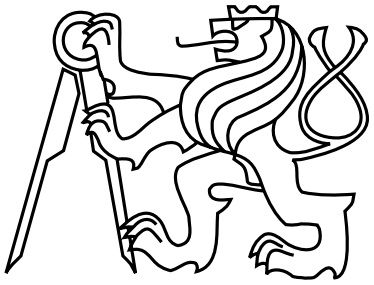
Na tomto místě bude oficiální zadání vaší práce

- Toto zadání je podepsané děkanem a vedoucím katedry,
- musíte si ho vyzvednout na studiijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Diplomová práce

Víceuživatelská aplikace typu klient-server pro záznam a analýzu testů použitelnosti

Bc. Ondřej Krejčíř

Vedoucí práce: Ing. Ivo Malý, Ph.D.

Studijní program: Otevřená informatika, Magisterský

Obor: Softwarové inženýrství

11. května 2015

Poděkování

Na tomto místě bych chtěl poděkovat mému vedoucímu práce Ing. Ivo Malému, Ph.D. za velice cenné rady, odborné připomínky, trpělivost, ochotu a za podporu, kterou mi dodával po celou dobu mého studia.

Dále bych chtěl poděkovat svým rodičům, kteří mi umožnili studium na této škole a za jejich podporu ve chvílích, kdy byla nejvíce potřeba. V neposlední řadě chci poděkovat svým přátelům, kteří mi důvěřovali a podporovali mne po celou dobu studia.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 11. 5. 2015 .....

Abstract

Every year, user interface testing is becoming an inherent component of project creation process. In the user interface designing process, the testing intensity and frequency are increasing and the amount of data which describes the progress of an individual applicability is so rising. Precisely because of the huge amount of data which need to be elaborated, assessed and saved even the importance of mediation applications is growing.

Creation of an application which would be able to carry out basic operations with typical categories of data that could come into existence during the user interface testing process is the aim of this diploma thesis. Such an application is intended for operatives who participate in user interface research. A significant emphasis is put on data distribution within a team because, as a rule, extensive tests come under the scrutiny of multi-membered specialized groups and not of individuals.

Abstrakt

Testování uživatelského rozhraní se každým rokem stává stále samozřejmější součástí procesu tvorby projektů. V procesu návrhu uživatelského rozhraní se stále více a častěji testuje a díky tomu velmi rychle roste objem dat, která popisují průběh jednotlivých testů použitelnosti. Právě kvůli obrovskému množství dat, které je potřeba zpracovat, vyhodnotit a uložit, roste i význam aplikací, které toto všechno umožní.

Cílem této diplomové práce je vytvořit aplikaci, která bude schopna provádět základní operace nad typickými druhy dat, které mohou během uživatelského testování vzniknout. Tato aplikace bude určena pro výzkumné pracovníky, kteří se přímo podílí na výzkumu uživatelského rozhraní. Velký důraz bude kladen na distribuci dat v rámci týmu, protože

rozsáhlá testování zpravidla nejsou doménou jednotlivců, ale více-členných skupin specialistů.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V posledních letech je výzkumu grafického rozhraní a hlavně interakce uživatelů s ním přikládán stále větší význam. Tomuto testování se věnuje stále více pracovníků a vznikají specializované oddělení pro návrh uživatelského rozhraní i u menších firem. Firmy zjišťují, že včasné testování uživatelského rozhraní jim může ušetřit spoustu prostředků v budoucnosti i přes vynaložení nemalého úsilí na začátku vývoje projektu.

Testování uživatelského rozhraní je často chápáno jako výzkum určitého již vytvořeného grafického prostředí pomocí pozorování interakce uživatelů. Během takového testování lze získat velké množství informací, které umožní odhalit a následně opravit spoustu chyb, které je možné nalézt až při práci se skutečnými uživateli. Avšak v případě již hotového rozhraní je pravděpodobné, že každá oprava chyby či změna je mnohem dražší, než kdyby byla odhalena dříve, například už ve fázi jeho návrhu. Právě proto je vhodné provádět testy již na graficky reprezentovaných prototypech a testovat neustále během celého vývoje. Ideální situace je tedy taková, že během vývoje aplikace pracuje vedle týmu, který vyvíjí logické jádro aplikace, ještě druhý tým, který se věnuje pouze výzkumu a návrhu grafického rozhraní. To nemusí být zpočátku ani implementováno do funkčního kódu, ale reprezentováno například pomocí papírových mockup prototypů nebo různých náčrtků.

Během takto provedeného testování vzniká obrovské množství dat, které musí výzkumný tým vyhodnotit. Velká část těchto dat je obvykle velmi podobně strukturovaná, ale existují i data, která se v jednotlivých projektech či dokonce i iteracích velmi liší, či existují právě v dané iteraci testů. Je tedy vhodné využívat pro testování a manipulaci s daty, takové prostředí, které umožňuje správu velkého objemu rozmanitých dokumentů či jiných datových souborů.

Cílem projektu je vytvořit program, který by usnadnil zpracování velkého objemu dokumentů vzniklých během uživatelského testování v rámci výzkumného týmu. Aplikace, která bude výstupem celého projektu, bude umožňovat skupině výzkumníků nejen vyhodnocovat, ale i vytvářet a upravovat různé typy dat typické pro testování s uživateli. Při vytváření aplikace bude kladen důraz na její použitelnost v kolaborativním prostředí, kdy bude umožněno, aby více uživatelů mohlo spravovat stejná data.

Kapitola 2

Popis problému, specifikace cíle

Tato kapitola popisuje řešenou situaci, smysl a cíl této diplomové práce. Kapitola je doplněna o průzkum stávajících řešení, včetně souhrnu používaných aplikací a aktuální verze aplikace IVE.

2.1 Popis problému

Při návrhu uživatelského rozhraní, ať už softwarového či hardwarového, je důležitou součástí tohoto procesu i analýza uživatelských potřeb a vůbec myšlení uživatelů. Tento výzkum je prováděn pomocí interview a testů s typickými představiteli skupin uživatelů, které pak budou s navrhovaným nástrojem reálně pracovat. Proces analýzy je časově náročná práce, během které výzkumní pracovníci získají obrovské množství různorodých dat, které pak musí dále zpracovávat a vyhodnocovat. Formát a význam těchto dat se mnohdy může lišit nejen pro každý projekt, ale dokonce i pro každé kolo testování v rámci jednoho projektu. Právě díky různorodosti dat může být zpracování a následné vyhodnocení získaných údajů velmi složité a vyčerpávající.

Analýza a testování uživatelského rozhraní je časově velmi náročné. V případě testování velkého projektu nebo nutnosti použití více nástrojů pro kvalitní sběr všech potřebných dat není vhodné, aby celou analýzu prováděl jediný pracovník. Práce celého týmu je v tomto případě mnohem vhodnější. Obrovský přínos práce v kolaborativním prostředí je, že každý člen týmu může na danou situaci pohlížet jinak nebo přijít s jinou myšlenkou. Právě možnost různých názorů a následná diskuze je pro celý proces návrhu rozhraní velmi důležitá. Další nesporný bonus, který čerpá z kooperace výzkumníků je možnost rozdělení rolí při testování a poté hlavně možnost rozdělení získaných dat a jejich zpracování. To však vyžaduje vytvoření určitého komunikačního protokolu mezi jednotlivými členy týmu, aby byly veškeré informace mezi nimi sdílené.

Samotný sběr dat je velmi komplikovaný proces, pro který je použito mnoho druhů zdrojů dat. Tyto zdroje se mohou lišit nejen rozdělením na základní struktury dat a jejich smyslem, ale také původem. Pro každé prostředí, ve kterém je daný test prováděn, ale i pro každou iteraci, může být použita jiná technika nebo jiný software pro sběr a záznam informací. Důsledkem toho může být v rámci jediného projektu zavedeno několik formátů stejných dat. Tato skutečnost komplikuje jejich budoucí zpracování a vyhodnocení.

2.2 Popis procesů práce výzkumného pracovníka

V této sekci jsou popsány základní procesy práce výzkumných pracovníků v oblasti testování použitelnosti rozhraní s uživatelem. Znalost těchto procesů je nutná pro správné pochopení požadavků na funkcionality výsledné aplikace. Podrobnější rozbor procesů lze nalézt v dizertační práci Ing. Ivo Malého [5].

2.2.1 User centered design

Moderní testování použitelnosti vychází z myšlenky vytváření uživatelského rozhraní pro potřeby uživatele. Tato filozofie bývá označována jako User Center Design.

Hlavní myšlenkou User Center Design (dále UCD) je umístění uživatele do středu aplikace. Zaměřit návrh celého rozhraní, se kterým uživatel manipuluje, tak aby se mohl v aplikaci snadno orientovat a maximálně využít její funkce. UCD se zabývá hlavně nalezením odpovědí na otázky typu: „Jaká je cílová skupina uživatelů rozhraní? Jak si uživatel představuje, že rozhraní funguje? Jaké funkce uživatel očekává?“ Výzkumem těchto otázek může UCD zodpovědět i další a pro majitele či výrobce aplikace důležitější otázky, zda je celý produkt opravdu použitelný a zda má pro uživatele skutečnou užitnou hodnotu.

2.2.2 Testování použitelnosti s uživateli

Testování použitelnosti je metoda pro zjištění použitelnosti určitého rozhraní na základě pozorování interakce uživatelů s rozhraním. Během samotného testu je participant z cílové skupiny uživatelů, pro které je rozhraní určeno, pozorován při plnění určité posloupnosti úkolů, které mu zadává výzkumný pracovník. U uživatelského testování nemusí být přítomen pouze jediný výzkumný pracovník, ale daného participanta (uživatele z cílové skupiny) může sledovat celý tým. Mezi hlavní pracovníky podílející se na testování patří moderátor, který kontroluje správný průběh testu, ale také se věnuje participantovi a dbá na jeho psychickou pohodu. Dalšími důležitými členy týmu jsou pozorovatelé (observers), kteří obvykle nezasahují do samotného průběhu testu a mají za úkol hlavně získávat a zaznamenávat informace z pozorování. Poslední skupinou, která bývá přítomna testům a která je pro plný účinek významu podle Kruga [4] nejdůležitější jsou takzvaní stakeholdeři. Tato skupina jsou pracovníci firmy, kteří se nějakým způsobem podílejí na vývoji aplikace. Jejich přítomnost je důležitá pro pochopení myšlenkových pochodů a potřeb participantů, tedy typických uživatelů aplikace, a aby sami viděli problémy uživatelského rozhraní. Je vhodné když jsou v této skupině přítomni i vedoucí pracovníci zainteresovaných oddělení, aby dokázali vyhodnotit, že změny, navržené na základě výsledků testování, je vhodné přijmout a investovat čas vývojového týmu do jejich provedení.

Testování použitelnosti není jednorázová metoda pro návrh aplikace, ale získává sílu z mnoha opakování. Každé kolo testů se skládá z několika mezikroků, od plánování a přípravy, přes samotné testování až k analýze a prezentování výsledků. V prvním kole návrhu uživatelského rozhraní jsou obvykle prováděna výzkumná interview, někdy je možné toto kolo vynechat a přejít rovnou k dalším. Už v počáteční fázi náčrtků a skic je však vhodné začít s prvním testováním. Toto testování pomůže odhalit obecné chyby v rozvržení, naznačit očekávání uživatelů a pomoci s upřesněním jednotlivých funkcionalit. Velká výhoda tohoto

testování je objevení problémů rozhraní a získání poznatků o cílové skupině uživatelů tak brzy jak je to jen možné. Dalším krokem je tvorba a testování wireframů či prototypů rozhraní, ať už se jedná o papírové nebo z části implementované rozhraní. V této části testování stále nemusí být testované rozhraní plně funkční a dokonce nemusí být ani plně graficky zpracované. Cílem tohoto kola testů je opět zjistit nejen obecné, ale i konkrétnější problémy daného rozhraní. Poslední iterace jsou už testování funkčních částí nebo dokonce kompletního uživatelského rozhraní. Tato část výzkumu je časově nejnáročnější, ale pokud jsou všechny obecné problémy odhaleny v předchozích iteracích, nemělo by zde docházet k větším zásahům do rozvržení rozhraní. Každou z těchto iterací lze opakovat vícekrát a v případě odhalení problému, který se nepodařilo odchytit včas, je možné vrátit se k předchozí fázi návrhu rozhraní.

Stejně jako v každém jiném testování platí, že čím dříve a důkladněji se testování rozhraní provede, tím více času a peněz ve výsledku ušetří. Tuto myšlenku se postupně daří šířit i v českých firmách a následkem toho význam uživatelského testování v posledních letech relativně rychle roste.

2.2.2.1 Plánování a příprava testů

Během plánování testu jsou vymezeny funkce rozhraní, na které budou jednotlivé úkoly testů cílit. Je zde rozhodnuto, jaká metoda bude pro testování použitá včetně výběru typů dat, která budou v průběhu testu snímána. Posledním krokem je určení počtu participantů v dané iteraci a charakterizování skupiny, ze které budou participanti vybíráni.

Po naplánování testu následuje pozvání vybraných participantů a příprava prostředí pro provedení testu. Pro vybrání vhodných participantů existuje mnoho metodik, které jsou voleny v závislosti na situaci a cílové skupině uživatelů, zvolené k účasti na testu. Samotná příprava testu se skládá z vytvoření podrobně rozepsaných úkolů testu, z přípravy testovacího prostředí. To znamená připravit místnost, kde bude testování probíhat a to včetně všech zařízení i rozhraní. Zajistit správnou funkci uživatelského rozhraní, které bude testováno a zkontrolovat také prostředí ve kterém je uživatelské rozhraní integrováno. Tyto kontroly jsou důležité pro odstranění nežádoucích vyrušení a vlivů, které mohou silně zkreslit průběh testu. Poslední částí přípravy je kontrola všech zařízení použitých pro monitorování interakcí uživatele s rozhraním.

2.2.2.2 Průběh samotného testu

Samotný test je provádět dle předem připraveného seznamu úkolů. Moderátor by měl dbát na to, aby bylo možno získat z každého testu maximum informací, proto může participanta korigovat v určených mantinelech chování. To je myšleno tak, aby udržel participantovu pozornost na plnění úkolů testu, nikoliv tak aby uživatele naváděl či mu dokonce radil jak v testu dále postupovat. Vlastnímu testování ještě předchází pre-test dotazník či interview pro zjištění dodatečných informací o participantovi. Průběh testu je monitorován zvolenými zdroji, důležitý zdroj informací je také moderátor testu, který je participantovi nejblíže a může tedy získat mnohem víc vjemů, které závisí spíše na citu moderátora, než na vyhodnocení dat například z videa. Po skončení testu je obvyklé, že moderátor provede ještě post-test interview (či dotazník) pro zjištění dodatečných informací k testu a dozvědět se podrobnosti ohledně určitých zajímavých částí.

2.2.2.3 Analýza a prezentace výsledků testů

Po dokončení celého kola testů následuje analýza výsledků. Jedná se o proces, kdy se ze získaných informací tvoří teorie o problémech v uživatelském rozhraní. Samotná analýza může probíhat více způsoby. Jedna z možností je hledat problémy rozhraní na základě informací z testů. Druhá možnost je využívaná v případě, že už jsou problémy zjištěny a je potřeba je potvrdit. Tato možnost funguje tak, že se ověřuje, zda došlo ke stejným či podobným problémům i v dalších testech. V praxi se používají obě metody současně, kdy jsou druhou metodou potvrzovány nálezy první metody.

Poslední fáze jedné iterace je prezentace výsledků testu. Výsledný dokument by měl obsahovat popis testu, přehled nalezených problémů, podrobný rozbor vybraných problémů a doporučení pro jejich řešení. Pro větší motivaci celého týmu a také pro zabránění případného odstranění, je vhodné do prezentace výsledků zařadit i pozitivní nálezy.

2.2.3 Prostředí testování

Důležitá součást každého testu je prostředí, ve kterém se test provádí. Prostředí by mělo být maximálně věrné reálné situaci, ve které bude uživatelské rozhraní v praxi používáno. Testování v reálném prostředí je však mnohdy komplikované a objem získaných informací nemusí být kvůli problémům s monitorováním akcí uživatele větší než testování v laboratoři. Určitě je vhodné v pozdější fázi vývoje provádět část testů i v reálném prostředí.

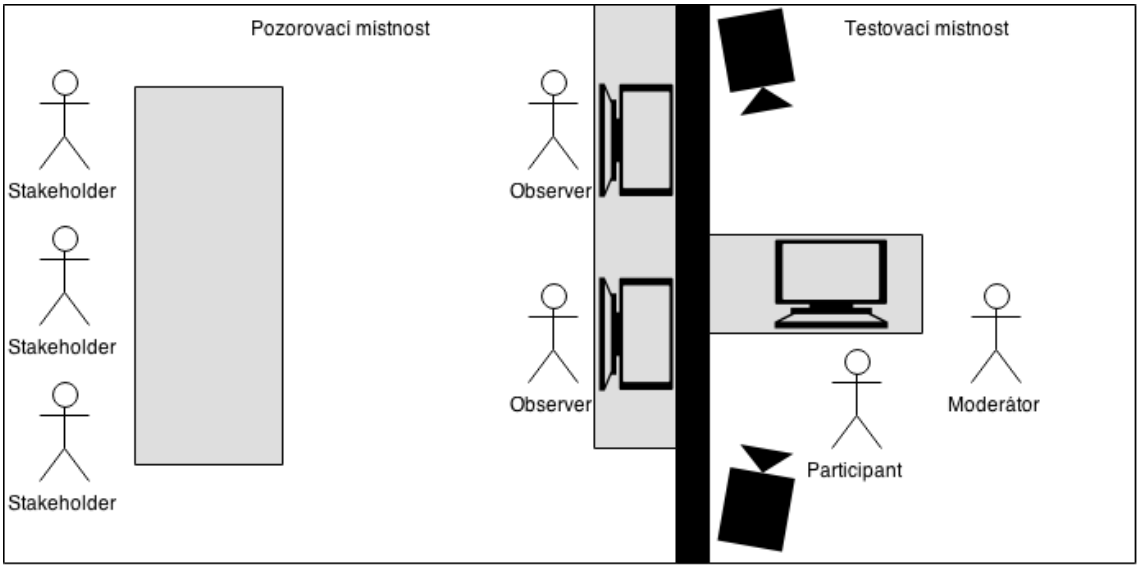
2.2.3.1 Testování v laboratoři

Testování v laboratoři je nejjednodušší způsob testování z pohledu výzkumného týmu. Prostředí je pro ně známé a připravit si ho mohou dopředu tak, že sníží možné technické problémy, které mohou vzniknout během testování, na minimum. Tento způsob testování je vhodný zejména kvůli možnosti přizvání stakeholderů k účasti na testování.

Laboratoř pro testování se obvykle skládá ze dvou oddělených místností (viz Obrázek 2.1). Jedna místnost slouží k testování a je zde instalováno uživatelské rozhraní, které je předmětem testu. Tato místnost by měla být neutrální tak, aby se zde participant cítil příjemně, ale aby byl motivován k maximálnímu soustředění na plnění úkolů testu. V této místnosti jsou instalována zařízení pro monitorování akcí participanta. Druhá místnost je určena pro observery, přenos dění z testovací místnosti do této je zprostředkován pomocí videa a zvukového záznamu, dříve se využívalo i polopropustné zrcadlo, avšak to vyžaduje, aby byly použity sousedící místnosti a odhlučnění místností je pak komplikovanější. Je důležité, aby byly tyto místnost navzájem zvukotěsné, aby participanta v průběhu práce nerušily nežádoucí vjemy z okolí. Pokud jsou testu přítomni stakeholdeři, pak jsou buď v místnosti observerů, ale vzhledem k pravděpodobné debatě, mnohdy i ostřejší a hlasitější výměně názorů, je vhodné vyčlenit pro ně vlastní místnost.

2.2.3.2 Vzdálené testování

Vzdálené testování umožňuje provádět uživatelský výzkum v jejich vlastním prostředí. To je zaručeno díky vzájemné spolupráci při domluvě způsobu a příslušného nastavení přenosu



Obrázek 2.1: Laboratoř pro testování použitelnosti se dvěma místnostmi

potřebných záznamů z testování uživatelského rozhraní. Toto testování je vhodné v případě, že se participant nemůže dostavit do místa laboratoře a není možné tohoto uživatele nijak nahradit. Kromě menší časové náročnosti je pozitivem, že participant pracuje v prostředí, které zná a se kterým umí pracovat. Nemusí se cítit tolik pod tlakem, ve stresu nebo naopak je více ve stresu, což simuluje reálné prostředí. Negativum tohoto testování může být narušení soukromí, kdy je spolu s participantem snímáno i dění okolo něj, minimální záznam "řeči těla"participanta a hlavně složitější nastavení celého systému. Komplikovanější může být i udržet pozornost participanta na test, aniž by ho rušily okolní vjemy.

Vzhledem k větší pravděpodobnosti problémů během přenosu testování není toto testování tak vhodné pro pozvání stakeholderů, ale v případě zaručení dobrých podmínek by neměl být problém zprostředkovat přenos i pro ně.

2.2.3.3 Testování v reálném prostředí

Testování v reálném prostředí se nejvíce přibližuje opravdovému nasazení daného rozhraní. Problémem však může být instalace a obsluha všech zařízení, která jsou potřeba pro snímání uživatele. Toto prostředí přináší mnoho nepředvídatelných událostí, které v neměnných laboratorních podmínkách nemohou nastat. Jedná se například o použití mobilních aplikací za chůze, o odlesky slunce na obrazovce zařízení a podobné. Testování v reálném prostředí přináší i prvek náhody, což může mít na výsledky jak pozitivní tak i negativní vliv. Tento způsob testování je však obtížněji monitorovatelný a je potřeba informace, které výzkumníci získají během testování, ověřit ještě znovu v řízených podmínkách.

