Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Rozšíření skriptovacích možností programu EasyArchiv

Plzeň 2018 Bc. Patrik Kořínek

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 16. května 2018

Bc. Patrik Kořínek

Abstract

This work aims to enhance the EasyArchiv system in order to enable execution of user generated code in client environments. The enhancement should allow improved user interaction by creating an option to add client scripts to existing programs. Client scripts should be able to interact with the system and display user dialogs. Implementation of the solution began after analysis of both the EasyArchiv system and all scripting languages that could be used for this enhancement. The result of this work allows the creation of JavaScript scripts that run in the environment of the client applications.

Abstrakt

Tato práce si klade za cíl rozšíření systému pro správu dokumentů EasyArchiv o možnost spouštění uživatelského kódu v klientských aplikacích. Toto rozšíření by mělo umožnit lepší interakci s uživateli systému pomocí přidání klientského skriptování k existujícím programům systému. Skriptování by mělo umožňovat interakci se systémem a vizualizaci uživatelských dialogů. Způsob implementace řešení byl vybrán po analýze systému EasyArchiv a skriptovacích jazyků, které by potencionálně mohly být k tomuto rozšíření použité. Výsledné řešení umožňuje vytváření JavaScript skriptů spouštěných v prostředí klientských aplikací.

Obsah

1 Úvod

Tato diplomová práce si klade za cíl rozšíření aplikací systému EasyArchiv o možnost spouštění uživatelských skriptů v prostředí všech klientských aplikací, ze kterých se systém skládá. Jednou z podmínek práce je rozšíření programů takovým způsobem, který minimálně ovlivní současné struktury a procesy. Současná funkcionalita systému nesmí být v žádném případě narušena.

Hlavním zaměřením těchto skriptů bude uživatelská interakce. Systém EasyArchiv má v současnosti možnost spouštět uživatelské skripty, ale pouze v rámci jádra systému (spouští se na serveru), takže nemohou interagovat s uživatelem na klientské straně. Klientské dialogy by poskytly rozšířenou možnost interakce s uživatelem za účelem získání doplňujících informací, ulehčení obsluhy a širší možnosti personalizace způsobu využívání systému. Poskytly by také jednoduchý způsob, jak uživatelům umožnit vytváření šablon, podle kterých by bylo možné předvyplnit opakující se nebo těžko dostupné informace.

Obsah jednotlivých kapitol je následující:

- EasyArchiv – popisuje systém, jeho účel a strukturu. Také se zaobírá rozborem jednotlivých programů, ze kterých se systém skládá.
- Použité technologie – kapitola obsahuje výčet a popis některých důležitých technologií použitých v systému EasyArchiv.
- Skriptování – souhrn požadavků na skriptovací část rozšíření a výčet zvažovaných technologií spolu s jejich silnými a slabými stránkami.
- Navržené řešení – popisuje návrh úprav pro jednotlivé části systému a omezení, s kterými budou implementovány.
- Implementace – popis průběhu implementace pro jednotlivé komponenty systému.
- Testování – kapitola zabývající se ověřováním funkčnosti řešení.
- Závěr – zhodnocení výsledků této práce.

2 EasyArchiv

„EasyArchiv/EasyPLM® [je] nejen komplexní systém správy dokumentace (Dokument Management Systém, DMS) založený na WORKFLOW, ale rovněž systém správy životního cyklu výrobků (Produkt Lifecycle management, PLM). Přes svůj název se ve skutečnosti jedná o sofistikovaný produkt, umožňující snadné ovládání a implementaci.“[1]

Obrázek 2.1: Logo EasyArchiv

Systém EasyArchiv je certifikován dle normy ISO 16363:12 - Důvěryhodná (Digitální) Úložiště.

2.1 Systém pro správu dokumentů

Systém pro správu dokumentů (Document Management System, DMS) je počítačový systém určený ke sledování, správě a uchování elektronických dokumentů. Obvykle umožňuje uchovat dokumenty v centrálním uzlu a poskytnout k nim přístup všem uživatelům s příslušným oprávněním. Jedná se tedy o systém sdílení dokumentů s rozšířenou funkcionalitou. Díky centralizaci umožňuje zálohu všech firemních dokumentů. Společnost tak může jednoduše ovládat přístup k účetnictví, digitálním zálohám tiskopisů a jakýmkoli elektronickým datům, se kterými pracuje, a zajistit, že nebudou ztraceny, poškozeny nebo upraveny neoprávněnou osobou.

Kromě zálohování je dalším bezpečnostním prvkem verzování dokumentů. DMS udržuje záznamy o změnách dokumentu a o tom, kteří uživatelé k nim přistoupili. Lze tak identifikovat člověka zodpovědného za všechny změny a případně dokument navrátit do dřívější podoby.

