přejde na Krok 4.

Krok 2. Zadání rizika

Úkol, podobně jako fáze, vedoucí týmu vybere z položky *Stories* u importu, nebo jej vepíše do položky *Enter story for play* a potvrdí tlačítkem. Pokud jsou všechna rizika v rámci fáze ohodnocena a vedoucí týmu nechce žádné přidat, vrátí se tlačítkem *End Phase* na Krok 1.

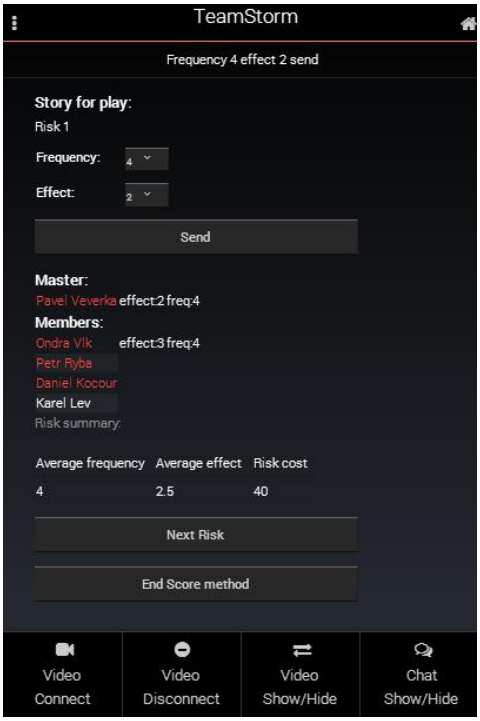
Krok 3. Hodnocení

Jakmile je riziko zadané všichni účastníci včetně vedoucího týmu ohodnotí míru výskytu rizika a jeho dopadu. Hodnoty každý člen týmu odešle vedoucímu, kde se postupně vypočítávají průměrné hodnoty výskytu a dopadu rizika a vynásobením těchto hodnot navzájem je vypočítána cena rizika. Tlačítkem *Next risk* jsou data odeslána na server a následuje Krok 2.

Krok 4. Ukončení

Všechna zpracovaná data jsou uložena na serveru v XML souboru a je vygenerována zpráva v PDF dokumentu. Všem uživatelům se zobrazí vyskakovací okno, které jim oznámí konec události a přesměruje je na stránku *Events*.

Výstupem ve zprávě je tabulka obsahující výsledky hodnocení jednotlivých rizik, mapa rizik a risk brun-down chart.



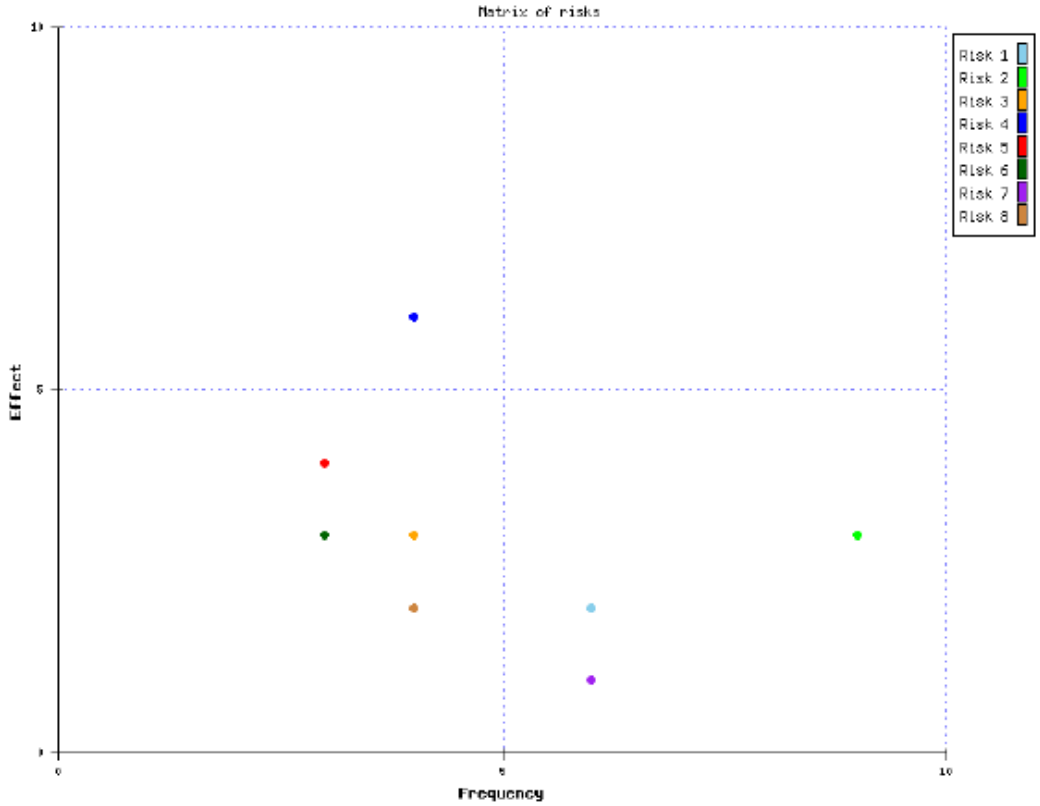
Obrázek 6.13 - Stránka Risk analysis

Risk	Frequency	Effect	Cost
Iteration #1			43
Electricity blackout	5	2	10
Workers unexpected day off	8	3	24
Workers ill	3	3	9