2.2.3.4 Shrnutí

Pro každé prostředí jsou specifické metody sběru dat. Obecně lze říci, že jsou tyto metody poměrně shodné a liší se jen v konkrétních detailech a použití rozdílné techniky. V rámci

každého prostředí lze sbírat jen část potřebných dat, ať už je to kvůli nereálným podmínkám, rušivým vlivům nebo komplikovanému záznamu uživatelských akcí. Proto je vhodné tato prostředí kombinovat pro získání maximálního množství informací.

Díky rozdílným přístupům a metodám ke sběru informací v různých prostředích existuje i mnoho rozdílných typů a podob získaných dat. Pro každé prostředí může být využívána jiná technika či jiný software, který má specifický výstupní formát dat. Právě to může být komplikací v jejich pozdějším zpracování a vyhodnocení výzkumnými pracovníky.

Rozdílná data mohou vznikat i díky různým potřebám výzkumu uživatelského rozhraní a dle jeho zaměření. Různé formáty mohou tedy záviset na základě aktuálních požadavků výzkumných pracovníků, kteří se mohou v rámci jedné iteraci zaměřit pouze na specifickou část uživatelského rozhraní a vytvořit speciální typ dat, která umožní popsat získané informace.

2.2.4 Základní zdroje dat

Díky různorodosti testů i jejich cílů jsou výstupem jednotlivých testů rozmanité typy dat. Obecně lze tato data rozdělit na textové zaznamenání akcí či textové poznámky a poté na různé nahrávky, ať už zvukové, video záznam nebo fotografie a snímky.

2.2.4.1 Záznam aktivit participanta a poznámky

Záznam aktivit participanta je ručně vytvářený soubor textových poznámek závislých na čase. Tyto poznámky vytváří během testu jeden, ale je obvyklé, že spolupracuje více výzkumníků. Tento způsob záznamu je velmi důležitý pro označování zajímavých částí testu, protože je čitelnější než video, ale navíc může obsahovat i dodatečné informace, které nejsou vidět z videa nebo slyšet ze zvukového záznamu.

Existuje několik variant poznámek. Jeden typ je vytvářen observery přímo v průběhu testu. Tyto poznámky jsou často jen v podobě krátkých zpráv o zajímavých místech v testu. Další druh poznámek může být získáván přímo od moderátora testu, který je v místnosti spolu s participantem a může vnímat mnohem více vjemů, než je vidět z video přenosu. Bohužel tyto poznámky může zaznamenávat až po ukončení testu, což může zkreslit jeho závěry, ale zároveň díky tomu může mít i utříděné myšlenky a ukázat tak na problémy, které nejsou při pohledu z videa vyhodnoceny jako důležité.

2.2.4.2 Snímky rozhraní či fotografie

Snímky rozhraní a fotografie nebývají součástí každého testování. Snímky mají význam jen při určitých testech a to vzhledem k možnosti použít namísto statických fotografií kompletní video záznam, který zaznamenává i kontext jednotlivých akcí participanta.

2.2.4.3 Video a audio záznam

Video a zvukový záznam je nejlepším typem záznamu průběhu testu. Je velmi důležitý pro svou znovupoužitelnost a umožňuje tak získávat nové informace opakovaně. Záznam videa a audia je plně nebo alespoň z části automatizovaný a není tedy tak náročný na obsluhu jako například záznam aktivit participanta.

Zvukový záznam je zaznamenáván pomocí mikrofonů umístěných blízko participanta a jeho úkol je zprostředkovat zvuky z okolí uživatele a v případě takzvané metody „přemýšlení nahlas“ i záznam jeho myšlenek.

Záznam videa může mít několik cílů. První je záznam uživatelského rozhraní, například obrazovky monitoru během testování. Z tohoto videa lze vyčíst, co uživatel viděl, s čím jak pracoval a na co klikal myší. Další možnost zaznamenávat obličej participanta, což může být velmi důležitá součást každého testu, protože z tohoto záznamu lze vyčíst emoce participanta. Poslední častý záznam je snímání celé scény a uživatele včetně rozhraní. Tento zdroj videa má za úkol snímat celou situaci a propojit tak do jednoho kontextu záznam obrazovky i uživatele.

2.3 Rešerše existujících řešení

Tato podkapitola je věnována průzkumu existujících řešení, která jsou v praxi výzkumníky používána. Mezi nejznámější patří systém Morae a Noldus, do rešerše byla kvůli své návaznosti zařazena i aplikace IVE.

2.3.1 Morae

Jako jeden z vedoucích softwarových nástrojů pro podporu testování použitelnosti na trhu je Morae. Pro své uživatele nabízí komplexní podporu při testování, analýze a vyhodnocování dat. Morae se skládá z více částí, které spolu souvisí, ale pracují v podobě oddělených programů. Tyto programy slouží jako podpora jak během samotného testování (Morae Recorder, Morae Observer), tak i poté během vyhodnocování sebraných dat (Morae Manager). Systém Morae je vyvíjený firmou TechSmith, cena tohoto produktu je 1995 USD bez DPH, přes svou cenu je však Morae nedocenitelný pomocník při testování softwarového rozhraní s uživatelem.

Systém Morae umožňuje i vytváření vlastních modulů, které více odpovídají aktuálním požadavkům uživatelů. Díky této funkci je spektrum možností použití celého systému velmi široké. V případě jiné funkcionality než poskytuje základní verze systému Morae, je tedy možné použít jiný vhodný nástroj, který by se stejnými daty uměl pracovat, nebo vytvořit vlastní plugin, který bude poskytovat stejnou funkcionalitu také. Morae pro import či export dat do jiných aplikací používá nástroj, který umožňuje data otevírat i v různých kancelářských programech a v nich poté provádět úpravy těchto souborů, tak aby byla data možno importovat do jiné aplikace, ale tento proces může být velmi zdlouhavý, nepohodlný a únavný. Proto je vhodné implementovat vlastní funkčnost pomocí nového pluginu.

Morae nemá podporu pro sdílení surových dat, ale oproti tomu podporuje tvorbu a sdílení výsledků díky aplikaci Morae Manager. Tento nedostatek znamená použití jiného způsobu šíření dat mezi jednotlivými výzkumnými pracovníky. Tato výměna dat je ovšem velmi nepohodlná a může vést k vytvoření několika kopií stejných dat jen s malými, ale přesto významnými odlišnostmi. Tyto drobnosti mohou ve výsledku znamenat velkou komplikaci při použití aplikace pro kolaborativní práci nad stejnými daty.

Podrobnější informace jsou na stránkách projektu [17].

2.3.1.1 Morae Recorder

Morae Recorder slouží ke snímání akcí uživatele. Obsahuje podporu nejen pro snímání uživatele pomocí více kamer, záznam zvuku snímaný mikrofony, ale například i snímání obrazovky a tedy přímý záznam aktivit participanta.

Morae Recorder neslouží jen ke snímání akcí, ale díky tomu, že je spuštěný na stroji, který je použit jako terminál pro testování, obsahuje i funkce pro zobrazení jednotlivých úkolů v testu.

Morae Recorder umožňuje zaznamenávaná data přímo ukládat nebo slouží jako zdroj přenosu pro Morase Observer.

2.3.1.2 Morae Observer

Morae Observer je určitým způsobem napojení na stream, jehož zdrojem je Morae Recorder. Umožňuje zobrazovat přenosy jak videa z externích kamer, tak snímaná z obrazovky uživatele a to včetně zvuku.

Morae Observer je často používán pro zprostředkování přenosu pro observery a stakeholdery. Umožňuje vytvářet poznámky/logy a označovat zajímavé momenty v průběhu testování.

K Morae Recorderu může být připojeno více Observerů, což zjednodušuje práci výzkumného týmu v případě nutnosti vzdáleného přístupu.

2.3.1.3 Morae Manager

Morae Manager slouží k zobrazování a vyhodnocování dat získaných z nahrávek logů atd. Umožňuje vygenerovat grafy či vytvářet záznamy a sdílet informace se stakeholdery.

Morae Manager slouží hlavně v procesech navazujících po sběru dat. Zabývá se zprostředkováním nástrojů vhodných pro vyhodnocení dat z testů a vhodnou prezentaci výsledků. Díky add-in umožňuje vytvářet reporty pro MS Word, stříhat videa či vybraná data a výsledky sdílet.

Morae Manager slouží i k vytvoření nastavení testů. Tato nastavení jsou pak použita ve výše zmíněném programu Morae Recorder.

2.3.2 Observer XT

Systém Observer XT, vyvíjený firmou Noldus, je komplexní nástroj pro podporu práci s daty z uživatelského testování. Tento nástroj sám o sobě poskytuje prostředí, které podporuje celý proces testování s uživateli od nastavení testu, přes sběr dat (viz Obrázek 2.2) až po jejich analýzu. Celý tento systém může být obohacen o další moduly pro podporu konkrétních požadavků.

Základní Observer XT obsahuje pět modulů, které slouží k záznamu a vyhodnocení informací získávaných během testu s uživateli. Dále obsahuje základní moduly pro záznam a tvorbu poznámek o akcích uživatele.

- Advanced Analysis Module
  - Tento modul umožňuje provádět dva typy analýzy: analýzu spolehlivosti a sekvenční analýzu.
- Media Module
  - Modul navržený pro přehrávání jednoho až dvou video nebo audio souborů. Umožňuje také vytvářet klipy ze zajímavých momentů videa.
- Multiple Media Module
  - Rozšíření Media modulu pro až čtyři mediální soubory najednou. Pro větší počet videí je nutné tento modul upravit.
- External Data Module
  - Tento modul zajišťuje import externích souborů do systému. Import se týká pouze dat ve specifickém formátu EDF (European Data Format). Data získaná pomocí jiných aplikací tedy lze po úpravě importovat a analyzovat je v Observer XT.
- Vykreslení grafů
  - Software Development Kit
- Vykreslení souřadnic do mapy
  - Nástroj pro tvorbu či úpravu vlastních komponent, které budou reagovat na aktuální požadavky výzkumných pracovníků.

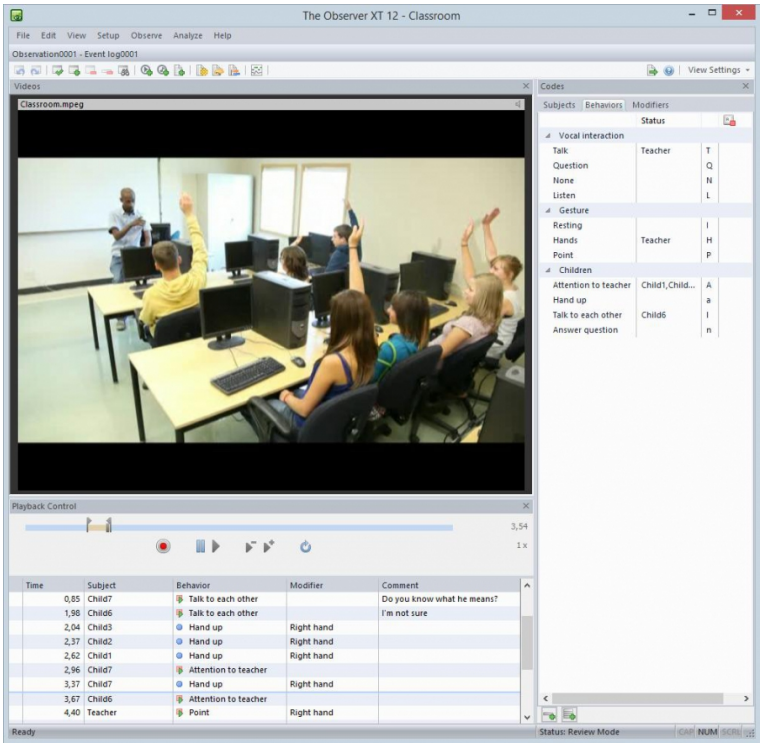
Tento nástroj umožňuje i určitou formu sdílení dat. V základu podporuje sdílení různých nastavení testů či celých testovacích profilů mezi uživateli. Dále umožňuje sdílet získané informace z testů. Toto sdílení dat může být komplikované vzhledem k případnému vytvoření více verzí dokumentů, pokud daný dokument upravuje mnoho observerů.

Díky modulu Software Developmnet Kit lze vytvářet vlastní moduly, které jsou kompatibilní se základní verzí systému, takže mohou výzkumní pracovníci reagovat na aktuální trendy v konkrétním projektu. Tato funkce ze systému Noldus dělá velmi silný nástroj, který je možný upravit přesně na míru uživatelům.

Podrobnější informace lze nalézt na stránkách projektu [20].

2.3.2.1 uASQ

Dotazníkový nástroj pro sběr zpětné vazby od participanta testu. Pracuje takovým způsobem, že během testu předkládá participantovi otázky a jeho odpovědi poté propojí s časovou osou jiných zdrojů, například videa. Poskytuje celkem tři typy otázek: otevřené, vícenásobný výběr nebo Likertovu škálu. Observer XT po skončení testu dovoluje snadné vyhodnocení získaných informací z odpovědí na otázky výběru a Likertovy škály. Pro analýzu otevřených otázek je k dispozici pokročilé vyhledávání.



Obrázek 2.2: Náhled programu Observer XT, konkrétně nástroje pro záznam uživatelských aktivit, zdroj: www.noldus.com

2.3.2.2 uLog

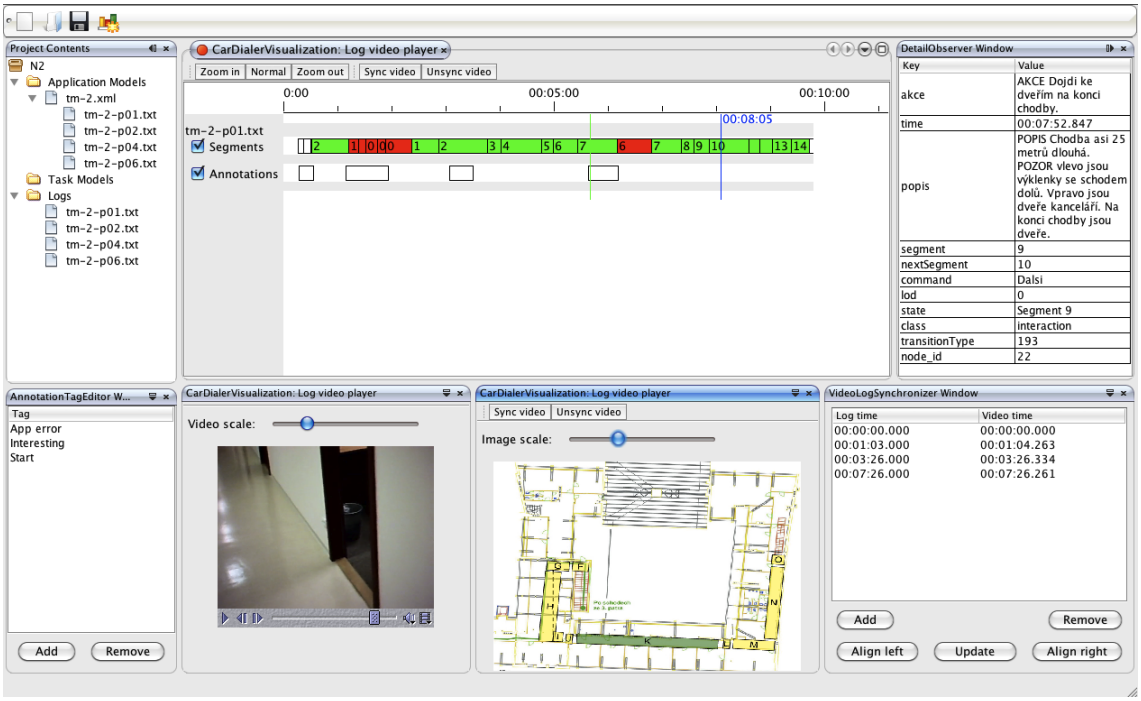
Tento modul umožňuje zaznamenávat interakci uživatele se systémem automatizovaně. Díky uLog je možné zvýšit konzistenci informací získaných během testování s uživatelem. Informace získané automatickým záznamem jsou velmi detailní, takže umožňují vytvářet přesné analýzy pohybu uživatele v daném systému. Pomocí základního nástroje Observer XT lze tento modul ovládat, vytvářet vizualizace nad získanými daty a analyzovat je. Pro vybrání konkrétních dat lze použít funkci pokročilého filtru ze základní aplikace.

2.3.2.3 Pocket Observer

Jedná se o klon aplikace určený pro použití v místech, kde nelze využít Observer XT. Existují verze pro smartphony, tablety, ale i pro jiné kapesní počítače. Tato aplikace je určena pouze pro sběr dat. Použití tedy úzce souvisí s Observer XT, kdy se celý test vytvoří v tomto programu, nahraje do Pocket Observeru, kde se zaznamenávají informace z jednotlivých testů. Po ukončení testů se data importují do Observer XT, zde se s nimi pracuje stejně jako s daty, které z něho pochází.

2.3.3 IVE

IVE je aplikace vyvíjená na Fakultě elektrotechnické ČVUT v Praze. Její vývoj zajišťuje Ing. Ivo Malý, PhD. Slouží jako interaktivní vizualizační nástroj pro analýzu souborů dat uži-



Obrázek 2.3: Náhled nástroje IVE včetně několika vizualizačních nástrojů, autor: Ing. Ivo Malý, Ph.D.

vatelského testování. Umožňuje výzkumnému pracovníkovi zpracovávat data z rozmanitých testů. Obsahuje širokou škálu modulů pro práci s různými datovými sety a díky modulárnímu přístupu je možné vytvářet další vizualizační nástroje pro konkrétní účely. Pro náhled nástroje IVE, včetně několika vizualizačních nástrojů, je přiložen Obrázek 2.3.

Jeho nevýhodou je, že neobsahuje podporu pro kolaborativní práci s daty. Data mohou být sdílena pouze pomocí externích aplikací. Taková výměna dat je ovšem velmi nepohodlná a může vést k vytvoření několika kopií stejných dat jen s malými, ale přesto významnými odlišnostmi. Tyto drobnosti mohou ve výsledku znamenat velkou komplikaci při použití aplikace pro kolaborativní práci nad stejnými daty.

Podrobnější informace jsou na stránkách projektu [15].

2.3.3.1 Funkce

Stávající řešení pokrývá velký rozsah funkcí pomocí na míru vytvořených modulů. Tyto moduly vznikají ve chvíli, kdy je objevena potřeba nového druhu dat nebo v případě jiného zobrazení stávajících dat.

- Přehrávání videa
  - Mezi nejdůležitější moduly je možnost přehrávání videa. Při sběru dat je často natáčen video záznam průběhu testování. Jedná se o záběr uživatele včetně okolí, ale i o variantu snímání obrazovky počítače či jiného zařízení, na kterém testování probíhá.

  - Takto získaná videa je vhodné spojovat s dalšími daty, jako jsou jednotlivé zprávy z logu nebo anotace videa.

- Anotování videa

  - Jednotlivá testování mohou trvat i více jak hodinu a obsahovat jen pár krátkých úseků, která jsou pro výzkumníka zajímavá. Tato místa je vhodné označit pro další zpracování tak, aby bylo možné je snadno a rychle dohledat bez složitého přetáčení celého videa.
  - Také je vhodné popisovat jednotlivé pasáže, aby tam bylo z anotací viditelné, co se v jaké části videa dělo.

- Zobrazení logů

  - Logy jsou základním typem dat sebraných během testování. Jedná se o seznamy událostí seřazených dle času. Tyto zprávy popisují, co v určitý čas uživatel dělal, co říkal. Na rozdíl od prostého záznamu videa či zvuku zde však může být silně znát i dojem zapisovatele z testování, proto nelze textové logy nahradit pohodlnějším záznamem.
  - Logy jsou většinou uloženy pomocí prostých textových souborů popřípadě je velmi snadné je do této podoby exportovat.

- Synchronizace časové osy

  - Velké množství dat je vázáno na konkrétní čas, v případě více dat souvisejících s průběhem jednoho testování je vhodné tyto data sloučit k jedné časové ose. V případě, že se spustí přehrávání videa, spustí se zároveň i časová osa a s ní i další časově závislé informace jako například zprávy logu.
  - Modul časové osy by měl být implementován do jádra aplikace, aby byl přístupný i z dalších poté vytvořených modulů. Osa by měla být maximálně univerzální, protože je velice pravděpodobné, že časová osa bude nějakým způsobem přítomna během každého testování.

- Vykreslení grafů

  - V případě statistických dat je vhodné pro tyto data vytvořit grafický výstup. Příklad může být v MS Excelu, který umožňuje velmi dobře vytvářet grafy pro různé vstupy. Opět by měl být tento modul vytvořen natolik univerzálně, aby bylo možné zvolit různé zdroje dat pro jednotlivé osy.

- Vykreslení souřadnic do mapy

  - Při testování v terénu, například při testování různých navigací apod. je vhodné snímat i polohu participanta dle souřadnic GPS. Tato data jsou velmi dobře čitelná při zobrazení na mapový podklad. Vzhledem k tomu, že je možné GPS souřadnice ukládat i s časovým údajem, může být v tomto případě data zakreslena i v podobě trasy.

2.4 Specifikace cíle

Cílem této diplomové práce je vytvoření aplikace pro platformu PC, která bude umožňovat výzkumným pracovníkům vytvářet a dále spravovat data získávaná z testování uživatelského rozhraní. Také poskytne nástroje pro jejich pozdější vyhodnocování. Umožní kolaborativní práci více uživatelů a bude poskytovat dostatek nástrojů pro podporu funkcionality CRUD pro práci s projekty, uživateli/participanty, úkoly a dalšími typickými entitami používanými při testování použitelnosti. Součástí této práce bude i vytvoření API/SPI pro tvorbu nových vizualizačních a editačních pluginů a několik vzorových vizualizačních pluginů. Při tvorbě bude jako předloha sloužit aktuální verze nástroje IVE tool.