Některé společnosti a organizace mají státem nařízenou elektronickou správu dokumentů, aby bylo možné udržovat dohled nad jejich činností. Jedná se například o společnosti zabývající se účetnictvím, výrobou medicínských materiálů nebo pracující s citlivými informacemi. DMS se však hodí jakékoli firmě, která chce zlepšit správu dokumentů napříč svým informačním

systémem, zvýšit bezpečnost a zlehčit komunikaci mezi zaměstnanci.

Některé DMS (například EasyArchiv) mohou být navíc integrované do jiných softwarů pomocí různých rozšíření nebo add-onů. DMS tak může získat více informací o dokumentu a přidat je jako součást metadat dokumentu. Nároky na uživatele se nadále sníží tím, že DMS je součástí programu, který právě používá.

Obvyklé schopnosti DMS:

- Začleňování dokumentů – vkládání souborů a správa metadat dokumentu.
- Integrace – propojení s programem, ve kterém se dokumentace vytváří nebo zpracovává.
- Verzování dokumentu – správa změn.
- Přístupová práva – zajištění, že s dokumentem operují jen oprávněné osoby.
- Dostupnost – přístup k dokumentu z jakékoli stanice s DMS.
- Vyhledatelnost – vyhledávání dokumentu podle metadat (která mohou obsahovat identifikátor, informace o/z dokumentu nebo jakákoli jiná data relevantní k dokumentu).

2.2 Správa životního cyklu produktu

Správa životního cyklu produktu (Product Lifecycle Management, PLM) je systém procesů zaměřených na snížení ceny nasazení produktu na trh, vytvoření poptávky, prodloužení doby, po kterou je produkt ve fázi zralosti a udržení co nejvyšších zisků během období, kdy prodeje začnou klesat. Efektivní PLM záleží na efektivní spolupráci mezi dodavateli a umožňuje sdílení dat napříč systémy návrhu, zajištění kvality, výroby a prodeje produktu.

PLM se tedy stará o správu veškerých podkladů pro návrh produktu, jeho výrobu a údržbu a slouží jako úložiště všech dokumentů, poznámek, reakcí, katalogů a informací o něm.

2.3 Workflow

Workflow (pracovní postup, technologický postup) je plánovaný a opakovatelný postup rozložený na jednotlivé kroky a sekvenční vazby. Jedná se o organizovaný řetězec operací vedoucí k cíli. Může jít o vytváření dokumentu, poskytování služby nebo výrobu produktu, popsanou krok po kroku v přehledném a jednoduše pochopitelném formátu.

Workflow často existuje jako součást existujících systémů DMS, ve kterém je oběh dokumentů řízen definicí pracovního postupu. S dokumentem je spřaženo, kdo má vykonat jaké podprocesy a jak hlídat jejich splnění, jak se má dokument chovat po jejich splnění a cíl celého procesu.

K řízení pracovního postupu je zapotřebí nadefinovat pravidla řídící procesy, dokumenty, se kterými se předávají, a metriky, sloužící k posuzování jejich splnění.

2.4 Funkce EasyArchiv

EasyArchiv je systém umožňující sledování, správu a uchování obecné a technické dokumentace v rámci firemního prostředí. Jeho silnou stránkou je možnost definice vlastních typů dokumentace v souladu s firemními normami a ustanoveními. Dokumentace vytvořená podle této vlastní definice pak poskytuje mnoho funkcí, které jsou uživatelé zvyklí využívat v rámci operačního systému, jako například řízení přístupu k souborům přidělováním přístupových práv, úpravy vlastností dokumentu nebo prohlížení souborů asociovaných s dokumentem. EasyArchiv nadále poskytuje další funkce, které umožňují práci s dokumentem:

- Rozšířené vyhledávání dle libovolného atributu v definici dokumentace a fulltextové vyhledávání.
- Organizování dokumentů do složek podle uživatelského výrazu (SQL příkaz).
- Uzamykání dokumentů, aby se zabránilo souběžné editaci dvěma různými uživateli najednou.
- Možnost manuálního Check In/Check Out (viz níže).

- Kompletní historie úprav dokumentu.
- Import a Export do různých formátů.
- Správa změnového a schvalovacího řízení.
- Dostupné jsou i mnohé integrace pro jiné programy. EasyArchiv tak může přímo přistoupit k dokumentům z programů jako například MS Office, Inventor, SolidWorks nebo e-mailový klient a spravovat je jako dokumentaci.

Součástí EasyArchivu je také možnost vytváření uživatelských skriptů v jazyce C#, které pak lze propojit s mnoha různými prvky a událostmi v systému. Jako reakce na uživatelské akce se pak tyto skripty spustí a poskytují tak rozšířenou funkcionalitu, která umožňuje upravit systém uživateli na míru.