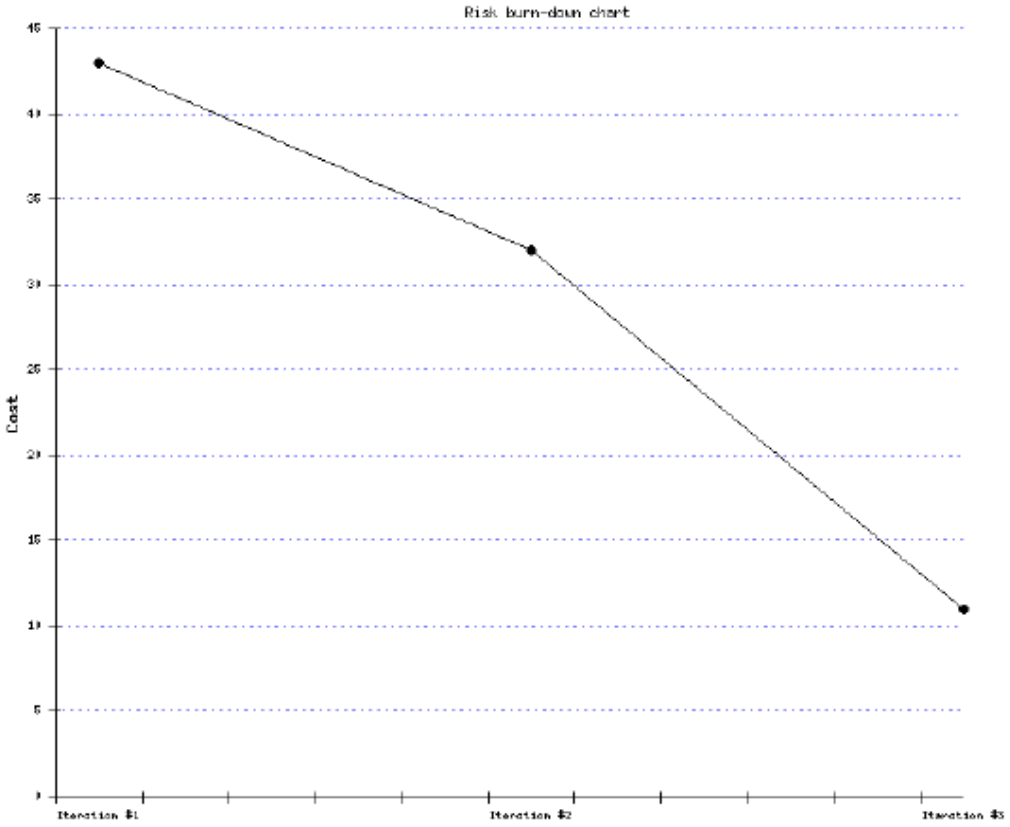
Obrázek 6.14 - Ukázka výstupu ve zprávě Risk analysis

```
<Risk>
  <Name>Iteration #1</Name>
  <WBS>1</WBS>
  <Frequency>0</Frequency>
  <Effect>0</Effect>
  <Cost>43</Cost>
</Risk>
<Risk>
  <Name>Risk 1</Name>
  <WBS>1.1</WBS>
  <Frequency>5</Frequency>
  <Effect>2</Effect>
  <Cost>10</Cost>
</Risk>
```

Obrázek 6.15 - Ukázka uložení hodnocení rizik v XML souboru



Obrázek 6.16 - Mapa rizik ve zprávě Risk analysis



Obrázek 6.17 - Risk burn-down chart ve zprávě Risk analysis

6.7 Požadavky systému

Pro provoz systému je nutný webový server, který podporuje PHP5 a MySQL. Klientská část vyžaduje povolený JavaScript. Aplikace byla testována ve webových prohlížečích Internet Explorer, Google Chrome a Opera. Minimální verze systému Android v mobilním zařízení je 4.0 Ice Cream Sandwitch, avšak cílovou verzí je 4.4 KitKat.

Pro demonstraci systému byla založena doména na free serveru přístupná na adrese vevis.php5.cz. Na této adrese je možné přistoupit k aplikaci z webového prohlížeče. Databáze je umístěna na stejném serveru.

- Server: adresa - www.php5.cz
  doména - vevis.php5.cz
- Databáze: adresa - http://nibiru.zarea.net/sqladmin
  server - localhost
  jméno - czvevis
  heslo - 8259E0F149

Instalaci aplikace na mobilní zařízení lze provést pomocí souboru TeamStorm.apk, který je umístěn na přiloženém DVD. Soubor je nutné nahrát do mobilního zařízení a po spuštění souboru je zahájena instalace aplikace. Během instalace je zobrazena výzva udělení oprávnění k nativním funkcím mobilního zařízení. Následně je vytvořena ikona, která slouží ke spuštění aplikace.

Testování aplikace proběhlo v nástroji Testdroid cloud [39] na sedmi virtuálních zařízení. Výsledky testu zobrazuje Tabulka 6.1. Test se skládal ze spuštění aplikace, přihlášení uživatele, odeslání zprávy a připojení videokonference. Z výsledků je patrné, že ač je aplikace cílena na nejnovější verzi Android 4.4, test na této verzi trval nejdéle. Nejpravděpodobnější příčinou je zastaralý zásuvný modul TokBox zajišťující provoz videokonference.

Přístroj	Verze Android	Instalace aplikace	Trvání testu
Asus Google Nexus 7 ME370T	4.1.2	10s	59s
Asus Google Nexus 7 ME370T	4.3	10s	1m 1s
LG Google Nexus 4 E960	4.3	6s	1m
LG Google Nexus 5 D821	4.4	3s	3m 29s
Samsung Galaxy Nexus GT-I9250	4.0.4	16s	1m 1s
Samsung Galaxy Nexus GT-I9250	4.2.2	8s	59s
Samsung Nexus S 4G SPH-D720	4.1.1	12s	1m 1s

Tabulka 6.1 - Výsledky testu v nástroji Testdroid

6.8 Modelová situace

Výsledný systém byl otestován na modelové situaci řešení projektu virtuálního týmu. Tým se skládal ze čtyř členů a jednoho vedoucího. K práci byl použit PC s Windows 8 a webovým prohlížečem Google Chrome, PC se systémem Windows XP a webovým prohlížečem Internet Explorer, mobilní telefon ZTE Grand X IN, mobilní telefon Samsung Galaxy S3 mini a tablet Prestigo MultiPad 4 ultra quad. Tým byl nejprve administrátorem zaveden do systému a bylo mu přiděleno zkušební TokBox ID. Vedoucí týmu následně založil projekt s názvem Proj1. Prvním nástrojem, který tým použil, byla SWOT analýza. Pomocí této analýzy tým identifikoval možné silné a slabé stránky projektu a z nich vyplývající hrozby a příležitosti. SWOT analýzu tým provedl hned na začátku projektu, aby si co nejdříve definoval základní fakta o projektu. Dále tým pomocí nástroje Scrum poker sestavil strukturu úkolů v jednotlivých fázích projektu a odhadl jejich náročnost. Z vytvořené zprávy tímto nástrojem, si udělal tým přehled o plánované práci v celém projektu i jednotlivých fázích. Jakmile byla stanovena struktura projektu, tým identifikoval rizika projektu v jednotlivých iteracích a pomocí nástroje Analýzy rizik je ohodnotil mírou pravděpodobnosti výskytu a možné škody, která by mohla vzniknout. Tým ze zprávy zjistil, že je nutné zaměřit se na několik rizik, ale že výsledný Burn-down chart má klesající tendenci, což je žádoucí. Rizika, ke kterým bylo nutné nalézt opatření, byla zpracována v nástroji Agile brainstorming. Tým tak získal počáteční dokumentaci a mohl zahájit práci na projektu. V každé iteraci může znovu hledat rizika, plánovat, hodnotit a odhadovat, aniž by se musel fyzicky scházet.