Hlavním smyslem celého projektu je umožnit výzkumným pracovníkům v oblasti UX spravovat a zpracovávat velké objemy dat, které jsou typické pro uživatelský výzkum. Proto je potřeba navrhnout úložiště, jehož struktura bude umožňovat s těmito obsáhlými daty pracovat. Vzhledem k aktuálnímu systému dat v IVE a porovnání s dalšími nástroji, lze předpokládat textový formát dat získávaných z testování s uživateli. Pro možnost pokročilejších filtrů a vyhledávání je vhodné zvolit takové úložiště, které bude umět s těmito dokumenty pracovat. Toto úložiště by mělo být schopné reagovat i na pravděpodobné změny struktury dat v závislosti na potřebě popsání různých typů informací.

Vzhledem ke skutečnosti, že na uživatelském výzkumu se obvykle nepodílí jednotlivci, ale celé týmy pracovníků, vzniká potřeba pro distribuci dat mezi uživateli aplikace. Návrh systému by tedy měl pokrýt i potřebu synchronizace projektů s vybranými výzkumnými pracovníky. Synchronizace dat by měla být realizována pro všechny účastníky daného projektu tak, aby měl každý z výzkumníků aktualizovaná data a mohl je tak dále analyzovat.

V době vzniku tohoto projektu nejsou známy veškeré možné moduly a funkce, které budou výzkumní pracovníci pro svou činnost vyžadovat. Proto by celá aplikace měla být navržena tak, aby bylo možné ji v budoucnu obohacovat o další integrované funkčnosti a dodatečné moduly. Veškeré integrované části aplikace by měly být navrženy maximálně obecně tak, aby ve většině projektů nevyžadovaly další nutné zásahy a změny. Oproti tomu dodatečné moduly by měly umožňovat konkrétní funkce potřebné i pro několik málo testů.

Kapitola 3

Návrh řešení

Tento projekt vychází z první verze aplikace IVE, která je popsaná v předchozí kapitole. V současné verzi IVE je několik nedostatků, které se snaží návrh nového systému odstranit.

Původní aplikace pracuje se souborovým úložištěm, to komplikuje automatickou synchronizaci dat napříč klientskými aplikacemi, ale umožňuje snadnou přenositelnost dat z různých zařízení. Proto je vhodné navrhnout takový úložný systém, který sjednotí data do kompaktního úložiště a skrze něj pak umožní uživatelům jejich správu a distribuci na další klientské aplikace.

Soubory představují jak logy z uživatelského testu, tak videa, obrázky či popisky videa. Data budou získávána pomocí externích aplikací, ale část těchto dat bude umožněno vytvářet i pomocí této aplikace.

V rámci tohoto projektu bude vytvořen klient pro platformu PC a Mac. Paralelně s touto prací vzniká také mobilní logovací aplikace, jejímž autorem je Eduard Füzesséry. Systém by měl být tedy navrhnut tak, aby reagoval na potřeby dalších klientů.

3.1 Návrh systému

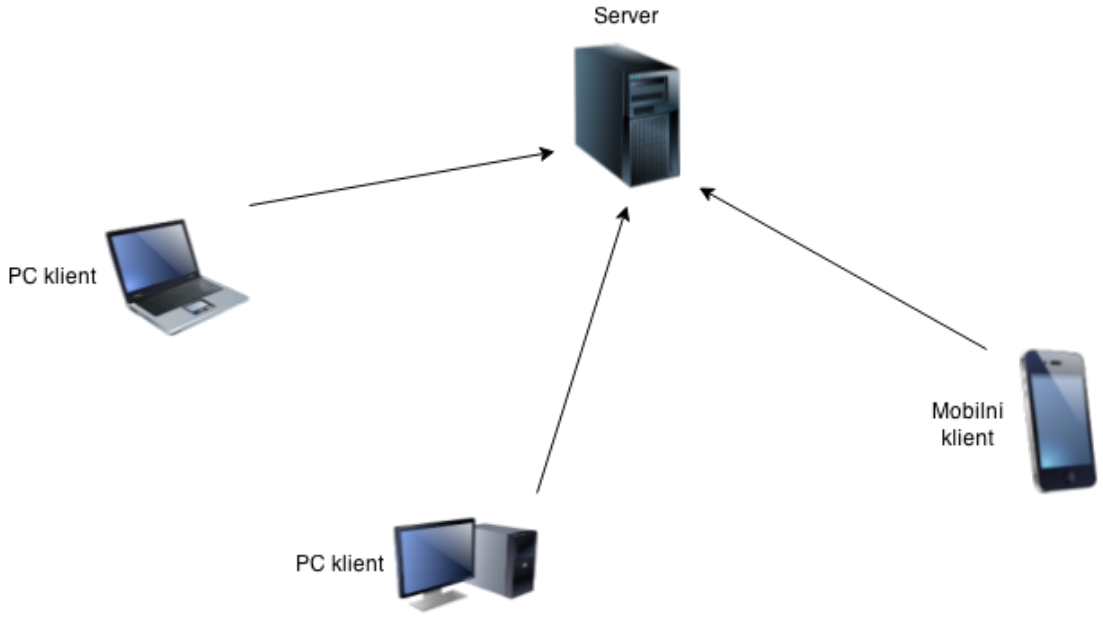
Celý systém se bude skládat ze dvou částí. Serverová část bude sloužit jako centrální bod systému. Klientské aplikace budou koncovému uživateli umožňovat správu dat, kvůli požadavku na funkčnost aplikace bez připojení k internetu bude použito i lokální úložiště pro přímou práci s daty.

Při návrhu obecného systému bylo zvoleno centrální úložiště dat namísto synchronizace pomocí klient-klient. Propojení klient-server redukuje množství problémů v synchronizaci dat. Výběr modelu centrálního repozitáře pro data byl ovlivněn i systémy pro správu kódu Git a SVN.

Klientských aplikací může být více druhů (viz Obrázek 3.1), v tomto projektu se však bude vypracovávat pouze desktop aplikace.

3.1.1 Serverová část

Serverová část by měla být co nejjednodušší a sloužit hlavně jako centrální bod celého systému. Server bude klíčový prvek pro veškerou distribuci dat podobně jako to je u verzovacího



Obrázek 3.1: Schéma celého systému s příklady možných klientských aplikací

systému Git. V ideálním případě bude pro server nasazeno takové úložiště, které už správu a distribuci dat řeší samo. V opačném případě bude nutno implementovat i rozhraní serveru, které umožní vkládání dat a jejich další správu.

Vzhledem k použití souborového systému a k předpokladu, že velká část výstupních souborů z uživatelského testování bude v textovém formátu, je vhodné celý systém orientovat na správu dokumentů. To umožňuje i předpokládanou snazší konverzi dat z předchozí verze IVE či z jiných programů.

Vzhledem k týmové práci na výzkumných projektech je vhodné implementovat do systému podporu verzování. Tato funkce by měla být podporována i serverem kvůli možnému rozšíření spektra aplikací, které budou s centrálou komunikovat.

3.1.2 Klientská část

Klientská část se bude skládat z vlastní databáze a klientské aplikace. Databázový systém by měl být kompatibilní se serverovou databází buď sám o sobě, respektive měl by umožňovat komunikaci se vzdáleným serverem sám nebo by měla být tato komunikace implementována v databázi. Tato aplikace bude modulární a bude vytvořena tak, aby se s minimálními úpravami jádra aplikace daly moduly vytvářet a integrovat.

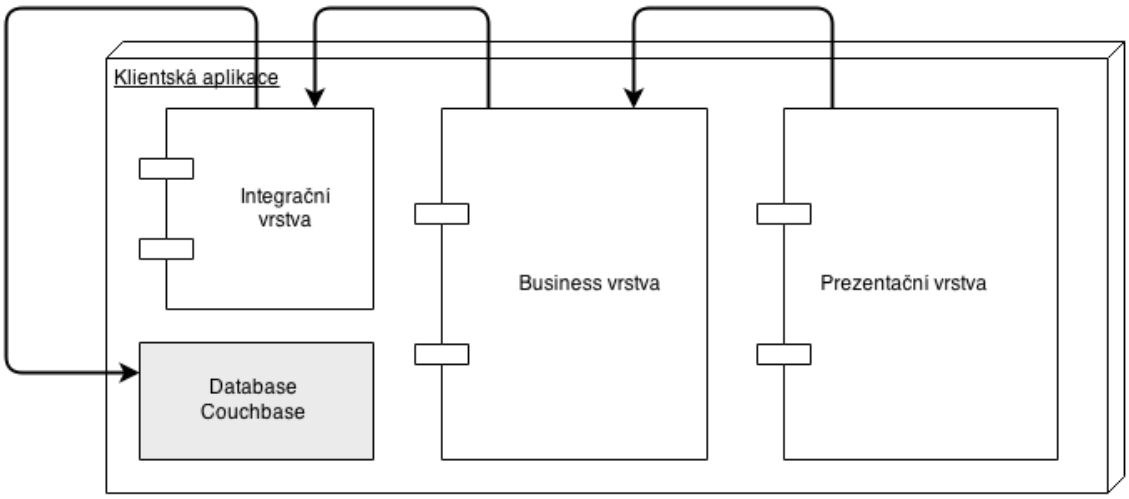
Aplikace bude rozdělena na několik vrstev, volání jejich jednotlivých funkcí je naznačeno na Obrázku 3.2.

- Integrační vrstva

  - Tato vrstva se bude starat o načtení dat z lokálního úložiště, ať už se bude jednat o data z databáze, či o samostatné soubory. Výstupem této vrstvy budou datové objekty.

- Business vrstva
  - Business vrstva bude obsahovat třídy useCase, pro komunikaci s integrační vrstvou a jinou práci s daty. Tato vrstva bude obsahovat i třídy modelu aplikace, jejichž instance budou datové objekty.

- Prezentační vrstva
  - Prezentační vrstva bude mít za úkol odchytávání událostí z UI. Prezentační vrstva bude komunikovat s business vrstvou a vzhledovými šablonami (v podobě java souborů popisujících uživatelské rozhraní).



Obrázek 3.2: Schéma rozvržení vrstev na moduly. Silné čáry představují posloupnost volání funkcionalit obsažených v daných modulech.

Díky takovémuto rozdělení na moduly by mělo být výrazně jednodušší změnit například typ databáze, nebo přenést klientskou aplikaci na server v podobě webové služby včetně webového grafického rozhraní.

3.2 Technologie

Tato část kapitoly je zaměřena na krátký popis možných technologií, které by mohly splnit požadavky na funkčnost v rámci aplikace. Sekce bude rozdělena na dvě hlavní části, první se bude věnovat možným technologiím, vhodným pro server. Poté bude následovat část, která se obdobně věnuje technologiím pro klientskou aplikaci.

3.2.1 Serverová část

Citelným nedostatkem ve stávajícím řešení je složitá distribuce dat. Pokud chtějí dva uživatelé pracovat se stejnými daty, musí si data sdílet pomocí externích nástrojů (dropbox, flashdisky), což není elegantní a určitě ne pohodlné řešení. Z tohoto důvodu bude od serveru vyžadována distribuce dat takovým způsobem, aby mohly jednotlivé klientské aplikace pracovat s daty a zároveň je sdílet ostatním právě přes server. Ideální je i vyřešit problém verzování souborů v případě, že dva uživatelé upravují stejná data současně.

3.2.1.1 Databáze

Pro úložiště dat je tedy vhodné využít možnost uložení v podobě souborů nebo s minimální konverzí dat. Vzhledem k potřebě verzování dat je vhodné použít úložiště, které už systém pro verzování poskytuje.

- MySQL
  - Jedna z nejpoužívanějších databází. Pro komunikaci s touto databází se používá jazyk SQL. Pro více informací viz webové stránky projektu [18].
  - Výhody
    - velmi rychlé a snadné dotazování
  - Nevýhody
    - problém uložení větších souborů
    - nutnost implementace vlastního serveru, který bude komunikovat s klienty
    - problémové zálohování dat
    - neřešeno verzování dat
    - přesná struktura dat
- CouchDB
  - NoSQL databáze, dokumentově orientovaný databázový systém. Zpracovává kolekce JSON dokumentů. Dokumenty v rámci kolekce nesdílejí společné schéma, přičemž nabízejí dotazovací kapacity prostřednictvím pohledů. Pohledy jsou definovány agregačními funkcemi a souběžně s tím filtrovány obdobně jako v MapReduce přístupu. Pro více informací viz webové stránky projektu [12].
  - Výhody
    - dokumentové úložiště
    - RESTfull HTTP API
    - podpora verzování dat
    - podpora synchornizace dat
    - není nutná přesně definovaná struktura dat
  - Nevýhody
    - pomalejší než MySQL

        * problém uložení větších souborů

- Couchbase
  - NoSQL databáze založená na CouchDB. Pro více informací viz webové stránky projektu [8].
  - Výhody
    * dokumentové úložiště
    * podpora verzování dat
    * podpora synchornizace dat
    * není nutná přesně definovaná struktura dat
    * profesionální SDK pro mnoho jazyků
  - Nevýhody
    * pomalejší než MySQL
    * problém uložení větších souborů

- Souborový systém + Git
  - Distribuovaný systém správy revizí. Mocná podpora pro vytváření větví. Pro více informací o systému Git viz [2].
  - Výhody
    * rychlá práce s verzemi
    * rozsáhlá podpora větví
    * distribuované
  - Nevýhody
    * není to databáze
    * nutnost implementace vlastního serveru jak pro práci s daty tak i práci s Gitem

3.2.1.2 Vyhodnocení

Kvůli velké rozmanitosti dat vzniklých během testování uživatelského rozhraní je vhodné použít volnější strukturu databáze, která umožní vkládat v podstatě libovolná data, bez zásahů do struktury databáze. Relační databáze MySQL není v tomto ohledu přizpůsobivá a je komplikované navrhnout takovou strukturu databáze, aby dobře popisovala data a zároveň dokázala přijímat i jiné, než dopředu nadefinované typy dat. Oproti tomu CouchDB a Couchbase přijímají tato data mnohem dynamičtěji a vlastně je jejich struktura tvořena daty v nich obsaženými, jediná podmínka je používat správný značkovací jazyk pro definici dat. Velkou výhodou CouchDB a Couchbase je, že jsou souborově orientované, což by mělo velmi usnadnit budoucí import dat ze stávajícího řešení IVE. Souborový systém v kombinaci s Gitem má proti výše zmíněným absolutní volnost v typech dat, nepodporuje ovšem jejich typování a chybí dotazovací jazyk na data v souborovém systému.

Synchronizace je pro tento projekt velmi důležitá, a pokud databázový systém nabízí funkci synchronizace dat napříč klientskými aplikacemi, je to velká výhoda. MySQL obsahuje funkce pro replikaci dat, ovšem data z testování mohou být upravována více uživateli a je tedy nutné aby synchronizační systém podporoval verzování. Couchbase a CouchDB podporují verzování dokumentů včetně možnosti náhledu do historie změn souboru. Tato funkce je velmi výhodná pro vyhledávání a opravu nechtěných změn v souboru. Nástroj Git je určený pro verzování zdrojového kódu, ale i jiných souborů v plain textu. Jeho velká síla oproti jiným systémům je rozsáhlá podpora větví, v tomto projektu však správa větví není potřeba. Synchronizace souborů, které jsou v Gitu spravovány, je prováděna pomocí několika základních příkazů, které by musely být ovládány skrze rozhraní aplikace.

Posledním důležitým parametrem bylo, zda je nutné vytvořit rozhraní pro komunikaci aplikací s úložištěm či nikoliv. MySQL databáze nemá v základní instalaci žádné rozhraní pro správu dat. Pro nejnovější verzi MySQL 5.7 existuje doplněk, který umožňuje vytvářet dotazy do databáze pomocí HTTP requestů. Bohužel je tato verze MySQL včetně doplňku pouze prototyp a je určený jen pro testování. CouchDB podporuje rozsáhlé HTTP REST-full API, které umožňuje správu souborů pro mnoho druhů klientských aplikací. Couchbase oproti tomu doporučuje používat svá SDK, která pracují nad HTTP API. Vývoj klientské aplikace pracující s Couchbase je díky těmto nástrojům velmi pohodlný a umožňuje vývojářům soustředit se pouze na konkrétní problémy v jejich softwaru, nikoliv řešit způsob připojení k databázi.

Pro vytvoření serveru byl zvolen dokumentově orientovaný databázový systém Couchbase. Velikým pozitivem při jeho porovnání s MySQL byla jeho orientace na práci s JSON dokumenty, které nemusí mít přesně definovanou strukturu. Právě dynamická struktura dokumentů umožní snadnou implementaci nových funkcionalit včetně nových a dopředu neznámých typů datových souborů. Proti CouchDB, která má také výše uvedené vlastnosti, disponuje rozsáhlými SDK, které velmi usnadní vývoj klientské aplikace. Proti souborovému systému v kombinaci s Gitem je rozdílovým parametrem automatická synchronizace a umožnění jednoduchého verzování přímo na straně Couchbase a minimální nutnost zásahů uživatelů budoucí aplikace do správy verzí.

3.2.2 Klientská část

Pro tvorbu klientské části aplikace byl zvolen jazyk Java. Hlavní důvod je multiplatformnost, kterou umožňuje JVM (Java Virtual Machine). Dále pak velké množství platforem pro budoucí aplikaci, které usnadní vývoj jak samotného jádra aplikace tak později i doplňkových modulů. Pro bližší informace o jazyku kniha Učebnice jazyka Java [3].

Systém se bude skládat ze dvou částí, jedna bude fungovat jako úložiště dat a druhá jako samotná klientská aplikace, která bude umožňovat správu dat v lokálním úložišti a jejich synchronizaci vůči serveru.

3.2.2.1 Databáze

Pro použití databáze v klientské části bude použitá stejná technologie jako u serverové. Couchbase dovoluje snadnou synchronizaci dat mezi dvěma databázemi a proto je velmi vhodné ji použít i jako úložiště v desktopové aplikaci.

Couchbase je sice pomalejší než MySQL, ale poskytuje obrovskou výhodu při synchronizaci dat se serverem a práce Couchbase v rámci klienta by neměla být tak náročná, aby byla rychlost oproti MySQL výrazněji znát.

3.2.2.2 Platforma

Pojem platforma je v tomto kontextu míněno jako soubor frameworků pro správu verzí a modulů aplikace. Dále pro framework pro práci s UI, jak se samotným návrhem vzhledu, tak i k implementování akcí při událostech vyvolaných UI.

Jako možné platformy pro implementaci klientské aplikace byly vybrány celkem tři v jazyce Java. Platformy poskytují podobné funkce, jen se liší použitými komponentami a technologiemi. Předchozí verze IVE byla realizována pomocí NetBeans RCP.

- NetBeans RCP
  - Tato platforma patří mezi Java RCP (Rich Client Platform). Pomocí této platformy je vyvinuto několik nástrojů, z nichž jeden z nejznámějších je NetBeans IDE. Pro více informací viz webové stránky projektu [19].
    - podpora v JDK
    - Out of the box, Maven, Ant
    - rozsáhlé návody a příklady
    - Swing UI toolkit
    - předchozí verze aplikace napsaná v této platformě
- Eclipse RCP
  - RCP použité pro vytvoření Eclipse IDE. Pro více informací viz webové stránky projektu [13].
    - Proprietální build system
    - SWT UI toolkit
    - Equinox - silná podpora OSGi systému modulů
- IntelliJ RCP
  - RCP použité pro vytvoření několika profesionálních nástrojů z dílny IDEA (IntelliJ, PhpStorm, Android studio a další). Pro více informací viz webové stránky projektu [14].

3.2.2.3 Vyhodnocení

Při porovnávání platforem mezi sebou, nebyly nalezeny výraznější rozdíly, které by rozhodly ve prospěch jediné platformy.

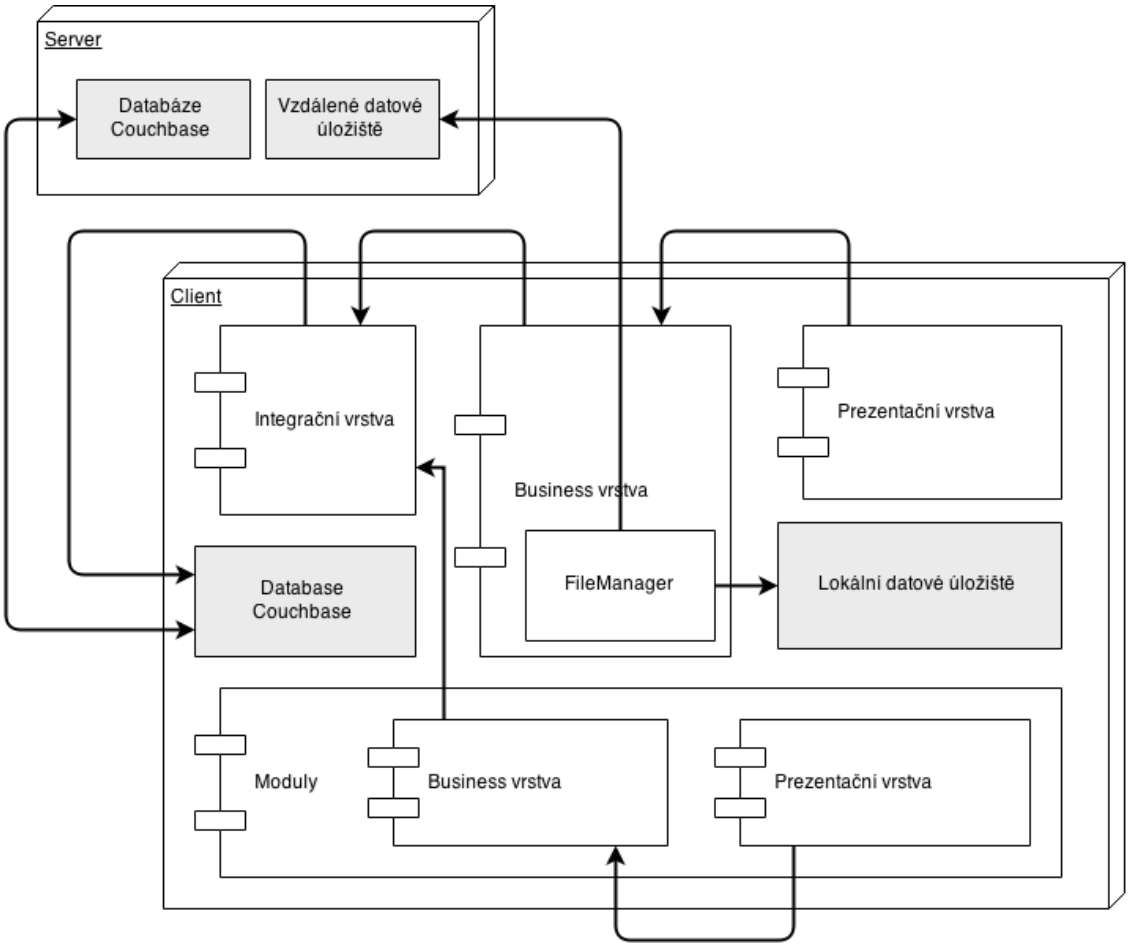
Hlavní důvod k výběru platformy NetBeans bylo použití v předchozí verzi aplikace. Díky tomu by mohly být určité moduly použity i pro novou verzi této aplikace.