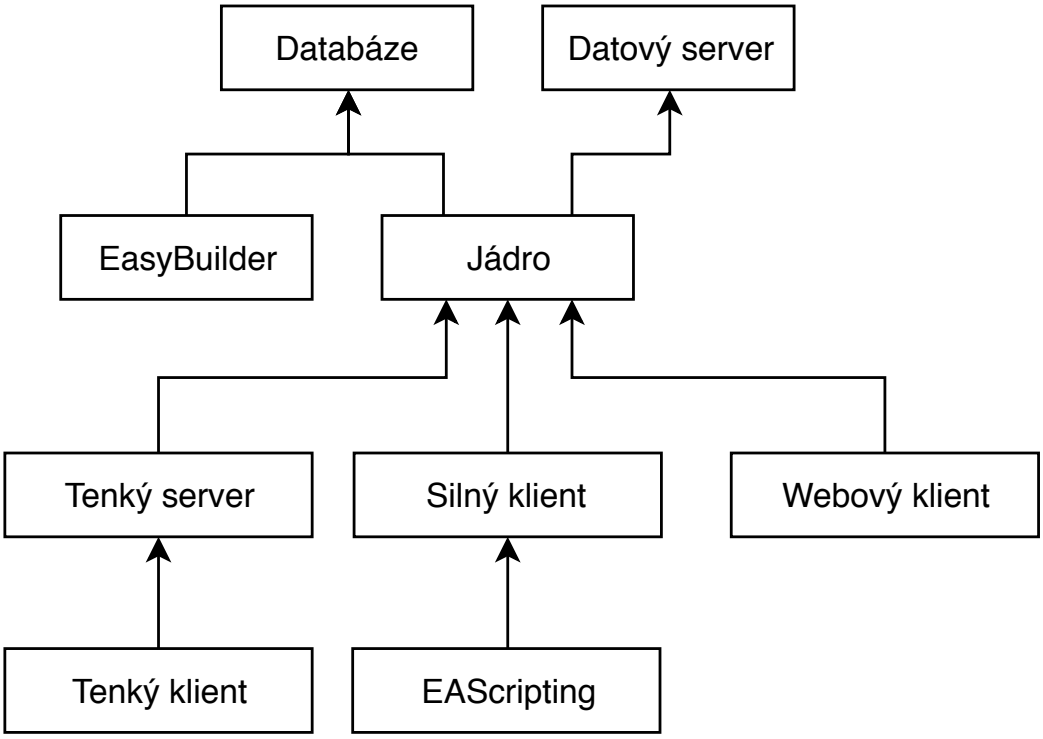
Tyto skripty umožňují například získávání aktuálních dat ze serveru, mohou uživatele informovat o typech dokumentů, ke kterým má přístup, o uživatelích, se kterými může sdílet dokumenty, nebo formovat SQL dotazy pro filtrování dokumentů dle uživatelských parametrů.

Současné skriptovací řešení je však součástí jádra (více v kapitole 2.6). Tato skutečnost umožňuje jednoduchý přístup k prakticky všem funkcím a informacím v jádře, ke kterým má konkrétní uživatel přístup, ale zabraňuje uživatelské interakci. Veškeré dialogy se otevírají na serveru, takže uživatel může interagovat pouze s ovládacími prvky, které byly vytvořeny jako součást prostředí EasyArchivu nebo součást formuláře dokumentu.

Cílem této práce je proto navrhnout řešení, které by umožňovalo vytváření a spouštění skriptů v klientském prostředí za účelem interakce s uživatelem. Klientské skripty by měly mít schopnost vytvářet dialogy, reagovat na uživatelské akce a interagovat s formuláři a prvky, které je vyvolaly.

2.5 Struktura systému EasyArchiv

Systém se skládá z několika programů. Tři z nich (silný, tenký a webový klient) jsou klienti poskytující stejnou funkcionalitu s různým způsobem přístupu. EasyBuilder je administrativní nástroj pro práci se systémem.



Obrázek 2.2: Vztahy jednotlivých prvků systému EasyArchiv

- Jádro je program EasyCom.
- Silný klient je program CGExplorer.
- Tenký server je program EAWebServices.
- Tenký klient je program EALightClient.
- Webový klient je program EAWebClient.
- EAScripting je skriptovací knihovna EAScripting.
- EasyBuilder je program EasyBuilder.