Obrázky z testování na modelové situaci jsou uvedeny v příloze, kde je systém zobrazen na různých testovaných zařízeních a platformách. Datové soubory a zprávy z testování jsou na přiloženém DVD.

7 Závěr

Využívání mobilních zařízení je dnes velmi na vzestupu. Praktičnost mít u sebe svá data a možnost s nimi ihned pracovat, objevuje čím dál více lidí. S rozšířením rychlého internetového připojení také vzrostla oblíbenost a s ní rozšíření nabídky tak zvaných "cloud" služeb, kde uživatel má možnost využívat nepřeberné množství služeb, které by mu jinak byly nepřístupné. S rostoucím výkonem a neustálým zvětšováním obrazovek, se mobilní zařízení stává plnohodnotnou pracovní stanicí dostupnou odkudkoli. Tím vzniká obrovský potenciál těchto zařízení hlavně pro management, jelikož pracovníci v tomto odvětví jsou často na cestách a potřebují mít neustále k dispozici přehled o dění jejich pracovníků a stavu jejich práce. Při řešení projektů, zejména v oblasti softwaru, jsou dnes rozšířené Agilní metodiky, které kladou na časté, nicméně krátké, setkávání a porady. To přináší problémy hlavně virtuálním týmům, které jsou mnohdy z různých koutů světa.

Tato práce řeší problematiku využití brainstormingových technik plánování a odhadování pro virtuální týmy v prostředí Agilních metodik. Cílem bylo prozkoumat v současné době používané Agilní metodiky a navrhnout systém, který by podporoval brainstorming virtuálních týmů. Aby byl systém co nejflexibilnější byl navržen pro běh na různých typech zařízení a operačních systémech. Proto byla druhá část této práce věnována mobilním zařízením a možnostem multiplatformního vývoje. Dále byl navrhnut a implementován systém využívající několik nástrojů pro brainstorming a plánování. Tento systém byl otestován na několika různých zařízení a demonstrován na modelovém případě. Výhodou tohoto systému je přehledné zpracování výstupů v datovém souboru a vytvoření zprávy, která lze plnohodnotně využít jako část dokumentace k projektu.

Další rozšíření systému by mohlo být přidání více metod pro plánování a odhadování. Systém by bylo také vhodné doplnit o komplexní kontrolu a agilní řízení projektů, jako je zadávání úkolů, časový management, kanban tabuli a jiné. Pro zlepšení bezpečnosti dat je vhodné přidat do komunikace nějakou formu šifrování. Dále by jistě uživatelé ocenili podporu importu více typů souborů z ostatních systémů pro řízení projektů než jen z MS Project. Také by bylo vhodné věnovat další čas na optimalizaci aplikace. Pro praktické nasazení systému by bylo dobré registrovat placenou doménu určenou výhradně pro potřeby systému a aplikaci zveřejnit na Google Play, tedy marketu Android aplikací.

Při práci na tomto systému jsem objevil spoustu nových poznatků a využil své znalosti z oblasti managementu projektů. Věřím, že má systém velký potenciál, jelikož spojuje moderní technologie a vyplňuje mezeru ve stávajících systémech pro řízení projektů.

Literatura

[1] Kolektiv autorů: *Manifest Agilního vývoje software* [online], 2001, [cit. 2014-01-10]. Dostupné z: http://agilemanifesto.org/

[2] COHN, M.: *Agile Estimating and Planning*, Prentice Hall, 2007, ISBN 0-1314-7941-5.

[3] *A Guide to The Project Management Body of Knowledge*, Fifth Edition, Project Management Institute, 2013, ISBN 978-1-935589-67-9.

[4] COHEN D., LINDVALL M. a COSTA P., 2004, "An Introduction to Agile Methods." Advances in Computers: 1-66.

[5] COCKBURN, A., *Crystal methodologies*, [online], 2009, [cit. 2014-01-10], Dostupné z: http://alistair.cockburn.us/Crystal+methodologies

[6] COCKBURN, A., *Crystal light methods*, [online], 2009, [cit. 2014-01-10], Dostupné z: http://alistair.cockburn.us/Crystal+light+methods

[7] COCKBURN, A. *Crystal Clear: A Human-Powered Methodology for Small Teams*. 2004.

[8] SCHWABER, K a SUTHERLAND, J. *The Scrum Guide*™ [online]. 2013 [cit. 2014-01-10]. Dostupné z: https://www.scrum.org/scrum-guide

[9] BECK, K a ANDRES, C. *Extreme programming explained: embrace change*. 2nd ed. Addison-Wesley, 2005, ISBN 03-212-7865-8.

[10] ANDERSON, D., *Kanban - Successful Evolutionary Change for your Technology Business*. 2010. Blue Hole Press. ISBN 0-9845214-0-2.