3.3 Návrh klientské aplikace

Cílem této práce je vytvořit více uživatelskou aplikaci typu klient-server pro záznam a analýzu testů použitelnosti. Návrh a její struktura je popsána v této sekci.

3.3.1 Rozdělení vrstev jádra aplikace

Škálovatelnost aplikace je docílena použitím modulů pro jednotlivé části či funkčnosti aplikace. V základní verzi aplikace jsou obsaženy celkem tři moduly, které představují vrstvy MVC aplikace. V aplikaci není implementován přímo model MVC, ale jeho odvozenina, která je typická spíše pro webové prezentace - integrační, business a prezentační vrstva (viz Obrázek 3.3). Význam jednotlivých modulů či vrstev je popsán dále.



Obrázek 3.3: Schéma celého systému včetně vrstvy modulů. Silně vyznačené linky s šipkami označují vznik a směr volání funkcí pro manipulaci s daty.

Právě rozdělení jádra aplikace na vrstvy/moduly umožňuje další rozšiřování systému, ať už se bude jednat o rozšiřování samotného jádra aplikace, či tvorbě nových modulů. Nové moduly tak nemusejí vyžadovat nepotřebné závislosti, které jinak jádro systému potřebuje.

3.3.1.1 Integrační vrstva

Integrační vrstva je nejnižší vrstva aplikace. Její jediný úkol je vytvořit jednotné rozhraní nad datovým úložištěm tak, aby umožnila snadnou výměnu databáze bez nutnosti rozsáhlých změn ve vyšších vrstvách. Typické vstupy do této vrstvy jsou v podobě modelových objektů, které jsou pomocí interních funkcí integrační vrstvy namapovány na datové úložiště.

V této práci nebyl kladen tak velký důraz na jasné oddělení modulů business a integrační vrstvy z důvodu složitější implementace a relativně velké propojenosti práce s databází i v rámci funkcí v business vrstvě. Tato skutečnost znamená i složitější implementaci a problémovější změnu typu databáze.

3.3.1.2 Business vrstva

Business vrstva obsahuje hlavní logiku celé aplikace. Veškeré náročnější operace jsou vykonávány v rámci této vrstvy. V podstatě jde o univerzální jádro aplikace, které by mělo umět operovat nad libovolnou databází respektive integrační vrstvou. Její veřejné metody jsou obvykle popsány pomocí Facade, která zaštiťuje jednotlivé useCase, tedy případy užití. Jedná se o systém, který podporuje atomické zpracování dat, jak jejich uložení a vyvolání, tak obsahuje i různé další úpravy. Data předávaná integrační vrstvě by měla být upravena tak, aby byla zajištěna kompaktnost celé databáze. Tok dat směrem k prezentační vrstvě by měl být v podobě modelových tříd či jejich dávek.

V rámci této aplikace je business vrstva provázána s vrstvou integrační více než je obvyklé, což v budoucnu může způsobit problémy při přechodu na jiný typ databáze. V dalším vývoji aplikace by měl být kladen důraz na větší oddělení těchto vrstev a přetvoření business jádra tak, aby jednotlivé třídy modelu byly využívány v integrační vrstvě a nikoliv tak, aby se v business vrstvě mapovaly do JSON souboru a poté teprve předávaly do nižší vrstvy.

3.3.1.3 Prezentační vrstva

Prezentační vrstva je určena k prezentování dat uživateli. Tato vrstva v podstatě přebírá upravené verze dvou vrstev z MVC a to View, hlavně v podobě šablon realizovaných pomocí tříd představujících konkrétní části uživatelského rozhraní. Druhá převzatá vrstva je velmi odlehčená a slouží pouze k odchytávání akcí uživatele a případně drobné úpravy dat z business vrstvy. Tato vrstva by však neměla operovat nad žádnými čistými daty z databáze, ale pouze s objekty. Měla by jen minimálně operovat s kontrolou oprávnění, pokud se to týká přístupu k jednotlivým funkcionalitám, spíše pokud potřebuje pro zjištění, zda lze zobrazit dané funkcionality.

Prezentační vrstva v tomto projektu slouží čistě ke zpracování dat od uživatele a jejich přípravu pro business vrstvu (kontroly zda, je vstup validní, ořezání prázdných znaků apod.). Dále slouží k zobrazení a interakci s jednotlivými komponentami jako je například struktura projektu, kde se stará o vykreslování stromu jednotlivých projektů, ale také o odchytnutí požadavků pro vytvoření nových či editaci starých souborů.

3.3.2 Základní datové typy

Součástí základní instalace aplikace by neměly být žádné neobvyklé či konkrétní datové typy. V rámci diplomové práce jsou implementovány pouze dva základní moduly pro správu logů a videa. Vzhledem k pozdějším potřebám uživatelů aplikace budou pravděpodobně vznikat potřeby dalších typů datových souborů jako například dokument fotografie a další. Veškeré tyto soubory nelze v rámci pilotní implementace zahrnout a je pravděpodobné, že ani později se nepodaří pokrýt všechny požadavky na funkce aplikace. Proto byl kladen důraz na snadné rozšiřování a úpravu součástí aplikace.

3.3.2.1 Obecné schéma datového souboru

Veškeré dokumenty uložené v databázi by měly být pro snazší správu popsány stejným způsobem. Bohužel kvůli rozdílnosti obsahu různých typů dat je obtížné zavést strukturu, která by kompletně popsala veškeré možné typy dat včetně těch, které zatím nejsou známy. Z tohoto důvodu byl zaveden pouze určitý druh popisu datového souboru, který poté obsahuje konkrétní strukturu dat.

Nejobecnější typ souboru, se kterým bude aplikace pracovat, bude obsahovat pouze název, popis, označení typu a identifikátor souboru. Dokument, který bude provázaný s určitým participantem, bude obsahovat ještě identifikátor daného participanta. V případě, že bude implementován nový typ datového souboru, systém s ním díky této hlavičce bude umět pracovat a nabídne koncovému uživateli aplikace vhodné akce pro jeho správu.

3.3.2.2 Záznam aktivit participanta

Záznam aktivit participanta neboli log je nejběžnějším typem dat, která vznikají při uživatelském testování. Z pravidla se jedná o záznam aktivity participanta v závislosti na čase. Jedná se tedy o relativně jednoduchou strukturu.

Záznam aktivit participanta může být předlohou pro velkou spoustu dalších typů dat. V pilotní implementaci bude jako časový údaj použito celé datum včetně času s přesností na milisekundy. To umožní přesné určení okamžiku, kdy došlo k akci daného participanta. Jako další možná data, pro které by mělo být možno použít tento typ, lze považovat například logování polohy dle GPS, což by nevyžadovalo výraznější změny vůči původnímu souboru, nebo například použití u popisování videa.

Právě u popisků lze narazit na problém s časem u záznamu akcí. Vzhledem k použití celého data včetně časového údaje pro identifikaci určitého textového záznamu (ať už se jedná o záznam aktivity participanta, pozici GPS či anotace) v čase, bylo by velmi komplikované, použít toto datum i pro popis videa. Pro anotování je tedy vhodné vytvořit odvozený modul, který umožní začít v pomyslném čase *nula* odpovídající začátku videa a od něj pak počítat časovou osu.

Použití Couchbase jako úložiště databáze umožňuje velmi rychlé úpravy struktury dokumentů bez nutnosti vytvoření speciálních tabulek nebo jiných zásahů do integrační vrstvy. Právě nutnost minimálních zásahů v nejnižší vrstvě aplikace je na úkor těsnější závislosti mezi business a integrační vrstvou.

3.3.2.3 Video

Video záznam také patří mezi základní data, která mohou během testování s uživateli vzniknout. Patří mezi nejdůležitější data, která se dají znovu těžit, mnohem lépe popisují komplikovanější situace. Jeden z problémů videa však může být právě jeho komplexnost a je vhodné ho doplnit popisky či anotacemi.

3.3.3 Moduly v aplikaci

V aplikaci jsou použity dva typy modulů, integrované v jádru aplikace a externí moduly.

Oba typy modulů mají stejnou strukturu a obsahují pouze dvě vrstvy, prezentační a business. Avšak liší se od sebe tak, že integrované moduly zachovávají logické rozdělení aplikace a jsou implementovány do jejích vrstev. V kontextu s jádrem aplikace, které je rozděleno na integrační, business a prezentační vrstvu, které jsou samotné definovány pomocí NetBeans modulů, může být tento systém trochu nepřehledný. Druhý typ modulu už je sám o sobě jeden NetBeans modul, který je rozdělen na vrstvy pomocí několika balíčků.

3.3.3.1 Integrované moduly

Modul v podobě souhrnu funkcionalit může být implementován přímo do základního jádra aplikace, takový je označen za modul integrovaný (viz Obrázek 3.4). Tyto moduly by měly být obecné a měly by mít velkou šanci na využití ve většině projektů. Integrované moduly nemusí být v kódu nijak striktně odděleny, jejich integrace do jádra tak umožňuje mnohem větší provázanost s jádrem než moduly dodatečné/externí. V aplikaci jsou integrovány dva takovéto moduly, první je pro správu jednoduchých záznamů akcí participanta a druhý je pro přehrávání videa.

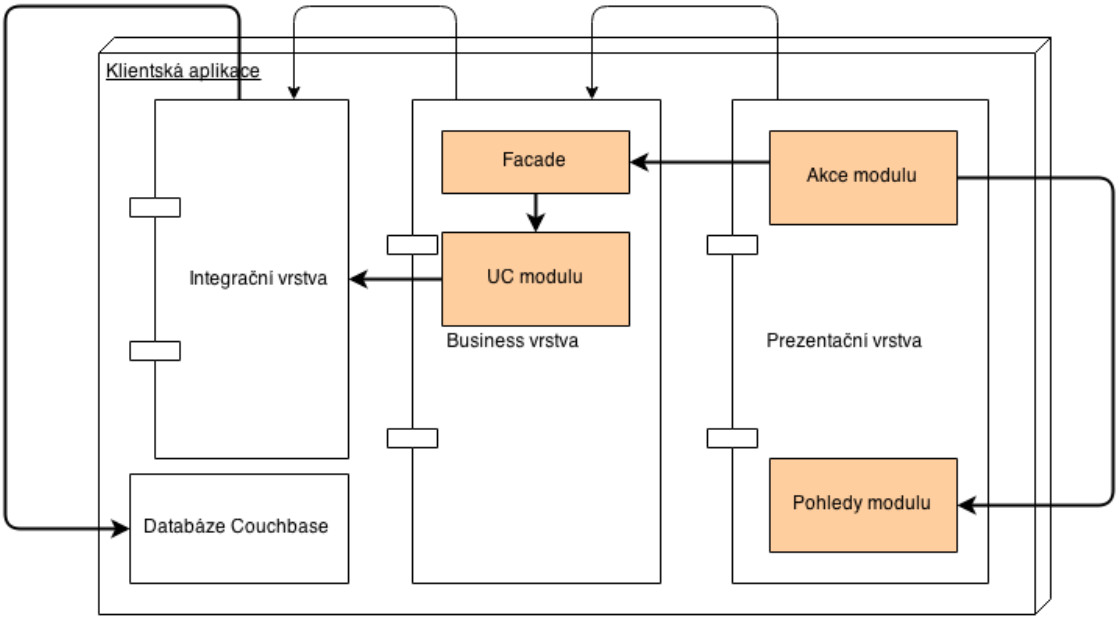
3.3.3.2 Externí moduly

Druhý typ modulu lze nazvat pluginem, který bude moci být nainstalován do základní aplikace bez nutnosti zasahovat do kódu a přebírá význam modulu z NetBeans platformy. Jedná se o oddělený soubor funkcionalit, který je minimálně závislý na svém okolí.

Základní typ modulu by měl obsahovat pouze dvě vrstvy a to business, která bude skrze integrační vrstvu přistupovat k databázi a prezentační, která bude uživateli umožňovat s daty pracovat (schéma externího modulu viz Obrázek 3.5). Tento přístup umožňuje právě paradox nezávislosti integrační vrstvy na vrstvě business, ale silnější opačnou závislost.

3.3.4 Datová úložiště

Vzhledem k použití velkého množství dat je vhodné použít taková úložiště, která umožní snadnou manipulaci a distribuci těchto dat mezi koncové uživatele. Datová úložiště mohou obsahovat různé druhy dat od textových souborů přes snímky obrazovky či fotografie až po videa. Právě rozmanitost zpracovávaných dat komplikuje jejich distribuci.



Obrázek 3.4: Schéma integrovaného modulu v aplikaci včetně vyznačení volání jednotlivých součástí. Modul je zakreslen do vrstev aplikace.

3.3.4.1 Databáze

Couchbase databáze je rozdělena na dvě logické jednotky. První, která představuje soubor serverů je pojmenován *cluster*. Cluster může být jeden, ale i víc serverů, které obsahují buckety. Bucket lze přirovnat k instanci relační databáze, jednotlivé soubory v bucketu mohou být chápány jako tabulky dané databáze. Pro práci s bucketem je potřeba znát název daného bucketu a pokud je zadáno tak i heslo k bucketu. Pro administraci bucketů či dalších nastavení daného clusteru je potřeba mít vytvořený administrátorský účet.

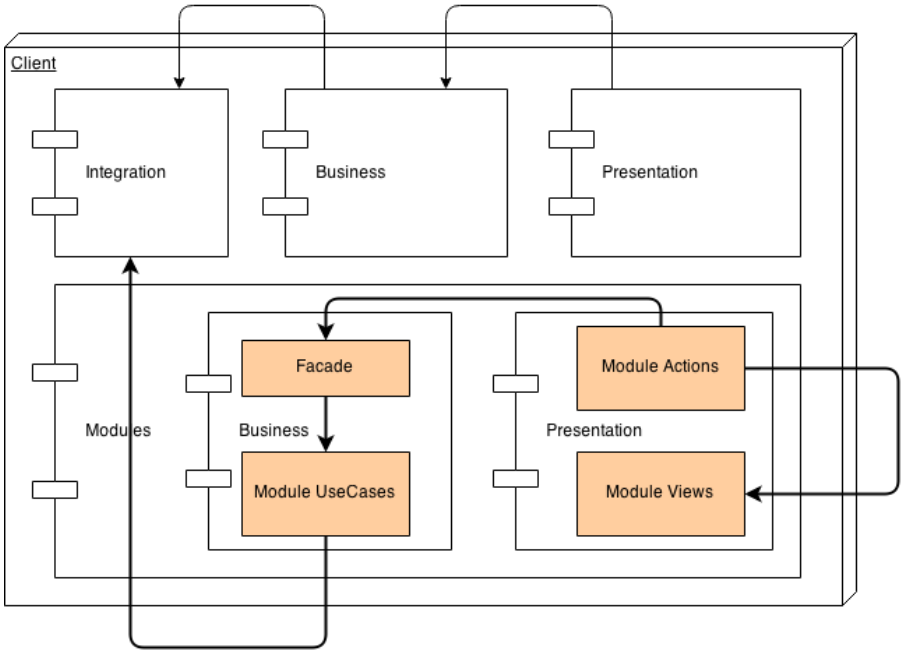
V této aplikaci bude zvolena jiná struktura databáze. Celý cluster bude představovat úložiště projektů. Bucket je chápán jako samostatný projekt, což zjednodušuje jeho synchronizaci se serverem. Jednotlivé soubory v bucketu reprezentují a popisují soubory, které vznikají během testování. Toto rozvržení bude použito kvůli přiblížení struktury předchozí verze aplikace, která pracovala nad daty v podobě adresářové struktury.

Bližší informace ohledně správy Couchbase serveru je možné nalézt v knize Pro Couchbase Server [6].

3.3.4.2 Externí soubory

Při uživatelském testování vzniká velké množství dat, která nelze uložit v podobě textových souborů. Mezi základní typy těchto dat patří například video, obrázek či zvukový záznam. Ukládání takových dat může být komplikované a v rámci této práce byly uvažovány dva způsoby řešení.

Prvním řešením bylo využít schopnosti noSQL databáze Couchbase, která je dokumentově orientovaná a umožňovala by tak uložit videa či obrázky v binární podobě. Toto řešení je



Obrázek 3.5: Schéma externího modulu v aplikaci včetně vyznačení volání jednotlivých součástí. Modul je zakreslen do vrstev aplikace.

zdánlivě jednoduché a vytváří pocit větší kompaktnosti dat. Je zde však mnoho problémů, které by bylo nutno řešit. Lze předpokládat problémové přehrávání souborů, kdy by bylo potřeba načíst celý soubor do paměti a poté ho přehrávat. Další možností bylo při prvním přehrání tento soubor zároveň s přehráváním ukládat do lokálního úložiště, avšak tento proces stále neřešil zbytečné ukládání dat v databázi, které by mělo jediný důvod a to pohodlnou synchronizaci.

Výsledné řešení je tedy vytvořit druhé úložiště pro binární data, která nebudou ukládána v databázi. Toto úložiště by mohlo být nasazeno i na jiném serveru než je nasazený Couchbase. Pro pilotní implementaci by měl být tento server schopný přijmout a uložit data, ale také tato data na vyžádání aplikaci zprostředkovat.

3.3.4.3 Synchronizace dat

Vzhledem k potřebě týmové práce na konkrétních projektech vzniká nutnost synchronizovat tato data napříč klientskými aplikacemi. V původním návrhu měla být dvojí synchronizace dat, avšak díky oddělení velkých souborů jako jsou videa, obrázky či audio nahrávky lze ponechat v rámci Couchbase pouze automatickou synchronizaci pracující na vrstvě Couchbase a poté ruční synchronizací externích souborů.

Synchronizace je tedy z toho důvodu rozdělena na dvě. První synchronizace bude zajištěna pouze fyzickou vrstvou databáze a pracuje s textovými dokumenty uloženými v Couchbase. Druhá bude realizována funkčností aplikace a bude se týkat hlavně binárních souborů, jako jsou videa či obrázky.

3.3.5 Autorizace v aplikaci

Kvůli rozdělení databázové struktury je potřeba dvojí autorizace. V běžném případě při práci s daty je potřeba pouze přístup k danému bucketu. Předpokladem k této situaci je, že přístup do bucketu má pouze takový uživatel, který zná jak název bucketu/projektu, tak i přístupové heslo. Proto není v aplikaci implementován jakýkoliv výběr bucketů ze serveru.

Při vytváření nových projektů/bucketů jak na vzdáleném serveru, tak na lokálním serveru uživatele aplikace, je nutné se přihlašovat pomocí administrátorského účtu. Dle předpokladu bude mít práva na vytváření nových bucketů administrátor vzdáleného serveru či uživatel klientské aplikace.

3.3.6 Datum a čas

Datum a hlavně čas je v uživatelských testech velmi důležitý parametr. V základních datech jako je například záznam uživatelské akce či poznámka je záznam času hlavní identifikátor, protože bez něj může daný záznam úplně ztratit smysl. Proto je důležité s datem a časem správně pracovat, popřípadě ponechat velikou volnost v jejich formátech, aby v uložení času mohlo být pro každý modul specifické a odpovídalo tak určení tohoto identifikátoru v rámci modulu.

V rámci této diplomové práce byl implementován jediný modul, který pracuje s časem. Při snaze o obecnost je zde použit celý formát času i s datem s přesností na milisekundy. Takovéto použití by mělo být dostatečně obecné pro použití ve velkém množství projektů.

Výše zmíněný formát však trochu komplikuje použití při posunu pomysleného času 0. To by mohlo být například při anotování videa, kdy může být začátek časové osy posunut na začátek videa. Toto však už závisí na konkrétní implementaci v daném modulu. Integrační vrstva dovoluje ukládat jakékoliv formáty přeložené do textového řetězce.

Formát data a času nebyl tedy na základě možnosti rozdílných přístupů k času zvolen pevně, ale byla pro jeho volbu ponechána veliká volnost, aby mohl být využit maximálně na základě požadavků, které mohou vyvstat až v průběhu delšího používání aplikace.

Kapitola 4

Realizace

V této kapitole je popsána implementace jednotlivých částí včetně způsobu řešení vybraných problémů. Cílem kapitoly je nastínit nejen způsob řešení problémů, ale seznámit i s filozofií celého projektu a zdůvodnit některé kroky, které vedly k realizaci daných komponent aplikace. V každé sekci této kapitoly bude vždy naznačen smysl daného problému, vysvětlení situace a požadavků a na závěr popsán postup řešení. Podrobnější informace o konkrétních vlastnostech lze nalézt ve knihách zabývající se NetBeans RCP [1] [7].

4.1 Rozdělení vrstev na moduly

Rozdělení vrstev bylo provedeno pomocí modulů NetBeans platformy. Tato implementace je vhodná zejména kvůli využití nástrojů platformy, pro případné výměny celých vrstev aplikace.

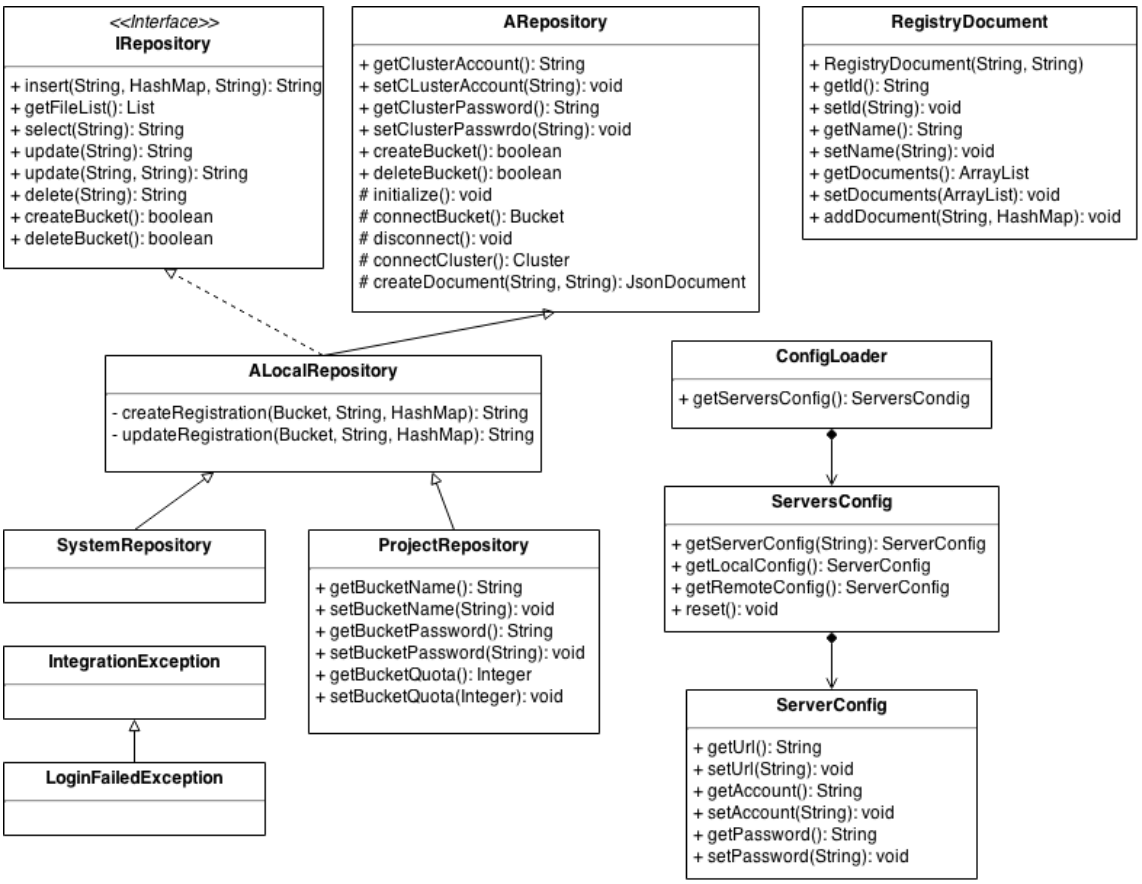
Výměna modulů může probíhat buď v grafickém rozhraní aplikace pomocí správce modulů, nebo v případě globální změny, týkající se základní aplikace, lze vybraný modul nahradit fyzicky přímo v kódu změnou závislostí.

4.1.1 Integrační vrstva

Integrační vrstva implementovaná v tomto projektu má za úkol vytvořit rozhraní pro komunikaci s databází Couchbase. Jako hlavní prvky rozhraní jsou potomci abstraktní třídy `ARepository`. V aplikaci jsou reálně použité dvě takové třídy. `ProjectRepository` pro správu dat v jednotlivých projektech druhá třída je pak `SystemRepository`, která má za úkol uchovávat data o jednotlivých projektech a v budoucnu například další systémová nastavení. `ARepository` poskytuje základní funkce pro práci s vybraným bucketem i jeho vytváření a registraci informací o něm v `SystemRepository`.

Integrační vrstva je doplněna o správce konfiguračních souborů, které jsou důležité pro definování přístupových údajů k vybraným clusterům. Data v konfiguračních souborech jsou uložena ve formátu Json.

Pro přehled tříd viz 4.1.



Obrázek 4.1: Diagram tříd integrační vrstvy. Na diagramu jsou znázorněny dědičné závislosti tříd Repository včetně implementovaného rozhraní. Dále je zobrazen i ConfigLoader pro načítání konfiguračních souborů integrační vrstvy včetně načtení nastavení serverů.

4.1.1.1 Repozitáře

`ProjectRepository` je poměrně obecné řešení a pracuje nad bucketem, který je vybrán uživatelem. Tato třída nemá tedy nastavený pevný název ani heslo bucketu, ke kterému se připojuje. Komunikace s databází přes repozitář se provádí pomocí metod rozhraní `IRepository`. Tento repozitář tedy dovoluje dynamicky definovat název i heslo v závislosti na potřebách uživatele, při správné konfiguraci dovoluje uživateli přístup k vybranému bucketu.

`SystemRepository` je obdoba dříve zmíněného repozitáře. Význam `SystemRepository` však není ve správě dat uživatele či projektů, ale obsahuje různá systémová nastavení, která jsou vhodná k uchování pomocí databáze Couchbase. Tento bucket vznikl hlavně jako implementace chybějící funkce Couchbase Java SDK pro výběr všech bucketů v clusteru, obsahuje informace o bucketech včetně jejich hesel v plain textu. Použití nešifrovaných hesel je bezpečnostní chyba, avšak jejich smysl je v projektu jen jako další identifikátor a týkají se jen nastavení pro lokální databázi.

Velký problém systému Couchbase je slabá podpora dotazovacích jazyků. Poslední dobou se sice komunita, která za vývojem systému stojí, snaží implementovat možnost dotazování, avšak celý tento modul je zatím jen v alfa verzi, která není plně funkční a pro tento projekt se prozatím ukázala jako nedostatečná (podrobnější informace na webových stránkách projektu [11]). Proto bylo třeba vytvořit takové prostředí, které by umožnilo alespoň výběr všech dokumentů obsažených v daném bucketu. Tato funkce byla docílena zavedením `RegistryDocument`, které obsahují seznam všech souborů v bucketu. Pro práci s `RegistryDocument` byla vytvořena další abstraktní třída `ALocalRepository`, kde jsou definovány metody pro vytvoření `RegistryDocument` a jeho další aktualizaci. Tato třída je potomkem `ARepository` a implementuje i rozhraní `IRepository`, zároveň slouží jako přímý rodič `SystemRepository` a `ProjectRepository` (kompletní vývojářská dokumentace [10]).

4.1.1.2 Konfigurační soubory

```
{
        "configs" : {
                "local" : {
                        "url" : "localhost",
                        "account" : "administrator",
                        "password" : "12345_my_password"
                }
        }
}
```

Obrázek 4.2: Příklad konfiguračního souboru servers.config ukazuje nastavení pro lokální databázový server s administrátorským jménem a heslem.

Význam konfiguračního souboru je hlavně v definování přístupových údajů k lokálnímu Couchbase serveru (příklad konfiguračního souboru 4.2)●. Konfigurační Json soubor má za

úkol nastavit přístupy k serverům. K dispozici je několik možností jak tento soubor použít. V základu lze nastavit administrátorské jméno a heslo k lokálnímu serveru, toto nastavení určitě není bezpečné a není doporučené ho používat. Druhá volba nastavení souboru je nastavení serveru, zde lze nastavit tři parametry: administrátorské jméno, heslo a url adresu vzdáleného serveru.

Z celého konfiguračního souboru je nejdůležitější právě url adresa vzdáleného serveru. Díky tomuto nastavení lze přepnout práci s daty přímo na remote server bez nutnosti použít Couchbase lokální. V případě správy vzdáleného serveru však už není umožněna práce s více projekty najednou, ale pouze s jediným, jehož jméno a heslo uživatel aplikace vložil při přihlášení.

Načtení všech použitých konfiguračních souborů zabezpečuje `ConfigLoader`. Pro načtení konfiguračního souboru serveru slouží třída `ServersConfig`, v této třídě jsou implementovány metody pro získání konfigurace lokálního serveru, vzdáleného serveru, ale i libovolného serveru dle jména nastavení. Nastavení konkrétního serveru obaluje třída `ServerConfig`. Tato nastavení jsou pak pomocí HashMapy načítána do `ServersConfig`.

Načítání tohoto konfiguračního souboru je nasazeno v integrační vrstvě kvůli úzkému kontaktu s databází Couchbase. V případě dalších konfiguračních souborů, které se už nebudou tolik týkat této databáze, je vhodné implementovat je do business vrstvy.

4.1.2 Business vrstva

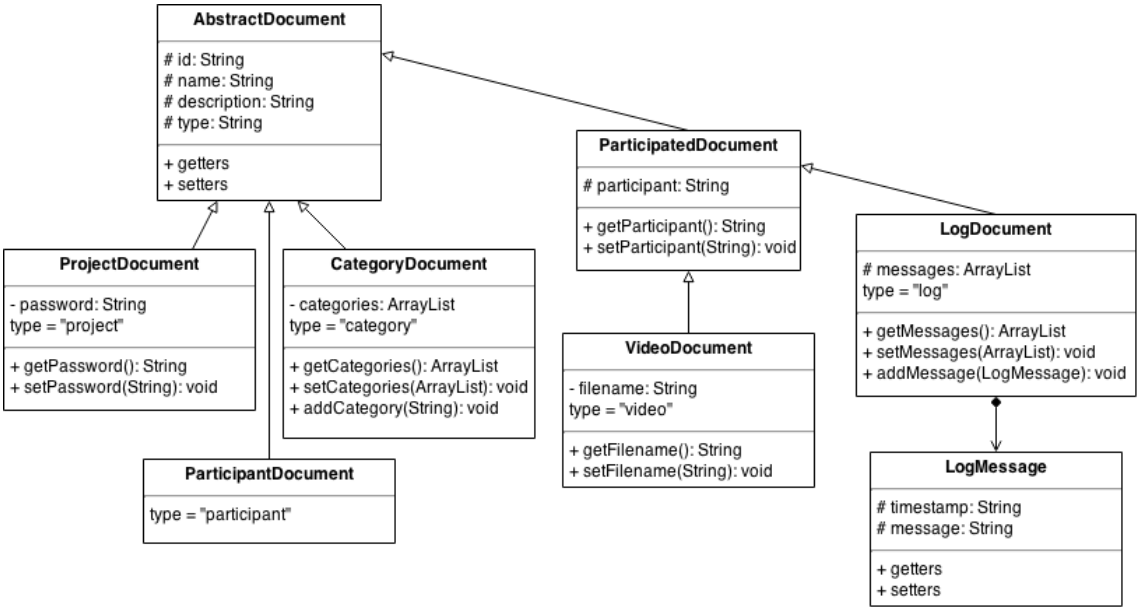
Business vrstva obsahuje logiku celé aplikace. Jde v podstatě o přípravu dat, se kterými pracuje prezentační vrstva a předkládá je uživateli nebo naopak přípravu vstupů z prezentační vrstvy a jejich zpracování pro integrační vrstvu. Tato vrstva je rozdělena na dva logické celky. Prvním celkem je datový model systému, který obaluje jednotlivé typy dat a zjednodušuje tak manipulaci s nimi. Druhou částí systému je vlastní implementace business procesů, které s daty pracují.

4.1.2.1 Třídy modelu

Třídy modelu (viz Obrázek 4.3) představují samotná data, se kterými systém pracuje. Jde o data, která jsou ukládána a načítána v databázi a dají se chápat jako jednotlivé dokumenty, které uživatel vytváří na základě testů použitelnosti. Tyto modelové třídy obsahují pouze proměnné reprezentující data a metody pro přiřazení a získání těchto dat, takzvané settery a gettery. Vzhledem k velké vzájemné podobnosti jednotlivých dokumentů a potřebě generalizování jejich popisu vzniklo několik abstraktních tříd, které slouží jako šablony pro konkrétní implementaci dokumentu.

Základní třída `AbstractDocument` obsahuje hlavičku, která poskytuje jednotlivé soubory přímo v databázi Couchbase. Tato hlavička obsahuje id dokumentu, jeho název, popis a typ dokumentu. Typ dokumentu je určený k rozeznávání dokumentů mezi sebou a umožňuje tak prezentační vrstvě rozpoznat jak daný dokument zobrazit.

Přímo z `AbstractDocument` dědí třídy, které nejsou závislé na participantovi. První třídou je `ProjectDocument`. Instance této třídy jsou ukládány do systémového bucketu a obsahují název projektu a tedy i bucketu a stejně tak i heslo v plain textu. Dalším dokumentem



Obrázek 4.3: Diagram modelových datových tříd v business vrstvě

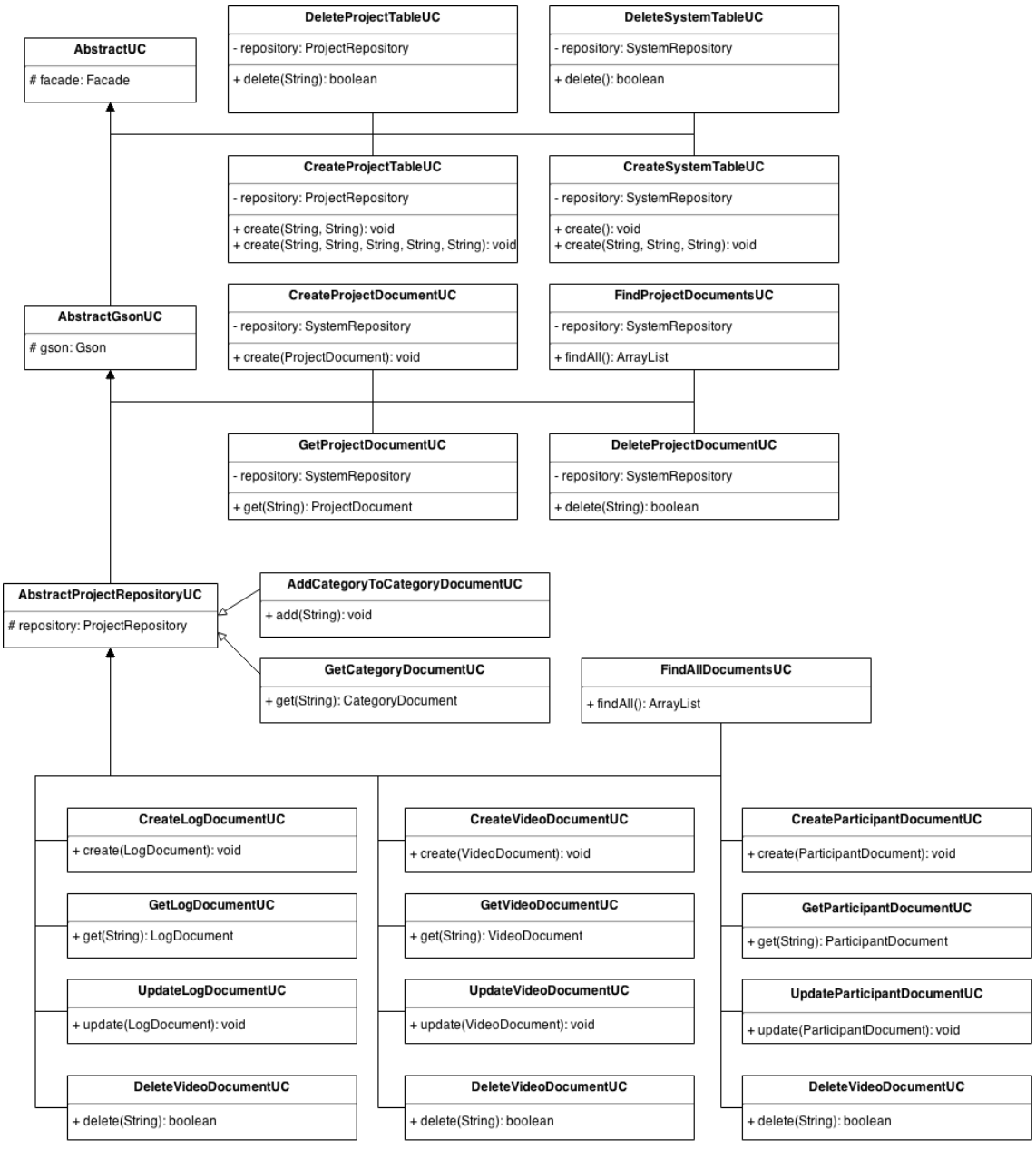
je `ParticipantDocument` pro konkrétního participanta v projektu. Tento a další zmíněné dokumenty už jsou ukládány v bucketech určených pro jednotlivé projekty. Dokument pro participanta obsahuje pouze staticky zvolený typ dokumentu a jeho obsah není oproti svému rodiči nijak rozšířen. Posledním potomkem, který lze instanciovat je `CategoryDocument`. V bucketu vždy existuje jen jediný dokument popisující kategorie. Oproti `AbstractDocument` je jeho obsah rozšířen o výčet kategorií, které lze pro projekt používat.

Jediná abstraktní třída, která je potomkem `AbstractDocument` je `ParticipatedDocument`. Tento dokument slouží jako předek pro konkrétní třídy popisující data získaná při testování s participantem a má tedy vazbu na participanta podle jeho identifikátoru.

Poslední dva dokumenty implementované v základní aplikaci jsou `LogDocument`, který obsahuje záznam aktivit participanta a `VideoDocument` pro popsání záznamu videa. `LogDocument` obsahuje výčet akcí, tyto akce jsou v podobě zprávy a časového údaje a jsou popsány vlastní třídou. `LogDocument` pak obsahuje seznam těchto akcí. `VideoDocument` obsahuje pouze název souboru, na který ukazuje.

4.1.2.2 Třídy případů užití a Facade

Třídy případů užití neboli *useCase* (dále jen jako UC) představují v základní aplikaci konkrétní akce, které pracují s datovými třídami (viz Obrázek 4.4). Z vyšší (v této aplikaci z prezentační) vrstvy jsou volány pomocí třídy `Facade`, která obsahuje základní funkce pro práci s business vrstvou. Obecně jsou tyto třídy rozděleny na třídy, které manipulují přímo s buckety, tyto případy užití jsou přímými potomky základního `AbstractUC`. Tato třída obsahuje přístup k `Facade`. Dále jsou třídy, které pracují nad dokumenty v databázi, tyto třídy jsou poděděny od abstraktního potomka základního UC `AbstractGsonUC`. `AbstractGsonUC` obsahuje instanci třídy Gson, která umožňuje vzájemného překladu Json na objekt. Přímí



Obrázek 4.4: Diagram tříd případů užití na business vrstvě

potomci pracují s dokumenty, které nejsou uloženy v projektovém bucketu. Posledním typem UC jsou potomci třídy `AbstractProjectRepositoryUC`. Tato třída pracuje s dokumenty v projektových bucketech a obsahuje instanci `ProjectRepository`. Potomci této třídy zajišťují CRUD metody pro práci s datovými objekty.

Třída `Facade` je implementace stejnojmenného návrhového vzoru, která zajišťuje snazší použití business vrstvy ve vyšších vrstvách a představuje jediný komunikační bod pro práci s celou business vrstvou. V tomto projektu je implementována ještě pomocí návrhového vzoru Singleton, který zajišťuje jedinou instanci této třídy napříč celým systémem. `Facade` obsahuje veškerá data potřebná pro správnou funkčnost business vrstvy a díky tomu její použití zjednodušuje budoucí vývoj dalších funkcí a modulů systému.

4.1.2.3 Správce souborů

Správce souborů je jediná abstraktní třída `FileManager`, jejíž potomci jen blíže specifikují typ a úložiště daných souborů. Tato třída také umožňuje synchronizaci lokálních datových úložišť se vzdálenými. Pomocí metod `push(String)` pro nahrání souboru na vzdálené úložiště a `pull(String)` pro stáhnutí. Další metody pro práci se soubory jsou `copyToLocalDirectory(File, String)` pro zkopírování souboru v počítači do lokálního úložiště, nebo `deleteFromLocalDirectory(String)` pro jeho smazání. Správce souborů dále obsahuje třídy pro vygenerování cest k úložišti či hashování jména souboru a zjištění, zda je soubor správného formátu.

Smysl a popis práce správce souborů je popsán v kapitole 4.4.