[11] VERSIONONE, Inc. *VersionOne* [online]. 2014 [cit. 2014-01-10]. Dostupné z: http://www.versionone.com/

[12] AXOSOFT, LLC. *OnTime Scrum* [online]. 2002 - 2013 [cit. 2014-01-10]. Dostupné z: http://www.ontimenow.com/scrum

[13] SCRUMDO, LLC. *ScrumDo* [online]. 2012 [cit. 2014-01-10]. Dostupné z: https://www.scrumdo.com/home

[14] MICROSOFT. *Project Professional 2013* [online]. 2014 [cit. 2014-01-10]. Dostupné z: http://office.microsoft.com/en-us/project-help/professional-project-management-desktop-software-project-professional-FX103797571.aspx

[15] Android Pushes Past 80% Market Share While Windows Phone Shipments Leap 156.0% Year Over Year in the Third Quarter, According to IDC. IDC. *IDC Analyze the Future* [online]. 2013 [cit. 2014-01-10]. Dostupné z: http://www.idc.com/getdoc.jsp?containerId=prUS24442013

[16] GARGENTA, Marko. *Learning Android*. Sebastopol, Calif.: O'Reilly, c2011, xvii, 245 p. ISBN 14-493-9050-1.

[17] MILLIONS OF CHOICES. GOOGLE. *Google Play* [online]. 2014 [cit. 2014-01-10]. Dostupné z: https://play.google.com/intl/en-US_us/about/apps/index.html

[18] CAMERON, Rob. *Pro Windows Phone 7 development*. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2011. ISBN 14-302-3219-6.

[19] Nokia and Microsoft Announce Plans for a Broad Strategic Partnership to Build a New Global Mobile Ecosystem. *Microsoft News Center* [online]. Feb. 10, 2011 [cit. 2014-01-10]. Dostupné z: http://www.microsoft.com/en-us/news/press/2011/feb11/02-11partnership.aspx

[20] MORRIS, Ben. *The Symbian OS architecture sourcebook: design and evolution of a mobile phone OS*. Hoboken, NJ: J. Wiley, xxii, 607 s. ISBN 978-0-470-01846-0.

[21] Mozilla and Partners to Bring Firefox OS to New Platforms and Devices. In: *The Mozilla Blog* [online]. JAN 6 2014 [cit. 2014-01-10]. Dostupné z: https://blog.mozilla.org/blog/2014/01/06/mozilla-and-partners-to-bring-firefox-os-to-new-platforms-and-devices/

[22] CONWAY, Joe a Aaron HILLEGASS. *IOS programming: the big merd ranch guide*. Atlanta: Big Nerd Ranch, 2012. ISBN 978-0-321-82152-2.

[23] ANDREW WHITECHAPEL, Sean McKenna a Aaron HILLEGASS. *Windows Phone 8 development internals: the big merd ranch guide*. Sebastopol, California: O'Reilly Media, Inc, 2012, xviii, 589 s. ISBN 978-073-5676-237.

[24] Aktualizace na Symbian Belle, nově Nokia Belle, definitivně na scéně. *Mobilizujeme.cz* [online]. 2012 [cit. 2014-01-10]. Dostupné z: http://mobilizujeme.cz/clanky/aktualizace-na-symbian-belle-bude-az-zacatkem-roku-2012/

[25] ADOBE SYSTEMS INC. *PhoneGap* [online]. 2014 [cit. 2014-01-10]. Dostupné z: phonegap.com

[26] GHATOL, Rohit a Yogesh PATEL. *Beginning PhoneGap: mobile web framework for JavaScript and HTML5*. New York: Distributed to the book trade worldwide by Springer Science Business Media, xii, 332 p. Books for professionals by professionals. ISBN 14-302-3903-4.

[27] APPCELERATOR INC. *Appcelerator* [online]. 2008 - 2014 [cit. 2014-01-10]. Dostupné z: http://www.appcelerator.com/

[28] Titanium Platform Overview. *Appcelerator Documentation* [online]. 2013 [cit. 2014-01-10]. Dostupné z: http://docs.appcelerator.com/titanium/3.0/?_escaped_fragment_=/guide/Titanium_Platform_Overview

[29] XAMARIN INC. *Xamarin* [online]. 2013 [cit. 2014-01-10]. Dostupné z: http://xamarin.com/

[30] Building Cross Platform Applications. *Xamarin Developer Center* [online]. 2013 [cit. 2014-01-10]. Dostupné z: http://docs.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/

[31] SENCHA INC. *Sencha* [online]. 2014 [cit. 2014-01-10]. Dostupné z: http://www.sencha.com/

[32] DOLEŽAL, Jan, Pavel MÁCHAL a Branislav LACKO. *Projektový management podle IPMA*. 2., aktualiz. a dopl. vyd. Grada, 2012, 526 s. Expert (Grada). ISBN 978-80-247-4275-5.

[33] Succeeding with Agile. MOUNTAINGOAT SOFTWARE. *Mike Cohn's Blog* [online]. 2010 [cit. 2014-01-10]. Dostupné z: http://www.mountaingoatsoftware.com/blog/managing-risk-on-agile-projects-with-the-risk-burndown-chart

[34] TOKBOX, Inc. *TokBox* [online]. 2014 [cit. 2014-05-10]. Dostupné z: http://tokbox.com

[35] FPDF library, [online]. 2013 [cit. 2014-05-10]. Dostupné z: http://www.fpdf.org/

[36] PHPlot, [online]. 2012 [cit. 2014-05-10]. Dostupné z: http://phplot.sourceforge.net/

[37] Getting started with PhoneGap in Eclipse for Android, [online]. 2012 [cit. 2014-05-10]. Dostupné z: http://www.adobe.com/devnet/html5/articles/getting-started-with-phonegap-in-eclipse-for-android.html

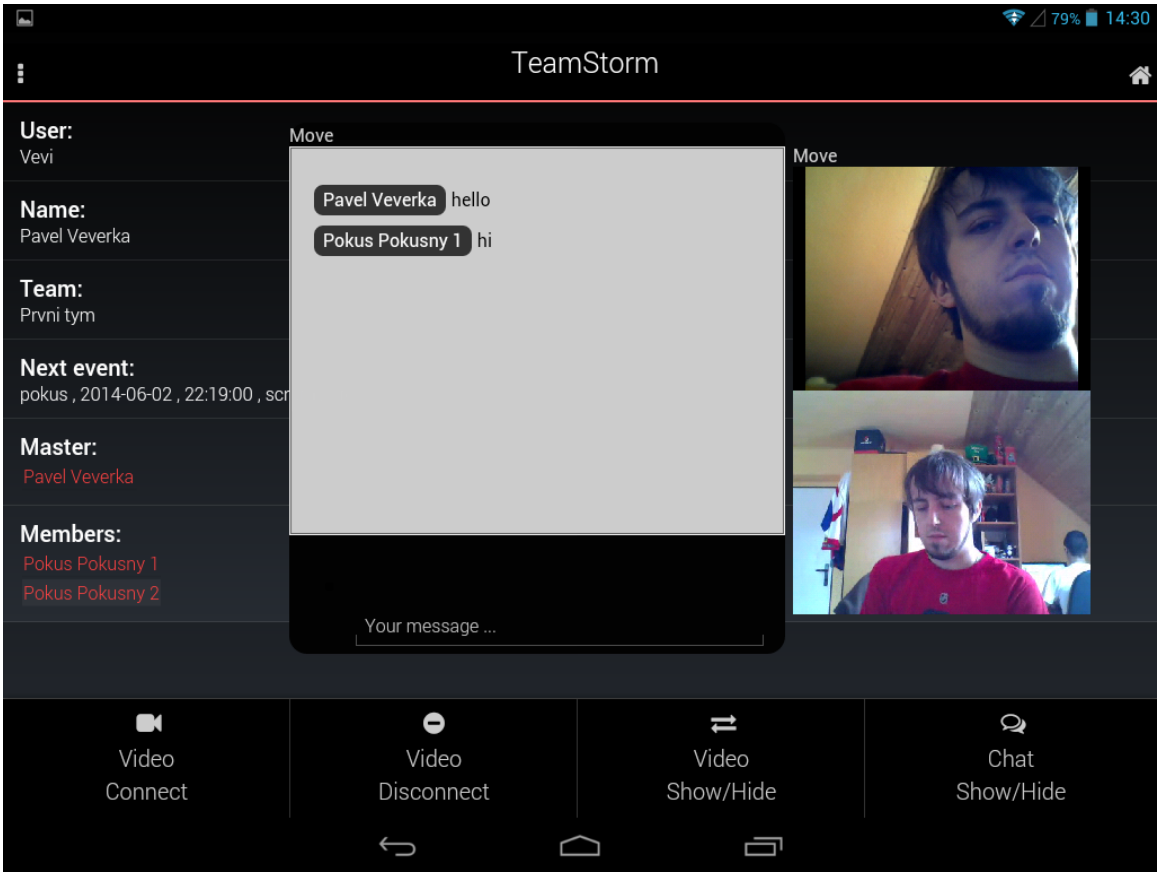
[38] CC licensing, [online]. 2014 [cit. 2014-05-10]. Dostupné z: http://creativecommons.org/licenses/by-sa/3.0/

[39] TESTDROID, [online]. 2014 [cit. 2014-05-10]. Dostupné z: https://cloud.testdroid.com