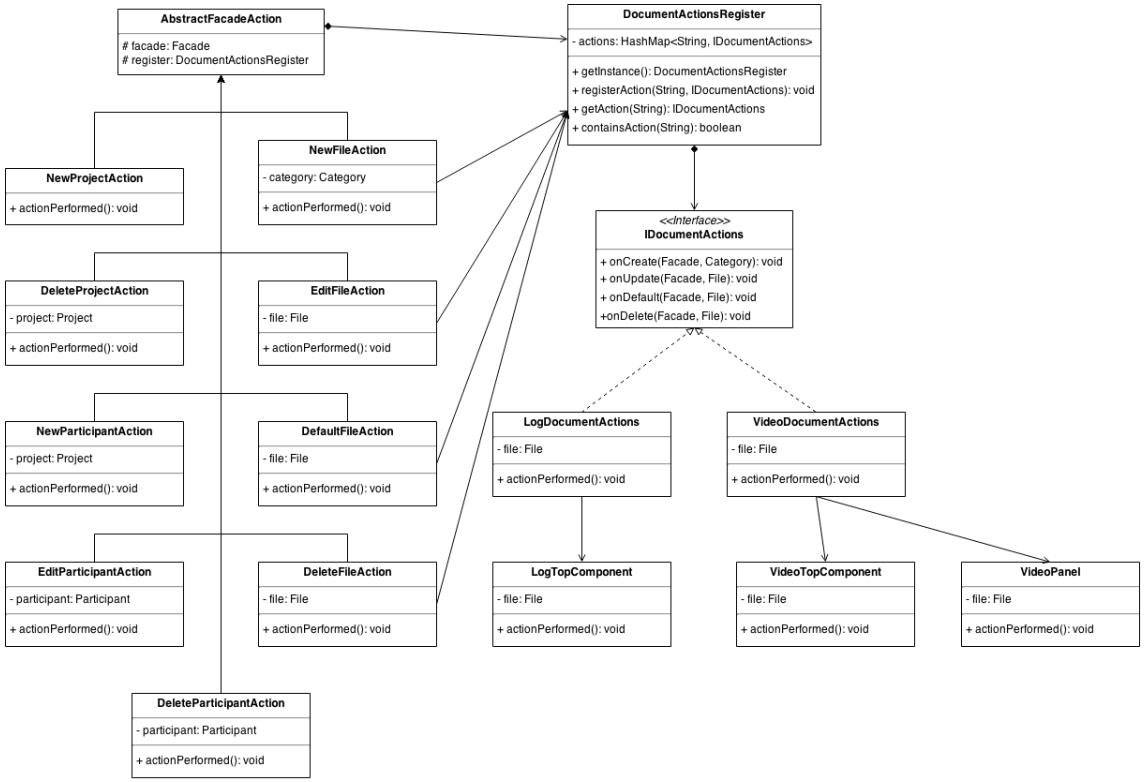
4.1.3 Prezentační vrstva

Prezentační vrstva slouží pro implementování grafického rozhraní, se kterým pracuje uživatel celé aplikace. Toto rozhraní je nejvyšší vrstvou celé aplikace a obsahuje vrstvu view, ale také vrstvu controller. Pomocí controllerů jsou připravována data, která se mají zobrazit a odchytávány akce uživatele. View slouží pouze k zobrazení dat.

Celé uživatelské rozhraní se skládá z potomků tříd `TopComponent` a `Panel`. `TopComponent` slouží k vykreslování rozhraní do hlavního okna aplikace. V základní verzi aplikace jsou dva druhy komponent `ProjectExplorerTopComponent`, která je vykreslována v části vyčleněné pro *explorer* (stromovou strukturu dat). Druhá komponenta slouží pro správu souborů a vykresluje se na místě editoru aplikace.

`ProjectExplorer` je ústřední komponenta celé aplikace a slouží k manipulaci s dokumenty jako takovými, k vykreslení struktury projektů či jejich vytvoření a editaci. Pro vykreslení struktury projektů je použita třída `BeanTreeView`. Kvůli relativně velké časové náročnosti je každý projekt pro vykreslení načten z databáze a poté zpracován jedenkrát, načítá se znovu jen po případné změně struktury projektu. Vytváření nových položek ve struktuře, ať už se jedná o nový projekt, soubor nebo nového participanta, je realizováno pomocí dialogových popup oken.

Uživatelé mohou upravovat strukturu `ProjectExplorer` svými zásahy. Tyto zásahy jsou odchytávány pomocí potomků třídy `AbstractFacadeAction` (vit Obrázek 4.5), která je potomkem `AbstractAction` z knihovny *Swing*. Konkrétní implementace této abstraktní třídy



Obrázek 4.5: Schéma akcí v prezentační vrstvě včetně významných komponent

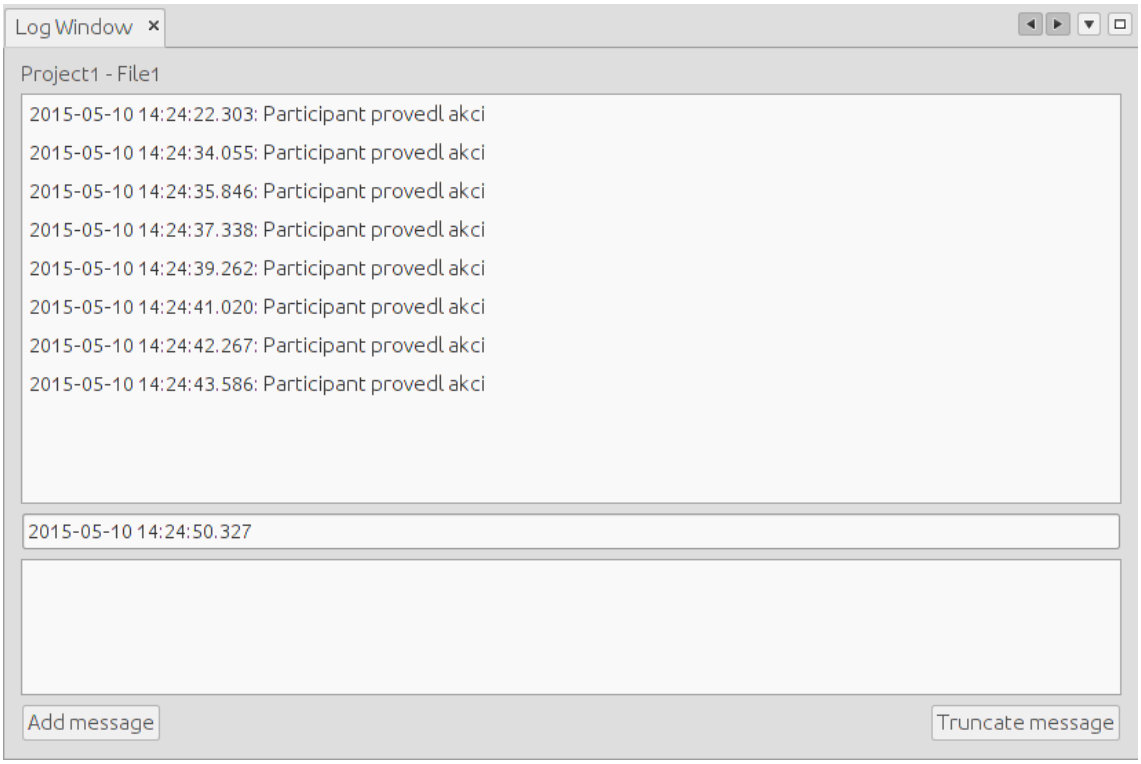
jsou volány v případě uživatelské interakce s rozhraním `ProjectExplorer`. Při aktivaci určité akce je obvykle volána funkce na `Facade` v business vrstvě.

Akce pro správu projektu a participanta jsou neměnné a jsou přímo definovány v těle potomka `AbstractFacadeAction`. Problém však nastává v případě různých typů souborů dat z testování. V aplikaci jsou implementovány dva druhy dat: log a video. Pro každé je vytvořen zvláštní modul, který je integrován do základní verze aplikace a v budoucnosti bude těchto modulů mnohem více, proto bylo nutné vytvořit dynamické načítání odpovědi na uživatelskou akci v závislosti na typu modulu. Právě z tohoto důvodu byl implementován `DocumentActionsRegister` a vytvořen interface `IDocumentActions`, který obsahuje obslužné metody `onCreate`, `onUpdate`, `onDelete` a `onDefault`. Tento interface implementují třídy akcí konkrétních modulů. Všechny akce obsahují privátní třídu `Listener`, která obsluhuje akce z grafického rozhraní, ať už `Panel` či `TopComponent`.

4.2 Základní datové typy a moduly v aplikaci

V rámci projektu byly identifikovány dva základní obecné datové typy. Prvním z nich je záznam aktivit participanta a druhým video nahrávka testu. Oba tyto datové typy jsou pokryty v základní verzi aplikace, která je výstupem této diplomové práce.

4.2.1 Záznam aktivit participanta



Obrázek 4.6: Snímek obrazovky pro zadávání logů

Záznam aktivit participanta je pro potřeby tohoto projektu realizován jen jako seznam dvojic čas-zpráva (ukázka realizovaného modulu viz Obrázek 4.6). Tento návrh dovoluje široké využití celého modulu pro mnoho dalších konkrétnějších modulů a není problém tento modul upravit pro jiné použití přepsáním několika tříd.

Modul záznam aktivit participanta je pro svou obecnost integrován přímo do základní verze aplikace a zasahuje do dvou vrchních vrstev aplikace. V business vrstvě jsou integrovány modelové třídy `LogDocument`, `LogMessage` a třídy případů užití, které s daty manipulují. V prezentační vrstvě je pak implementováno celé uživatelské rozhraní pro manipulaci s daty. Obě tyto vrstvy už byly, včetně diagramů, popsány v kapitole 4.1.

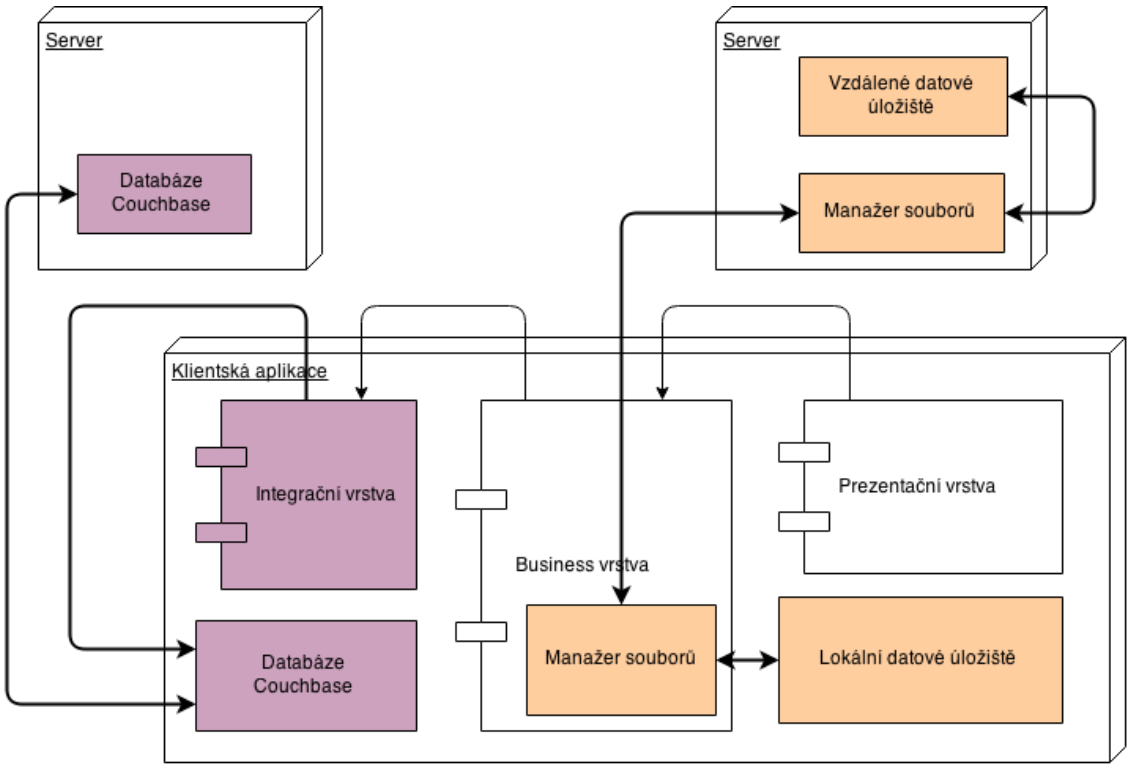
4.2.2 Video

V rámci tohoto projektu bylo implementováno přehrávání video souborů s koncovkou .avi, .mp4 a .wma. Pro správu těchto souborů v úložištích je vytvořen `VideoManager` (potomek `FileManager`). Pro vytváření a editaci videí byl vytvořen nový `VideoPanel`, který umožňuje k definici dokumentu přidat i cestu k souboru a díky tomu je možné, aby `VideoManager` toto video zkopíroval do lokálního datového úložiště.

Pro přehrávání videa byla vytvořena komponenta `VideoTopComponent`, která pro samotné přehrávání využívá knihovnu vlcj [23]. A umožňuje uživateli video přehrát, zastavit a díky posuvníku i vybrat část, která má být přehrávána.

Modul videa byl pro svou obecnost integrován přímo do základní verze aplikace a zasahuje do dvou vrchních vrstev systému. V business vrstvě je integrována modelová třída `VideoDocument` a třídy useCase, které s dokumentem manipulují. Oproti modulu záznamu akcí se v této vrstvě nachází také `VideoManager`. V prezentační vrstvě je pak obdobně jako u záznamu akcí implementováno celé uživatelské rozhraní. Obě vrstvy už byly, včetně diagramů, popsány v kapitole 4.1.

4.3 Datová úložiště



Obrázek 4.7: Schéma použití obou serverů

V projektu byly z nutnosti oddělení datových souborů od Couchbase databáze nasazeny dvě různá úložiště. Databáze Couchbase je určena pro správu dokumentů a dat textového charakteru. Druhé úložiště slouží k uchovávání externích souborů. Rozdělení práce jednotlivých úložišť je schematicky zobrazeno na obrázku 4.7.

4.3.1 Databáze

Pro databázové úložiště byl zvolen systém Couchbase obsahující velkou část funkcí, které jsou v tomto projektu vyžadovány. Největší výhodou této databáze je automatická synchronizace pomocí replikace celých bucketů.

4.3.2 Externí soubory

Tento server byl pro účely pilotního provozu implementován pomocí jednoduchého skriptu v jazyce PHP [21] a jeho funkčnost je velmi jednoduchá. Na zvolené adrese jsou implementovány dva typy requestů, jeden pro vložení dat a druhý pro jejich získání.

První je klasický dotaz metodou *GET* s url parametry filename a type (*http://adresa.server/index.php?filename=jmenosouboru.avi&type=video*), které reprezentují celý název souboru a jeho typ, podle něj poté server vyhodnotí, zda je soubor v úložišti k dispozici a pokud ano, poskytne ho klientu ke stažení.

Druhou metodou je dotaz *POST*, kterým se soubor na server vkládá. Dotaz *POST* musí obsahovat parametry se jménem souboru a jeho typem pro správné zařazení do filesystému na serveru. Poslední povinný atribut je samotný soubor, který má být na serveru uložený. Serverový skript soubor přijme a uloží do příslušné složky.

4.4 Synchronizace dat

V systému byla implementována dvojí synchronizace dat. První je zajištěna přímo fyzickou vrstvou databázového systému Couchbase. Druhá je implementována pomocí manuálního stahování a nahrávání souborů vůči serveru.

4.4.1 Couchbase synchronizace

Synchronizace dat probíhá vždy na pozadí přímo na databázové vrstvě Couchbase. Aplikace v pilotní verzi nemá k dispozici manuální synchronizaci se serverem, pouze automatickou, která ovšem manuální synchronizaci plně nahradí. Synchronizace je prováděna pomocí replikace celých bucketů.

Synchronizace se provádí pomocí nástroje *XDCR* ve webovém rozhraní serveru. Podpora tohoto nástroje zatím nebyla do Couchbase SDK implementována, proto je nastavení nutné řešit přes externí rozhraní. Při vytvoření replikace je nutné nastavit referenci na cluster, jehož bucketů se bude replikace týkat. Pro povolení je nutné znát přístupové údaje k danému clusteru. Po vytvoření reference je nutné zvolit, jaké buckety se mají replikovat. Pro správnou funkci dat je nutné vytvořit bucket jak na straně vzdáleného serveru, tak na lokálním serveru. Pro další informace ohledně nastavění serveru viz webové stránky projektu [9].

Na obrázku 4.7 je synchronizace Couchbase dat znázorněna pomocí fialové barvy. K těmto datům má přístup integrační vrstva, která obaluje vybranou databázi.

4.4.2 Synchronizace externích dat

Externí data, která nejsou uložena v Couchbase databázi jsou synchronizována pouze manuálně. Důvodem k tomuto kroku je možná velikost souborů, kdy nemusí být výjimkou, aby video soubor dosáhl velikosti kolem gigabyte, takže by stahování či nahrávání takto velkého souboru mohlo vést k příliš velkému vytížení internetového připojení a omezilo tak další práci a synchronizaci dat.

Synchronizace externích dat probíhá pomocí potomků třídy `FileManager`. Tato třída má za úkol spravovat lokální úložiště externích souborů a synchronizovat ho pomocí metod *pull* (stáhnout) a *push* (nahrát) se vzdáleným serverem.

Na obrázku 4.7 je synchronizace externích dat znázorněna pomocí růžové barvy. K těmto datům má přístup business vrstva.

4.5 Import dat

Při použití externích nástrojů pro záznam dat z uživatelského testování je potřeba tato data nějakým způsobem provázat s celým systémem. Protože není dopředu zřejmé, jaký formát dat bude potřeba importovat, je vhodné implementovat vlastní pluginy pro import těchto dat.

Usnadnění tvorby pluginů pro import by mohlo být v použití stávajících tříd pro práci se soubory jako je například `FileManager`. Vytvořit třídu dědice a upravit dané funkce tak, aby vyhovovaly potřebám pro import.

4.6 Autorizace v aplikaci

Při vytvoření nového projektu jsou tyto údaje uloženy na localhostu v systémovém bucketu, který by nikdy neměl být synchronizován se serverem.

Druhá autorizace je vůči serveru. Tato autorizace je vyžadována pouze v případě speciálních nastavení či správě bucketů. Jelikož jde o jméno a heslo administrátora serveru, není vhodné toto heslo držet kdekoliv v podobě plain textu, v ideálním případě by nemělo být toto heslo uložené nikde.

4.6.1 Účel konfiguračního souboru

Konfigurační Json soubor (viz Obrázek 4.2) má za úkol nastavit přístupy k serverům. K dispozici je několik možností jak tento soubor použít. V základu lze nastavit administrátorské jméno a heslo k lokálnímu serveru, toto nastavení určitě není bezpečné a není doporučené ho používat. Druhá volba nastavení je definování přístupů ke vzdálenému serveru kde je možné nastavit tři parametry: administrátorské jméno, heslo a url adresu vzdáleného serveru.

Z celého konfiguračního souboru je nejdůležitější právě url adresa vzdáleného serveru. Díky tomuto nastavení lze přepnout práci s daty přímo na vzdálený server bez nutnosti použít Couchbase lokální. V případě správy vzdáleného serveru však už není umožněna práce s více projekty najednou, ale pouze s jediným, jehož jméno a heslo uživatel aplikace vložil při přihlášení.

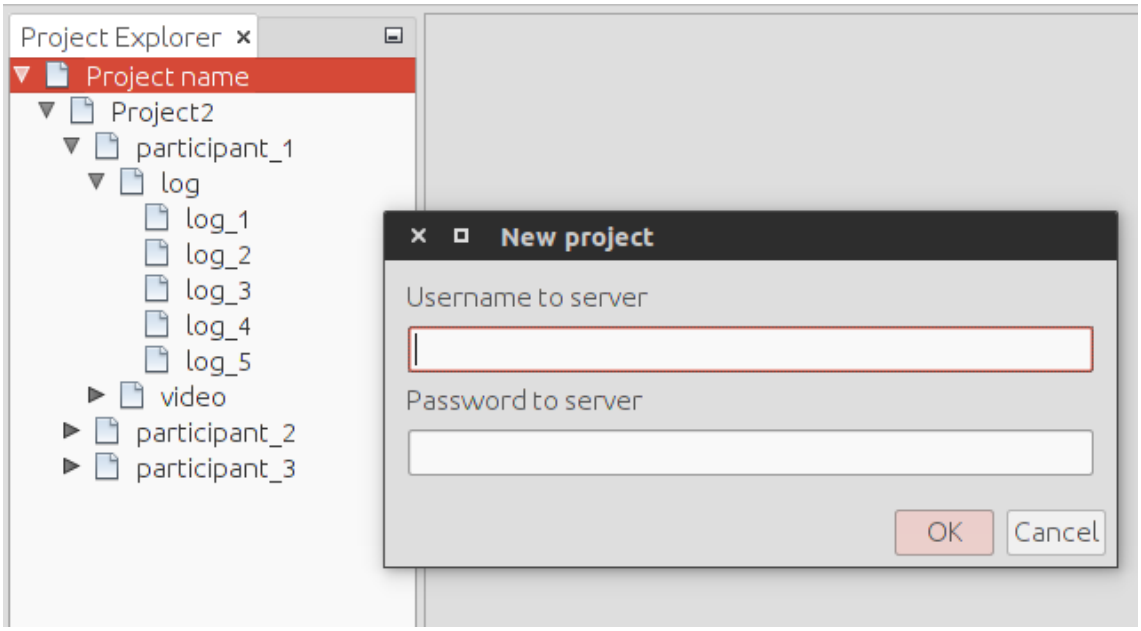
4.6.2 Bezpečnost hesel

Hesla k jednotlivým bucketům jsou uložena v systémovém bucketu, který by neměl být synchronizován se serverem. Hesla nejsou nijak šifrována, ale počítá se pouze s lokálním využitím těchto hesel.

Administrátorský účet by neměl být pro zvýšení bezpečnosti uložen ani v lokálním prostředí. Přestože aplikace umí pracovat s konfigurací hesel, neměly by být v konfiguračním souboru uloženy, kvůli jejich plain textové podobě.