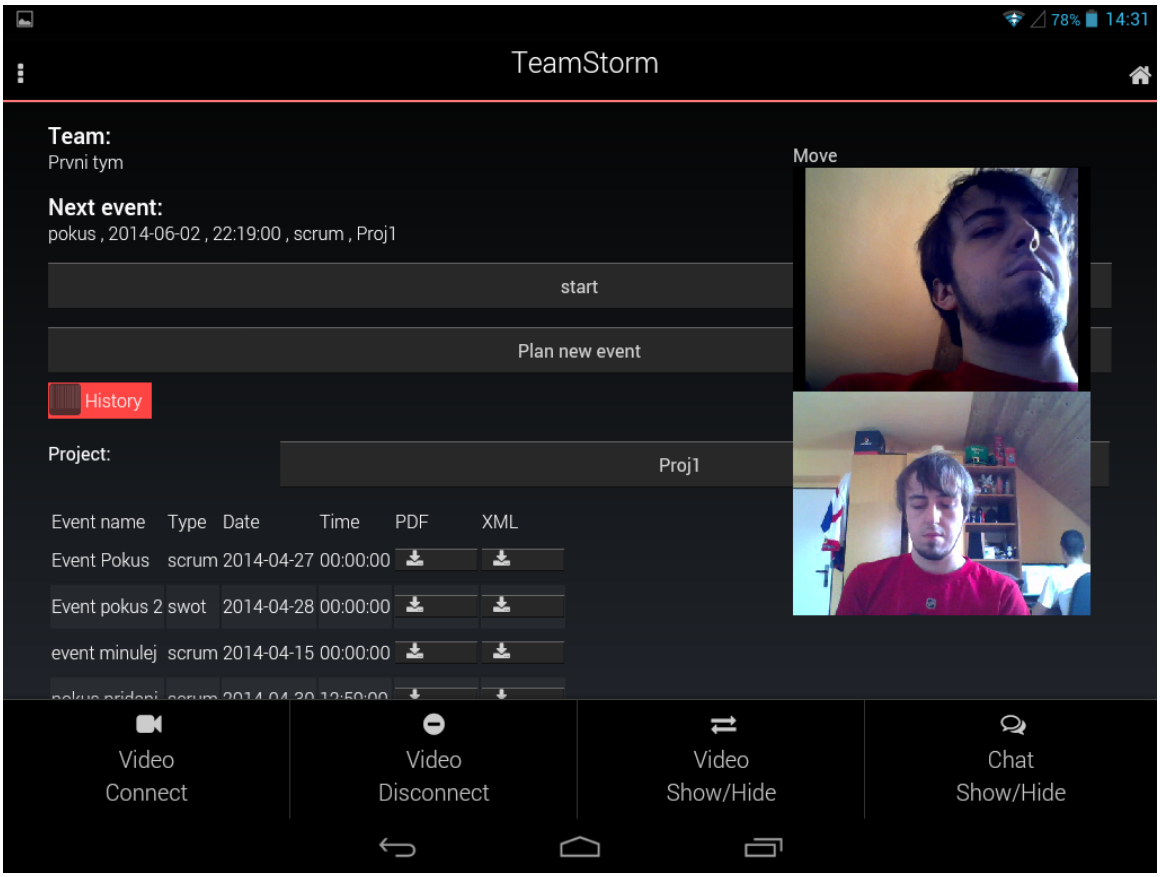
Seznam obrázků

Seznam příloh

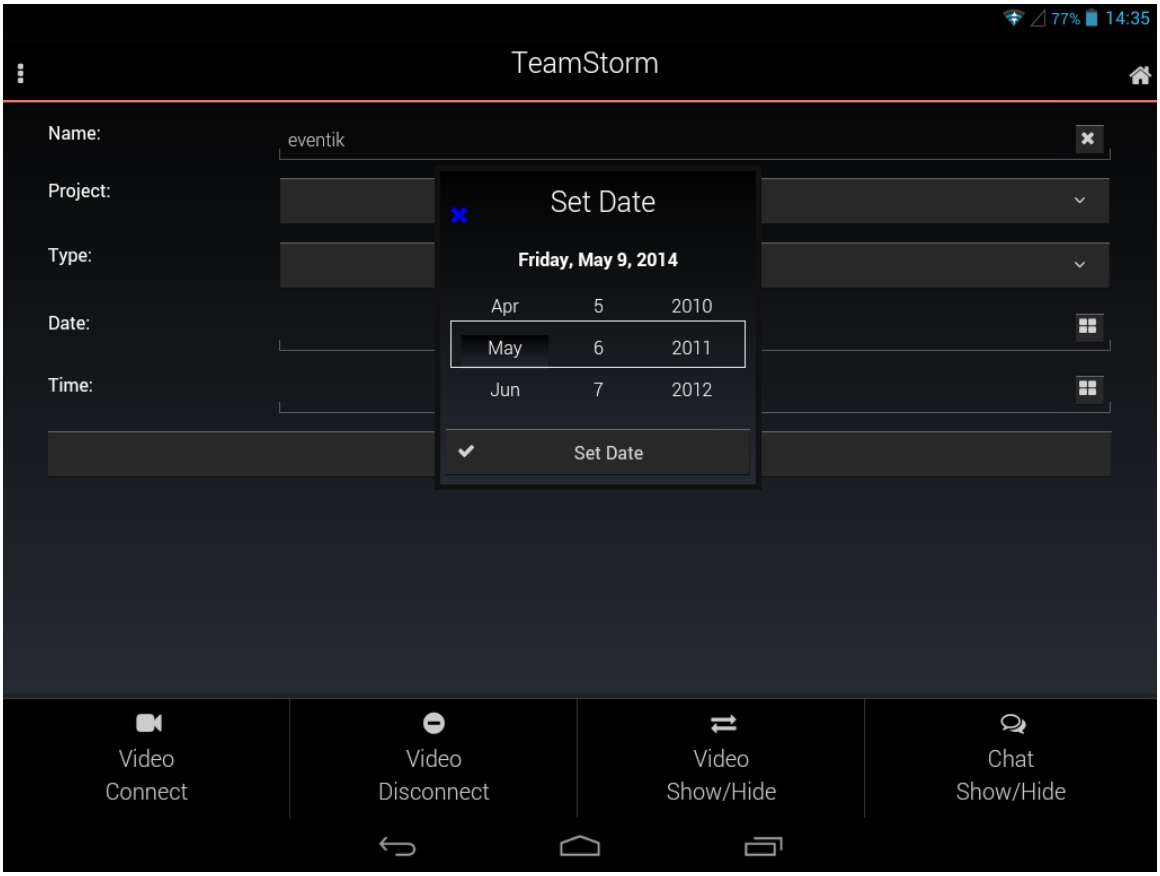
Příloha 1 - Ukázka multiplatformního použití



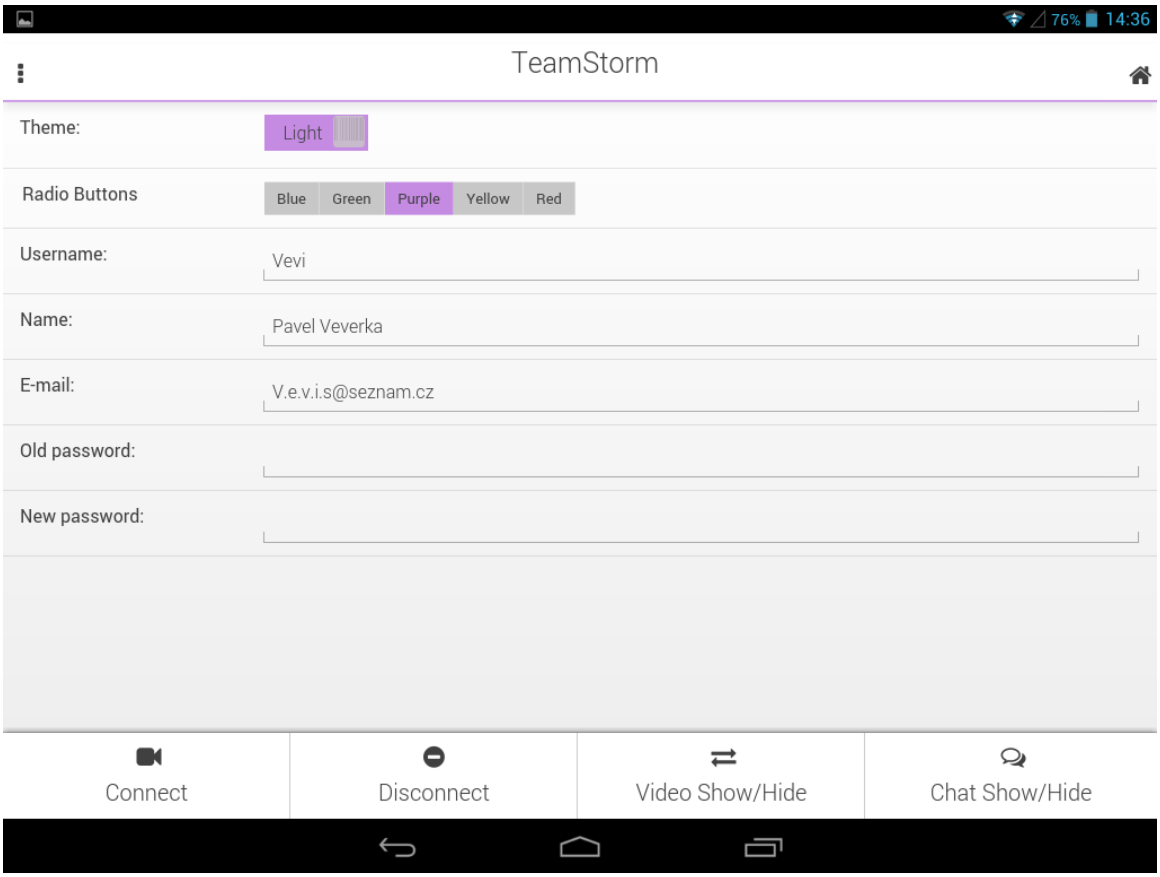
Příloha 2 - Ukázka aplikace na tabletu Hlavní strana a komunikace



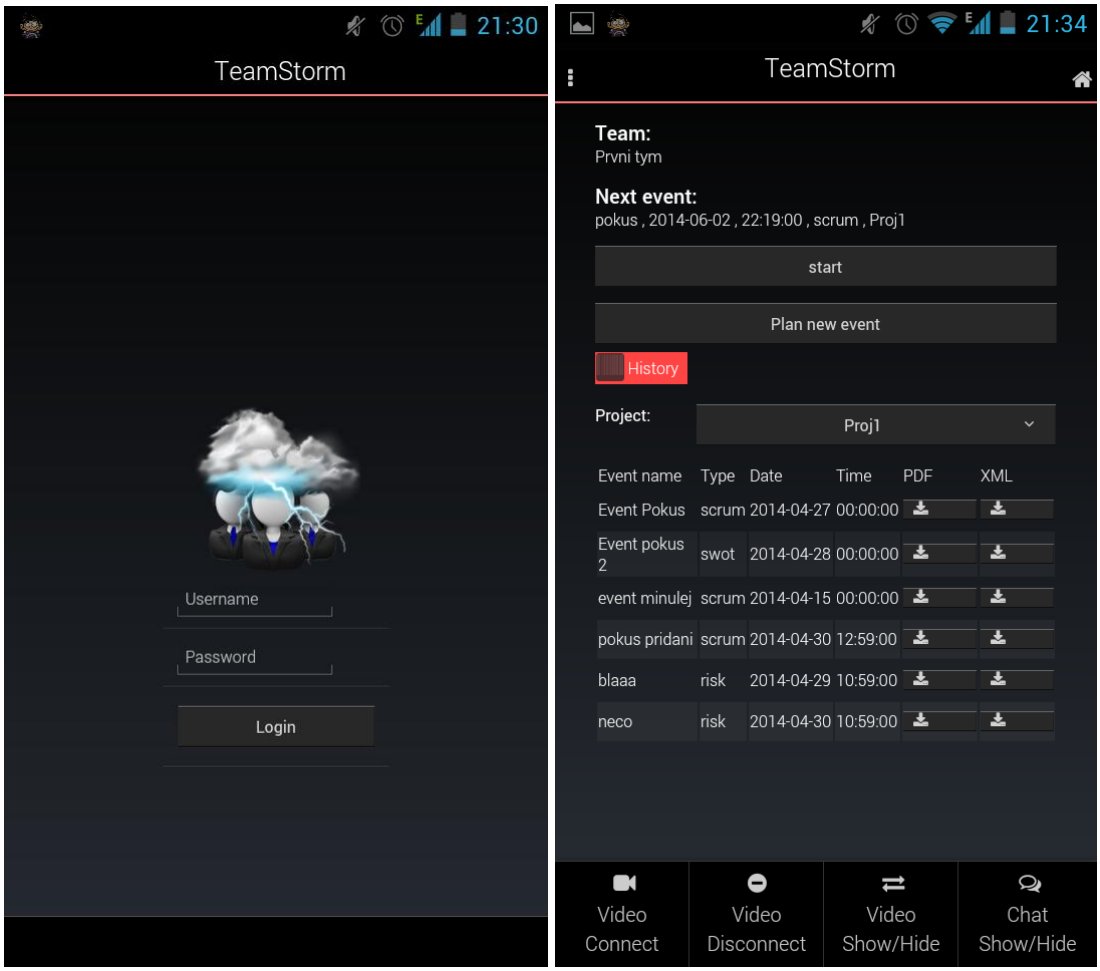
Příloha 3 - Ukázka aplikace na tabletu stránka Events a videokonference



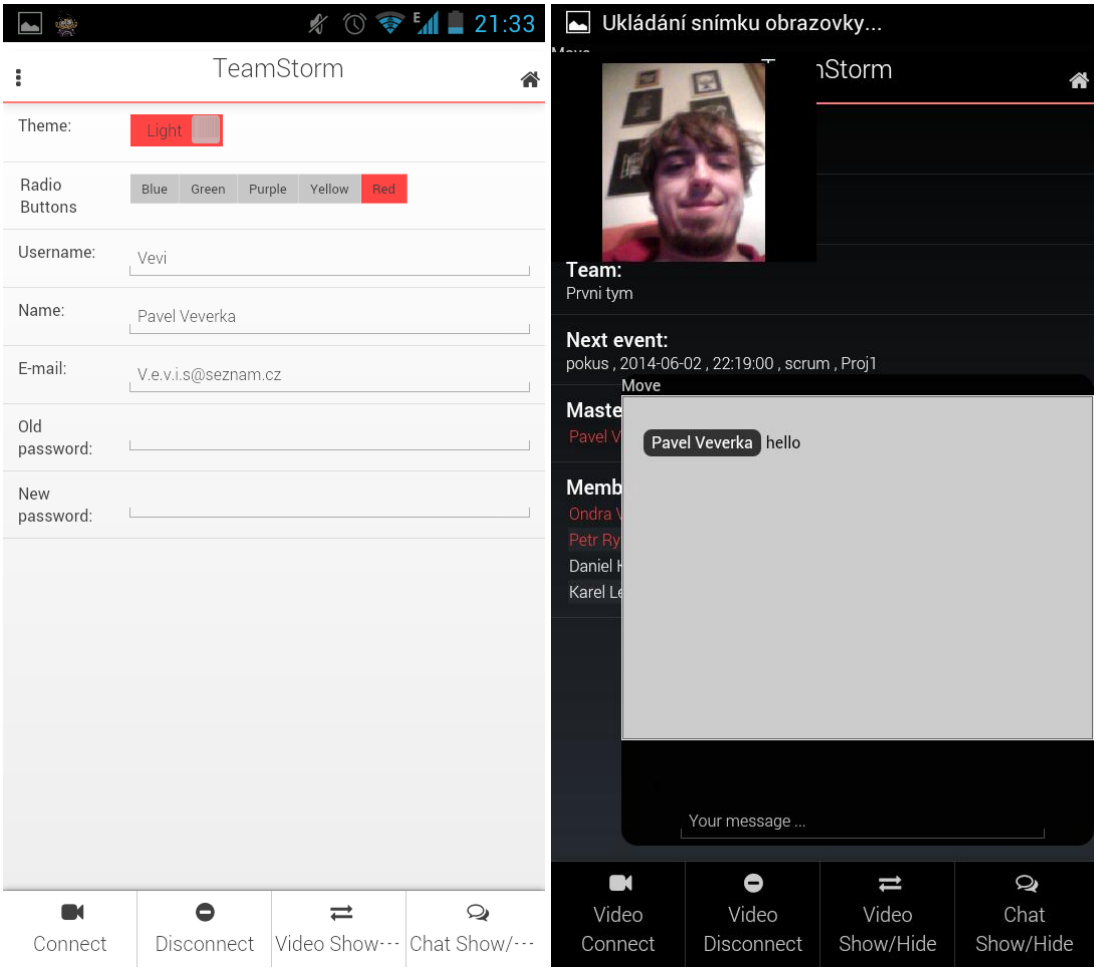
Příloha 4 - Ukázka aplikace na tabletu Date picker