4.6.3 Jiná možnost přihlášení administrátora

Doporučený způsob přihlášení k administrátorskému účtu je využití popup okna (viz Obrázek 4.8). Toto okno je aktivováno vždy v případě potřeby prvního přístupu k administrátorskému účtu. Pokud je správně vyplněno jméno a heslo v konfiguračním souboru, pak není popup okno nikdy vyvoláno.



Obrázek 4.8: Snímek přihlašovacího popup okna

4.7 Vytváření nového modulu

Pro vytvoření nového modulu jsou v základní aplikaci implementovány pomocné třídy a rozhraní, které jsou pro správný chod aplikace nutné dodržovat. Pro vytvoření nového dokumentu je vhodné modelovou třídu navrhnout jako potomka správného abstraktního dokumentu, stejně tak je vhodné použít podobný postup i pro vytvoření UC. Pokud je vytvářen integrovaný modul, pak je pro snazší budoucí práci doporučeno využít zaregistrování těchto UC na `Facade`. V případě tvorby dodatečného modulu není možné registrovat tyto UC na `Facade`.

Důležitý bod, který je stejný jak pro integrované tak pro externí moduly, je vytvoření příslušných akcí. Pro vyvolání obslužných dialogů a `TopComponent` slouží akce v `ProjectExplorer` (viz Obrázek 4.5). První krok při tvorbě prezentační vrstvy modulu je tedy vytvoření třídy,

která implementuje rozhraní `IDocumentActions`. Je nutné definovat všechny metody z tohoto rozhraní. Poté lze přistoupit k vytvoření dialogů a `TopComponent`, které se mají vykreslit jako odpovědi na tyto akce. Na závěr je potřeba registrovat objekt těchto akcí v `DocumentActionsRegister`. Vzhledem k tomu, že je pro registr použit návrhový vzor *Singleton*, je vhodné, aby byla třída akcí bezestavová. Aby bylo možné modul použít v daném projektu, je potřeba ho registrovat ještě jako kategorii v bucketu daného projektu.

4.8 Formát data a času

Pro konverzi datových formátů byla použita knihovna *Joda Time*, která silně rozšiřuje formáty data a času Java v1.7. Novější verze Java 1.8 už pracuje s daty a časy podobně jako tato knihovna, ale pro tento projekt byla použita verze starší. Další informace jsou na oficiálních webových stránkách projektu [16].

Pro základní zaznamenávání časů byl zvolen formát `yyyy-MM-dd HH:mm:ss.SSS` s přesností na milisekundy. Takto detailní formát byl vybrán pro záznam časově citlivých dat.

Kapitola 5

Testování

Tato kapitola se zaměřuje na zdokumentování způsobu a průběhu testování včetně předložení výsledků. Na závěr kapitoly bude výsledná aplikace porovnávána s existujícími řešeními.

5.1 Způsob testování

Pro otestování funkčnosti aplikace byly vytvořeny dva krátké testy použitelnosti, kdy byla data zaznamenávaná pomocí výstupní aplikace této práce.

Pro každý ze dvou testů byl vytvořen seznam úkolů pro participanta. Pomocí aplikace byla zaznamenávána data získávaná v průběhu testu. Cílem tohoto testování bylo ověřit použitelnost implementovaných modulů pro záznam a analýzu dat z testů. Právě díky tomu, že smysl testů je vytvořit pouze vhodné prostředí pro otestování aplikace, jsou vytvořené scénáře velmi krátké a není na jejich výsledky kladen velký důraz.

První scénář se zaměřuje na vytvoření seznamu úkolů pro otestování práce se seznamem aktualit na webových stránkách Fakulty elektrotechnické Českého vysokého učení technického v Praze. Jako testovací prostředí byl zvolen webový prohlížeč Mozilla Firefox s otevřenou jedinou kartou *https://www.google.cz*. Jednotlivé úkoly nejsou definovány přesněji kvůli potřebě získání více záznamů o interakci. Úkoly prvního scénáře jsou:

- Nalézt stránky fakulty
- Nalézt nejnovější zprávu o dění na fakultě
- Nalézt zprávu o florbalovém týmu této fakulty
- Vrátit se zpět na nejnovější zprávu

Příprava a počáteční nastavení pro druhý scénář je stejná jako pro první. Oproti předchozímu se však nezaměřuje na monitorování interakce s rozhraním webové stránky, ale webového prohlížeče Mozilla Firefox.

- Nalézt stránky fakulty
- Vytvořit záložku na tyto stránky v prohlížeči

- Vytvořit složku záložek a pojmenovat ji "School"
- Přesunout záložku fakulty do složky "School"

Obě tato testování byla zaměřena na ověření použitelnosti struktury projektů a předně modul *Záznam aktivit participanta*.

5.2 Průběh testování

Každý scénář byl testován se čtyřmi uživateli a pro každého uživatele byl vytvořen jeden log soubor a jedno video. Vzniklé záznamy aktivit jsou v textové podobě přílohou této práce.

Všechna testování probíhala stejně. Pro testování byly použity dva notebooky, jeden byl určen pro zprostředkování rozhraní participantovi, na druhém byla spuštěna aplikace pro záznam průběhu testu. Po započetí testu byl participant sledován a jeho interakce s rozhraním byly zaznamenávány.

5.3 Výsledky testování

Výsledky testů ukázaly, že implementované moduly jsou použitelné v reálné praxi.

Modul video, který je určen pro přehrávání videa by bylo vhodné doplnit o načítání dat z modulu záznam aktivit participanta a sjednotit časové osy. Implementovaná funkčnost však dostačuje základním požadavkům.

Záznam aktivit participanta narazil na drobný problém s nedostatečným popsáním začátku testu. Je vhodné, aby první zpráva zaznamenaná tímto modulem byla označená jako začátek testu. Pro konkrétnější potřeby výzkumných pracovníků je doporučené vytvořit vlastní modul pro záznam aktivit participanta, který bude lépe pracovat s časovými údaji. Jako významná vlastnost grafického rozhraní, která byla implementovaná v tomto modulu, je možnost uložení zprávy s časem aktuálním při uložení v případě ponechání prázdného pole pro datum a čas zprávy.

5.4 Srovnání s existujícím řešením

V porovnání se systémy Morae, Noldus či předchozí verzi IVE, je tato aplikace pouze na začátku svého vývoje a neobsahuje mnoho modulů pro práci s daty. Nevybavenost aplikace je hlavním problémem celého projektu, ale vzhledem k návrhu systému s ohledem na možnost snadné tvorby nových modulů je počet rozšiřujících modulů jen otázkou času.

Kapitola 6

Závěr

Cílem této diplomové práce bylo vytvoření víceuživatelské aplikace typu klient-server pro záznam a analýzu testů použitelnosti, která by umožňovala zaznamenávat data získaná během testování a následně je i umožňovala vyhodnocovat. Aplikace by měla umožňovat víceuživatelskou správu těchto dat a jejich vzájemné sdílení.

Pro správné pochopení potřeb výzkumných pracovníků věnujících se testování rozhraní, byla na začátku projektu provedena analýza pracovních procesů práce při testování uživatelského rozhraní. Díky tomu bylo možné navrhnout celý systém tak, aby byl pro výzkumné pracovníky dostatečně flexibilní, ale mohl jim maximálně umožnit jeho další rozvoj.

Během práce na tomto projektu jsem narazil na několik problémů. Prvním a jedním z největších problémů byla nedostatečná podpora dotazovacího jazyka Couchbase databáze. Řešením tohoto nedostatku bylo zavedení dokumentu, který obsahoval informace o všech souborech v daném bucketu. Toto řešení mělo smysl jen v případě, když nebylo možné zakomponovat dotazovací jazyk a mělo by být tímto jazykem nahrazeno tak rychle, jak to bude možné.

Dalším problémem byla synchronizace dat jednotlivých projektů. Řešením problému bylo využití nástroje *XDCR*, který je však přístupný pouze pomocí webového rozhraní Couchbase serveru. Pro pilotní nasazení výstupní aplikace lze použít i nástroje, které vyžadují více technických znalostí a nastavení, ale pro budoucí použití by bylo vhodné přenést i tuto funkcionalitu přímo do rozhraní aplikace.

Vzhledem k použití systému Couchbase bylo potřeba vyřešit i synchronizaci binárních či jiných souborů, které by neměly být součástí databáze Couchbase. Jedním z možných řešení bylo uložení těchto dat v binární podobě do dokumentů Couchbase, avšak kvůli malé flexibilitě a čistotě návrhu bylo od tohoto řešení nakonec odstoupeno. Pro synchronizaci souborů bylo implementováno souborové úložiště obsluhované skriptem v jazyce PHP, které dostačuje potřebám testovacího provozu. Do budoucna by však měla být struktura tohoto úložiště přepracována včetně obsluhujících algoritmů, které by hlavně měly být doplněny o bezpečnostní prvky.

Základní moduly, integrované v této aplikaci, byly podrobeny testům, které měly zjistit, jak dalece je nástroj vhodný k záznamu a analýze dat. Díky testování nebyly odhaleny nedostatky v implementovaných modulech, ale zároveň byla během testů odhalena potřeba dalších modulů, které uživatelům aplikace pomohou s analýzou získaných dat. Potřeba nových modulů byla během vývoje předpokládána a proto byla celá aplikace navržena tak,

aby byla tvorba nových modulů co nejsnazší a aby byla umožněna implementace externích modulů, které mohou být doinstalovány k základní verzi aplikace.

I přes výše zmíněné komplikace se podařilo hlavní cíle diplomové práce splnit a vytvořit tak rozšiřitelný nástroj, který je vhodným základem pro další vývoj víceuživatelské aplikace typu klient-server pro záznam a analýzu testů použitelnosti. Další vývoj by se měl zaměřit zejména na propracování integrační vrstvy a nasazení dotazovacích algoritmů pro vyhledávání v databázi Couchbase, s tím souvisí i úprava načítání souborové struktury aplikace. Velkým posunem bude přenesení funkcionalit, které zprostředkovává nástroj *XDCR* přes webové rozhraní přímo do aplikace, protože aplikace by měla sama obsahovat ovládání všech svých periferií. Dále pak v rozšíření základny obecných modulů pro větší flexibilitu aplikace a zároveň ještě více zjednodušit jejich vývoj. Vzhledem k právě probíhajícímu vývoji dotazovacího jazyka nad Couchbase je vhodné nahradit současný systém registrů tímto jazykem. Nahrazení tohoto jazyka umožní dynamičtější načítání struktury souborů, což vyústí v její snazší a méně náročné obnovení v případě změny. Jako poslední vylepšení by mělo být přepracování serveru pro práci s externími soubory, přidání autorizace a lepší přístup pro manipulaci se souborovým úložištěm.

Literatura

[1] H. Bock. *The Definitive Guide to Netbeans Platform 7.* Apress, 2012.

[2] S. Chacon. *Pro Git.* CZ.NIC, cz.nic edition, 2009.

[3] P. Herout. *Učebnice jazyka Java.* Kopp, třetí edition, 2014.

[4] S. Krug. *Don't make me think.* COMPUTER PRESS, druhé edition, 2000. česky.

[5] I. Malý. Analysis of usability tests with context model. Master's thesis, České vysoké učení technické v Praze, 2012.

[6] D. Ostrovsky and Y. Rodenski. *Pro Couchbase Server.* Apress, 2014.

[7] D. Salter and R. Dantas. *Netbeans IDE 8 Cookbook.* Packt Publishing, 2014.

[8] Couchbase.
Dostupné z: `http://couchbase.com/`, stav z 27. 4. 2015.

[9] Couchbase Admin Documentation.
Dostupné z: `http://docs.couchbase.com/admin/`, stav z 27. 4. 2015.

[10] Couchbase Developer Documentation.
Dostupné z: `http://docs.couchbase.com/developer/`, stav z 27. 4. 2015.

[11] Couchbase query language documentation.
Dostupné z: `http://docs.couchbase.com/developer/n1ql-dp4/n1ql-intro.html`, stav z 25. 3. 2015.

[12] CcouchDB.
Dostupné z: `http://couchdb.apache.org/`, stav z 13. 2. 2015.

[13] Eclipse rcp.
Dostupné z: `http://wiki.eclipse.org/Platform`, stav z 15. 11. 2014.

[14] Intellij rcp.
Dostupné z: `http://www.jetbrains.org/pages/viewpage.action?pageId=983889`, stav z 15. 11. 2014.

[15] IVE project webpage.
Dostupné z: `http://dcgi.felk.cvut.cz/home/malyi1/IVE/`, stav z 7. 5. 2015.

[16] Joda time project webpage.
Dostupné z: http://www.joda.org/joda-time/, stav z 27. 4. 2015.

[17] Morae product webpage.
Dostupné z: https://www.techsmith.com/morae.html, stav z 3. 4. 2015.

[18] MySQL.
Dostupné z: https://www.mysql.com/, stav z 13. 2. 2015.

[19] Netbeans rcp.
Dostupné z: https://netbeans.org/features/platform/, stav z 15. 11. 2014.

[20] Noldus product webpage.
Dostupné z: http://www.noldus.com/, stav z 3. 4. 2015.

[21] PHP.net.
Dostupné z: http://php.net/, stav z 15. 4. 2015.

[22] StackOverflow.
Dostupné z: http://stackoverflow.com/, stav z 27. 4. 2015.

[23] vlcj project webpage.
Dostupné z: http://www.capricasoftware.co.uk/#/projects/vlcj, stav z 27. 4. 2015.

Příloha A

Seznam použitých zkratek

GPS Globální polohovací systém

IDE Integrated Development Environment, Vývojové prostředí

JDK Java Development Kit, soubor nástroj pro vývoj pro platformu Java

JSON JavaScript Object Notation, způsob zápisu dat

JVM Java Virtual Machine

MPEG Rodina standardizovaných kompresních formátů

MVC Model-View-Controller, softwarová architektura

PC Osobní počítač

PHP Hypertext Preprocessor, programovací jazyk

RCP Rich Client Platform, soubor knihoven a add-onů pro vývoj

SDK Software Development Kit, soubor nástrojů pro vývoj

SVN Subversion, systém pro správů kódu

SWT Standart Widget Toolkit, knihovna grafických uživatelských prvků

UC Use Case, případ užití

UCD User Centered Design

UI User interface, uživatelské rozhraní

USD Americký dolar

UX User Experience, uživatelský prožitek

XDCR Cross Datacenter Replication, protokol pro výměnu dat

Příloha B

Seznam použitých pojmů

Bucket Konkrétní databáze v Couchbase

Cluster Ve významu Couchbase seskupení jednoho i více Couchbase serverů

Interface Rozhraní

Java Programovací jazyk

Java Virtual Machine Virtuální stroj ke spuštění programů v Javě

Map Reduce Programovací model pro zpracování velkých množin dat

Maven Nástroj pro správu a automatizaci buildů aplikací

MS Word, MS Excel Nástroje pro manipulaci s dokumenty firmy Microsoft

MySQL Druh relační databáze využívající dotazovací jazyk SQL

Mockup prototyp Ranný prototyp uživatelského rozhraní, zaměřuje se na interpretování akcí včetně textů

NoSQL Druh databáze, která nevyužívá tradiční tabulková schémata

Observer Člen týmu provádějící uživatelské testování, který má za úkol pozorovat průběh testu

Participant Účastník testu, jehož interakce s rozhraním je monitorována

PC, Mac Druhy počítačů

Repozitář Třída, která obaluje konkrétní data v databázi

Stakeholder Přizvaný pozorovatel testu

SVN, Git Systémy pro správu a verzování zdrojového kódu

Swing, SWT UI Knihovny grafických uživatelských prvků

SQL Jazyk pro tvorbu dotazů nad databází

Test použitelnosti Test, který má zjistit nakolik je dané rozhraní vhodné k použití

Use Case Případ užití, konkrétní atomický proces

User Centered Design Filozofie návrhu uživatelského rozhraní

User Experience Uživatelský prožitek

Příloha C

Obsah přiloženého CD

- `/`
  - `project` – *složka s projektem*
    - `app` – *složka s klientskou aplikací IVE*
      - `run` – *složka se spustitelnými soubory*
      - `src` – *složka se zdrojovými kódy*
    - `scripts` – *složka se dalšími skripty*
  - `.pdf` – *PDF verze tohoto textu psaná v LaTex*

Příloha D

Instalační a uživatelská příručka

Instalační a uživatelská příručka obsahuje návod, díky kterému se mohou budoucí uživatelé aplikace, jež je výstupem tohoto projektu, seznámit se základy práce s tímto systému.

D.1 Instalační příručka

Tato část příručky se zaměřuje na aktivity, které vedou ke správnému spuštění a nastavení aplikace před samotným použitím pro správu dat.

D.1.1 Couchbase

Postup instalace Couchbase je shodný jak klientském stroji tak i na vzdáleném serveru. Instalace Couchbase je popsána v dokumentaci Couchbase serveru *http://docs.couchbase.com/admin/admin/install-intro.html*. Pro pilotní implementaci výstupní aplikace, bylo použito základní nastavení databáze Couchbase popsané v návodech.

D.1.2 Spuštění aplikace

Pro spuštění aplikace je potřeba mít vytvořené spustitelné soubory. Pro případné vytvoření souborů je vhodné použít vývojové prostředí NetBeans.

D.1.3 Build aplikace v NetBeans

Po spuštění NetBeans IDE a otevření projektu IVE (název pracovní verze je IVEnext) je možné začít s generováním spustitelných souborů. Build bude zahájen po kliknutí pravým tlačítkem myši na balík *IVEnext-parent* a po výběru akce z kontextového menu *Build with Dependencies*. Tento postup zajistí build všech závislostí včetně jejich stažení k projektu. Výsledná aplikace bude vytvořena do složky */IVEnext/Application/target*. Spustitelné soubory budou vytvořeny do složky */IVEnext/Application/target/bin*.

D.1.4 Spuštění aplikace z Netbeans

V případě spouštění aplikace s NetBeans je možné kliknutím pravým tlačítkem na balík *IVEnext-app* a výběru příkazu *Run* z kontextového menu. Tímto postupem se během několika sekund až minut spustí celý program.

D.1.5 Spuštění aplikace bez Netbeans

Spustitelné soubory jsou ve složce */IVEnext/Application/target/bin*. Pro spuštění aplikace stačí vybrat a spustit příslušný soubor. Soubory s koncovkou *.exe* jsou určeny pro systém Windows, soubor bez koncovky pro systémy unixového typu. Tímto postupem se během několika sekund až minut spustí celý program.

D.1.6 Konfigurační soubor

Konfigurační soubor slouží předně k nastavení adresy Couchbase serveru pro klientskou aplikaci. V základním nastavení není potřeba tento konfigurační soubor vytvářen. Smysl a postup práce s konfiguračními soubory je rozebrán v kapitolách 4.1.1.2 a 4.6.1. Tento konfigurační soubor je vygenerován, ve stejné složce jako spouštěný program, při každém startu aplikace kdy není detekován. Pokud tomu tak není lze ho vytvořit ručně a pojmenovat *servers.config*.

D.2 Uživatelská příručka

Uživatelská část příručky má za úkol seznámit budoucího uživatele se základy práce s výstupní aplikací.

D.2.1 Správa projektů

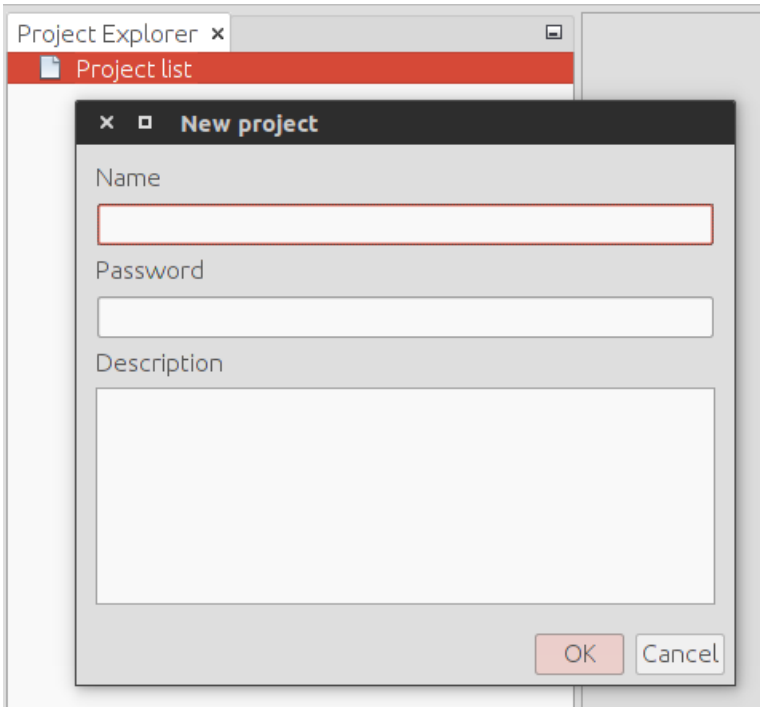
V rámci této aplikace je kvůli omezením databáze umožněno projekty jen vytvářet nebo mazat. Veškeré změny prováděné s projekty jsou zprostředkovány přes rozhraní ProjectExploreru, který vykresluje obsah všech přidaných projektů.

D.2.1.1 Vytvoření

Pro vytvoření nového projektu je potřeba pravým kliknutím na *Project list* aktivovat kontextové menu a poté vybrat možnost *New Project*. Tím bude aktivováno vyskakovací okno (viz Obrázek D.1), které umožní zadat potřebné údaje pro vytvoření projektu.

D.2.1.2 Smazání

Mazání projektu je prováděno opět pravým kliknutím na příslušnou položku projektu a výběrem *Delete Project* z kontextového menu.



Obrázek D.1: Vyskakovací okno obsahující formulář pro vytvoření nového projektu

D.2.2 Přihlášení

Během úprav týkajících se projektů, může dojít k žádosti o přihlášení uživatele (viz Obrázek 4.8). Poté bude potřeba vložit přístupové údaje k administrátorskému účtu.

D.2.3 Správa participantů

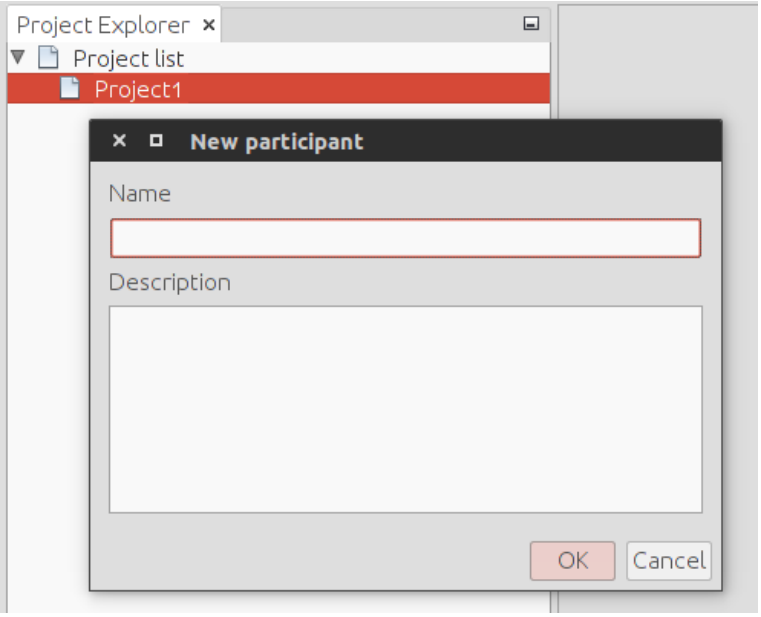
Správa participantů je realizována pomocí komponenty *ProjectExplorer*. Participanty lze vytvářen, upravovat i mazat.

D.2.3.1 Vytvoření

Pro vytvoření nového participanta je potřeba pravým kliknutím na cílový projekt aktivovat kontextové menu a poté vybrat možnost *New participant*. Poté bude aktivováno vyskakovací okno (viz Obrázek D.2), které umožní zadat potřebné údaje pro vytvoření participanta.

D.2.3.2 Úprava

Pro úpravu konkrétního participanta lze využít kontextovou nabídku vyvolanou pravým kliknutím na položku participanta a výběru *Edit participant*.



Obrázek D.2: Vyskakovací okno obsahující formulář pro vytvoření nového participanta

D.2.3.3 Smazání

Pro smazání konkrétního participanta lze využít kontextovou nabídku vyvolanou pravým kliknutím na položku participanta a výběru *Delete participant*.

D.2.4 Správa dokumentů

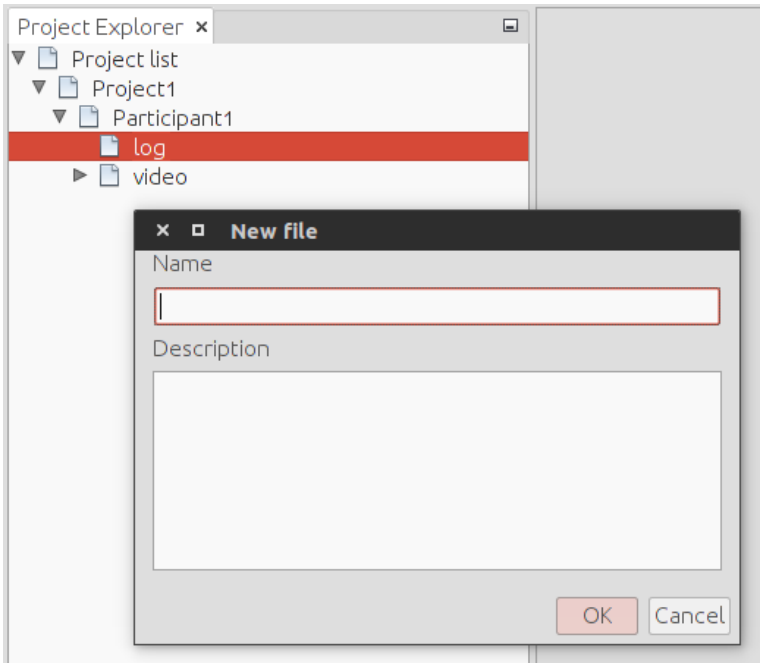
Správa souborů je realizována pomocí komponenty *ProjectExplorer*. Participanty lze vytvářen, měnit, mazat a otevírat v výchozím nástroji. V rámci aplikace existuje několik typů souborů rozdělených do kategorií ve stromové struktuře *ProjectExploreru*.

D.2.4.1 Vytvoření

Pro vytvoření nového souboru je potřeba pravým kliknutím na cílový typ souboru (zanoření v ProjectExploreru následující po konkrétním participantovi) aktivovat kontextové menu a poté vybrat možnost *New File*. Poté bude aktivováno vyskakovací okno (viz Obrázek D.3, které umožní zadat potřebné údaje pro vytvoření souboru.

D.2.4.2 Úprava

Pro úpravu konkrétního souboru lze využít kontextovou nabídku vyvolanou pravým kliknutím na položku souboru a výběru *Edit file*.



Obrázek D.3: Základní vyskakovací okno obsahující formulář pro vytvoření nového souboru

D.2.4.3 Smazání

Pro smazání konkrétního souboru lze využít kontextovou nabídku vyvolanou pravým kliknutím na položku souboru a výběru *Delete file*.

D.2.4.4 Otevření v modulu

Pro otevření konkrétního souboru v modulu je nutné dvakrát kliknout na položku souboru v *ProjectExploreru*.

D.2.5 Moduly

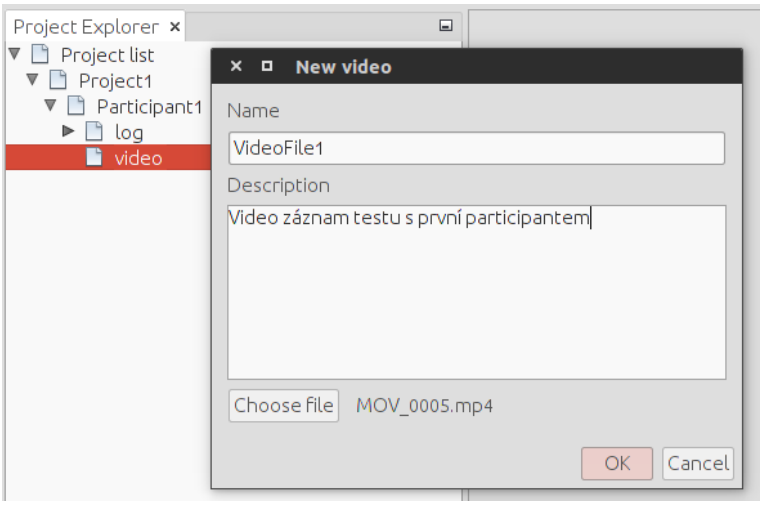
V základní verzi aplikace byly implementovány dva základní moduly pro práci s daty. Záznam aktivit uživatele a přehrávač videí.

D.2.5.1 Záznam aktivit uživatele

Nástroj pro záznam aktivit uživatele je velmi jednoduchý a obsahuje pouze přehled dříve přidaných aktivit a ovládací prvky pro přidání dalších záznamů (viz Obrázek 4.6. Pro zadání zprávy slouží druhé největší pole nacházející se ve spodní části této obrazovky. Pomocí tlačítka *Add message* je záznam přidán a v případě stisku tlačítka *Truncate message* je zpráva vymazána. Užitečná je funkce pro zadání času. V případě, že je pole pro datum a čas vyplněno, automaticky systém počítá se vstupním formátem `yyyy-MM-dd HH:mm:ss.SSS`, tak toto datum uloží, ale pokud je pole ponecháno nevyplněné, systém bere aktuální čas v době uložení zprávy.

D.2.5.2 Video přehrávač

Vytvoření souboru je oproti obecnému procesu změněné. U dokumentů popisující video je další nutný prvek soubor videa. Změněný formulář pro vytvoření dokumentu typu video je na Obrázku D.4.



Obrázek D.4: Vytvoření dokumentu popisujícího video je odlišné od obdobného procesu pro záznam aktivit uživatele. Je nutné krom popisu souboru ještě přiložit vlastní video soubor, který bude přehráván.

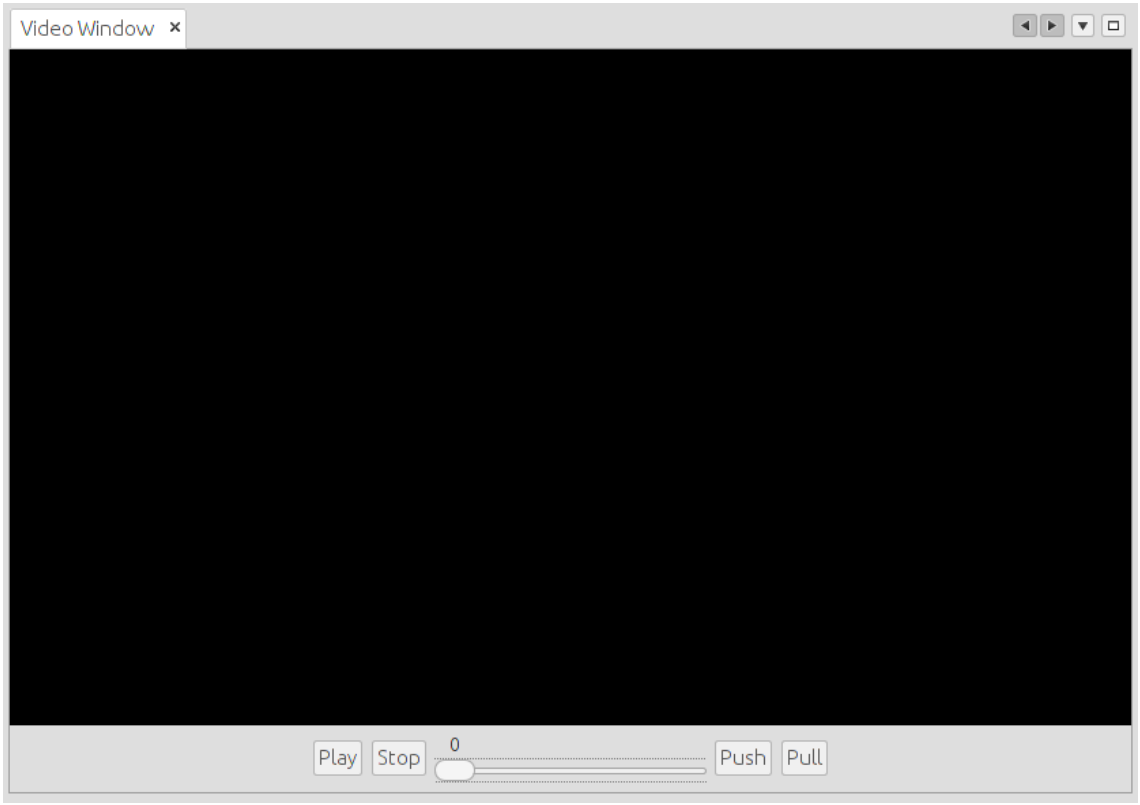
Nástroj pro přehrávání videa je velmi jednoduchý a skládá se jen z několika základních ovládacích prvků pro spuštění, pozastavení a ukončení přehrávání videa. V celém nástroji je hlavním prvkem část určená pro přehrávání videa (viz Obrázek D.5).

D.2.6 Synchronizace

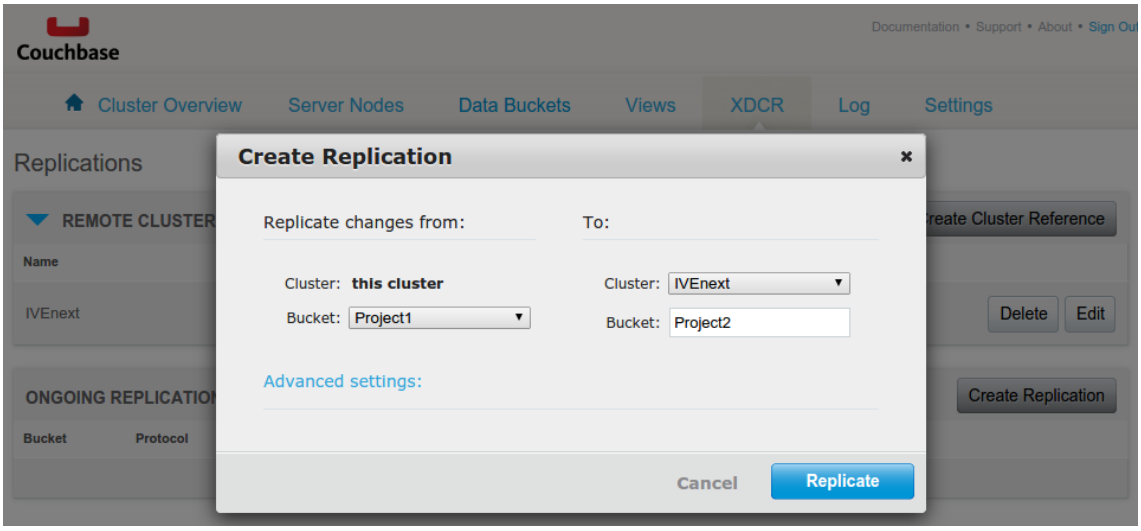
Synchronizace je z velké části automatická a využívá funkce replikace obsahu z Couchbase pomocí nástroje *XDCR*. V případě, kdy je potřeba synchronizovat velké datové soubory například video, je tato funkce implementována v rámci konkrétního nástroje.

D.2.6.1 Couchbase

Pro nastavení synchronizaci je určena funkce replikace dat mezi Couchbase servery. Správu replikací obstarává sám Couchbase server pomocí nástroje *XDCR*, který je přístupný pouze přes webové rozhraní serveru. Mezi vzdáleným serverem a lokálním je nutné nejprve na vzdáleném serveru zajistit v konkrétním bucketu povolení replikace. Poté je nutné vytvořit referenci z lokálního serveru na vzdálený, k této akci je potřeba znát přístupové údaje ke vzdálenému serveru. Poslední úkon v celém nastavení je vytvořit replikační záznam pro konkrétní buckety. To lze provézt v rozhraní lokálního serveru pomocí stejného nástroje pod tlačítkem *Create Replication*. Tato akce vyvolá formulář (viz Obrázek D.6) pro nastavení replikace mezi dvěma buckety.



Obrázek D.5: Přehrávání videa je umožněno díky jednoduchému přehrávači, který umožňuje přehrávat/pozastavit video (Play/Pause), nebo přehrávání ukončit (Stop). Pro nastavení konkrétního času videa slouží posuvník.



Obrázek D.6: Příklad nastavení replikace pro konkrétní buckety

D.2.6.2 Externí soubory

V základní aplikaci je synchronizace implementována přímo v modulu Video přehrávač pomocí tlačítek pro stažení *Pull* nebo nahrání *Push* souboru na server. Rozhraní je zobrazeno na Obrázku D.5. V případě ukončení akce se změní text *Pull* na *Pulled* nebo *Push* na *Pushed*.

Příloha E

Data získaná při ověřování použitelnosti aplikace

Tato příloha obsahuje soubory získané během ověřování použitelnosti aplikace.

E.1 První scénář - testování webového rozhraní

- Nalézt stránky fakulty
- Nalézt nejnovější zprávu o dění na fakultě
- Nalézt zprávu o florbalovém týmu této fakulty
- Vrátit se zpět na nejnovější zprávu

E.1.1 Participant 1

2015-05-01 15:32:34.744 Začátek testu

2015-05-01 15:33:11.546 Participant se zorientovává v aplikaci

2015-05-01 15:33:34.325 Participant zadal url fel.cvut.cz do adresní řádky

2015-05-01 15:34:05.895 Participant nalezl nejnovější zprávu o dění na fakultě

2015-05-01 15:34:32.126 Participant nalezl zprávu o florbalovém týmu v archivu

2015-05-01 15:34:56.857 Participant se vrátil zpátky na hlavní stránku fakulty pomocí drobečkové navigace

2015-05-01 15:35:05.642 Konec testu

E.1.2 Participant 2

2015-05-01 15:41:15.134 Začátek testu

2015-05-01 15:41:23.813 Participant se zorientovává v aplikaci

2015-05-01 15:42:12.434 Participant zadal url seznam.cz do vyhledávače Google

2015-05-01 15:42:45.525 Participant pomocí vyhledávače Seznam nalezl stránky fakulty

2015-05-01 15:43:27.402 Participant má problémy zorientovat se na stránce

2015-05-01 15:44:41.941 Participant nalezl seznam aktualit, ale není si jistý zda je to opravdu to co hledá

2015-05-01 15:44:55.520 Participant identifikoval nejnovější zprávu o dění na fakultě

2015-05-01 15:46:48.396 Participant na stránce hledá starší články

2015-05-01 15:47:32.757 Participant otevřel archiv, ale musel být upozorněn moderátorem, že postupuje správně

2015-05-01 15:48:35.283 Participant nalezl konkrétní článek o týmu

2015-05-01 15:48:57.469 Participant se vrátil na hlavní stránku pomocí navigace prohlížeče

2015-05-01 15:48:59.378 Konec testu

E.1.3 Participant 3

2015-05-01 16:05:06.122 Začátek testu

2015-05-01 16:05:12.890 Participant se zorientovává v aplikaci

2015-05-01 16:05:52.258 Participant vyhledal stránky pomocí Google

2015-05-01 16:06:05.463 Participant si spletl aktuality a akce, ale hned poté nalezl správný článek

2015-05-01 16:06:32.386 Participant využil archiv aktualit k nalezení článku o florbalovém týmu

2015-05-01 16:07:56.924 Participant se vrátil na hlavní stránku úpravou url adresy

2015-05-01 16:07:59.278 Konec testu

E.1.4 Participant 4

2015-05-01 16:15:34.449 Začátek testu

2015-05-01 16:15:40.921 Participant se zorientovává v aplikaci

2015-05-01 16:16:01.782 Participant zadal url fel.cvut.cz do adresní řádky

2015-05-01 16:17:19.115 Participant má problémy zorientovat se na stránce

2015-05-01 16:18:28.080 Participant nalezl nejnovější zprávu o dění na fakultě

2015-05-01 16:19:27.506 Participant s problémy nalezl archiv aktualit

2015-05-01 16:19:45.254 Participant využil k nalezení článku v archivu aktualit nástroj "hledání na stránce"prohlížeče

2015-05-01 16:20:18.867 Participant se chtěl vrátil zpátky na hlavní stránku fakulty pomocí kliku na logo ČVUT, byl zmatený při načtení webových stránek cvut.cz, poté načetl stránky úpravou url adresy

2015-05-01 16:20:23.653 Konec testu

E.2 Druhý scénář - testování rozhraní prohlížeče

- Nalézt stránky fakulty
- Vytvořit záložku na tyto stránky v prohlížeči
- Vytvořit složku záložek a pojmenovat ji "School"
- Přesunout záložku fakulty do složky "School"

E.2.1 Participant 5

2015-05-01 16:30:06.442 Začátek testu

2015-05-01 16:30:12.538 Participant se zorientovává v aplikaci

2015-05-01 16:31:52.926 Participant nalezl stránky pomocí url

2015-05-01 16:32:05.834 Participant vytvořil záložku pomocí nástroje Záložky -> Přidat stránku do záložek a vložil záložku do lišty záložek

2015-05-01 16:32:32.848 Participant vytvořil složku "School"pravým klikem v liště záložek

2015-05-01 16:32:45.355 Participant přetáhl záložku fakulty do složky "School"

2015-05-01 16:32:52.754 Konec testu

E.2.2 Participant 6

2015-05-01 16:39:30.387 Začátek testu

2015-05-01 16:39:58.146 Participant se zorientovává v aplikaci

2015-05-01 16:40:05.893 Participant nalezl stránky pomocí Google

2015-05-01 16:40:24.968 Participant vytvořil záložku pomocí pravého kliku na liště záložek a definoval přímo adresu i název záložky

2015-05-01 16:40:36.230 Participant vytvořil složku "School"pravým klikem v liště záložek

2015-05-01 16:40:42.582 Participant přetáhl záložku fakulty do složky "School"

2015-05-01 16:40:46.263 Konec testu

E.2.3 Participant 7

2015-05-01 16:47:25.563 Začátek testu

2015-05-01 16:47:31.721 Participant se zorientovává v aplikaci

2015-05-01 16:47:43.346 Participant nalezl stránky pomocí Google

2015-05-01 16:48:07.079 Participant vytvořil záložku pomocí pravého kliku na liště záložek a definoval přímo adresu i název záložky

2015-05-01 16:48:29.812 Participant vytvořit složku "School"pravým klikem v liště záložek

2015-05-01 16:48:42.062 Participant přetáhl záložku fakulty do složky "School"

2015-05-01 16:48:48.553 Konec testu

E.2.4 Participant 8

2015-05-01 16:55:20.810 Začátek testu

2015-05-01 16:55:28.432 Participant se zorientovává v aplikaci

2015-05-01 16:55:47.463 Participant nalezl stránky pomocí url

2015-05-01 16:55:57.832 Participant vytvořil záložku pomocí pravého kliku na liště záložek a definoval přímo adresu i název záložky

2015-05-01 16:56:02.617 Participant vytvořil složku "School"pravým klikem v liště záložek

2015-05-01 16:56:16.554 Participant přetáhl záložku fakulty do složky "School"

2015-05-01 16:56:20.769 Konec testu