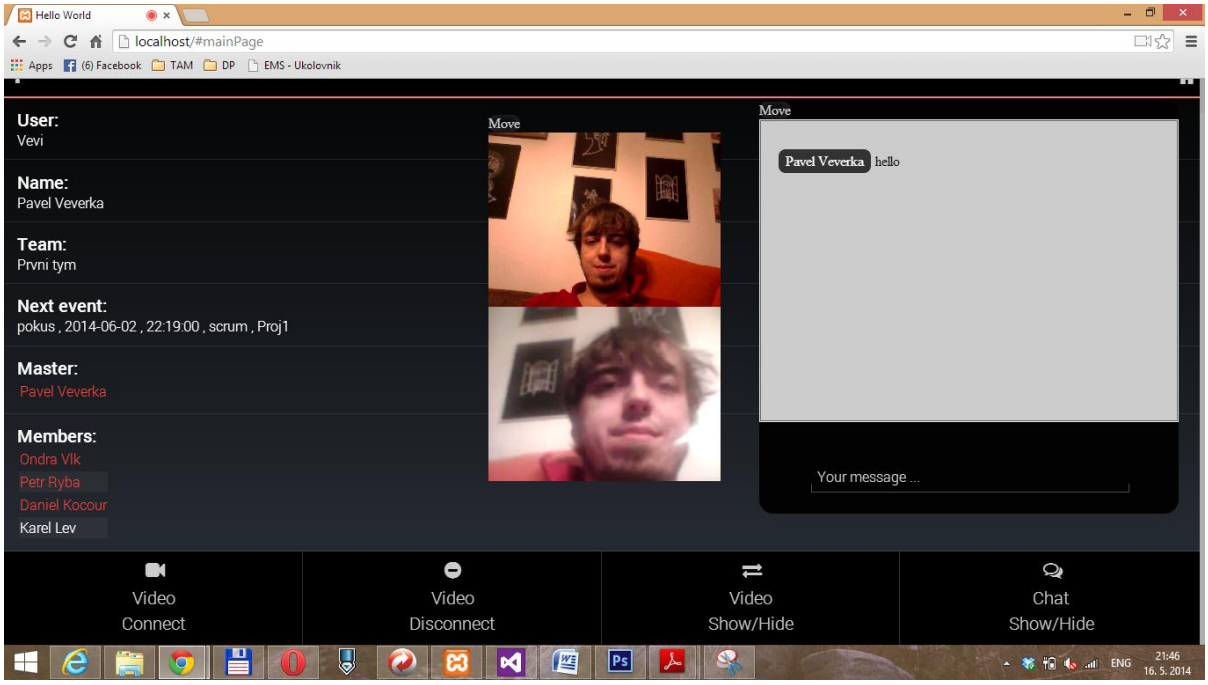
Příloha 5 - Ukázka aplikace na tabletu bílé provedení



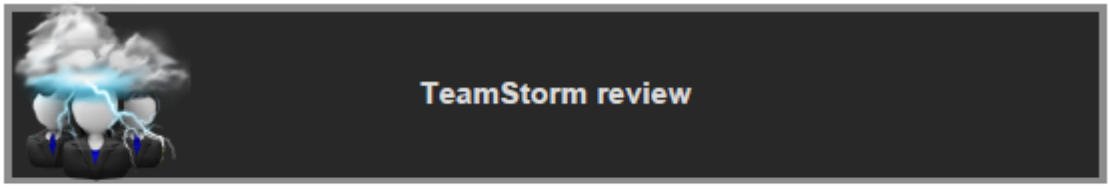
Příloha 6 - Ukázka aplikace na mobilním telefonu stránka Login a Events



Příloha 7 - Ukázka aplikace na mobilním telefonu bílé provedení a komunikace



Příloha 8 - Ukázka aplikace na desktopu otevřená v Google Chrome



Scrum Pocker

Event	Date	Time
pokus	2014-06-02	22:19:00

Team	Leader	Members
Prvni tym	Pavel Veverka	• Ondra Vlk • Petr Ryba • Daniel Kocour • Karel Lev

Project	Description
Proj1	Testing project

Příloha 9 - Úvodní strana PDF zprávy

Příloha 10 - Logo pod licencí Creative Common

Obsah DVD

- *xvever04.pdf* - Dokumentace k diplomové práci v PDF.
- *Aplikace* - Složka se zdrojovými kódy.
- *Aplikace/TeamStorm.apk* - Instalační soubor pro Android zařízení.
- *Aplikace/teamstorm* - Složka s projektem PhoneGap.
- *Aplikace/server* - Složka se soubory určené pro server systému.
- *Aplikace/teamstorm.sql* - Soubor s testovací databází.
- *Výstupy* - Složka s výstupy z modelové situace.