2.6 Silný klient (a jádro)

Silný klient je program vytvořený v jazyce C++, který umožňuje uživatelský přístup ke všem funkcím systému EasyArchiv, jak jsou popsané v kapitole

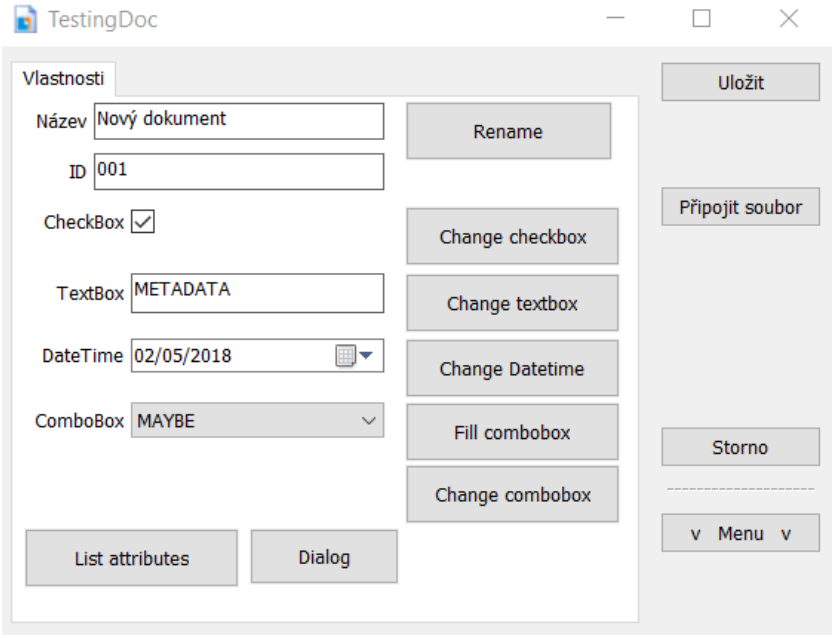
2.4. Na rozdíl od tenkého a webového klienta (kteří jsou popsáni v kapitolách 2.7 a 2.8) není silný klient aplikace typu klient + server. Skládá se z úzce propojeného jádra systému a uživatelského rozhraní.

Výhodou tohoto řešení je vyšší rychlost aplikace v případě, kdy je datový server a databáze systému EasyArchiv rychle dostupná – například na lokální firemní síti. Silný klient je vytvořen v jazyce C++ a je tak výkonnější než jeho tenký a webový protějšci. Díky tomu, že má přímý přístup k databázi a nemusí čekat na odpověď serveru, který by mu musel informace zdlouhavě zpracovávat a předávat, je silný klient ve většině situací schopný rychlejší reakce než ostatní klienti.

Některé uživatelsky významné funkce:

- Přihlášení uživatelů – ověří uživatelské údaje oproti údajům v databázi.
- Vytváření a správa dokumentů – uživatel vyplní předdefinovaný formulář a poskytne soubor, který bude uložen jako dokument. Informace ve formuláři se stávají metadaty a umožňují tak pokročilou práci se souborem v rámci systému. Po uložení se soubor nahraje na souborový server a metadata do databáze.
- CheckIn dokumentu – pokud si uživatel přeje pracovat s dokumentem, ke kterému má přístup, provede tzv. CheckIn. Soubor se uloží do pracovního adresáře a uživatel s ním nadále může pracovat, jak si přeje.
- CheckOut dokumentu – když uživatel dokončí svou práci na dokumentu, provede CheckOut. Dokument se nahraje zpět na datový server, ale s vyšším číslem verze. Detaily o změnách v dokumentu se uloží do databáze.
- Prohlížení dokumentů – uživatel může prohlížet, vyhledávat a filtrovat dokumenty, ke kterým má přístup.

Z hlediska této práce je nejdůležitější funkce vytváření a správa dokumentů. Uživatelem definované formuláře, do kterých se vyplňují metadata dokumentu, jsou funkcí, která nejvíce získá z obohacení skriptovacích možností o uživatelskou interaktivitu.



Obrázek 2.3: Vytváření dokumentu v systému EasyArchiv

2.6.1 Jádro a skriptování

Jádro EasyCom obstarává všechny funkce, které jsou společné pro tři klienty. Od přihlašování uživatelů, ověřování jejich identit a udržování jejich relace (session), až po získávání dat z databáze, jejich formátování a předávání implementaci serveru.

Skriptování v jádře je implementováno pomocí C# knihovny, která je obalena C++ knihovnou (více v kapitole 3.3). Jádro tak získá od klienta podrobnosti o potencionálním volání skriptu, data o tom, jaké skripty lze z aktivního formuláře volat, ověří, zda prvek, se kterým uživatel interagoval, je propojen se skriptem a pokud ano, tak kód skriptu získá z databáze a předá ho pomocí obalové knihovny skriptovací knihovně. Ta kód přeloží, pokud je to zapotřebí, a spustí ho. Větší podrobnosti o technologii lze najít v kapitole 3.3.

Díky tomu, že jsou všechny skripty spouštěny přímo z jádra, mají přístup k prakticky jakékoli funkci, která se v jádře nachází. Z bezpečnostních důvodů jsou jejich možnosti omezené, ale stále mají hodně svobody, co se týká získávání dat z databáze, informací o uživatelích a o ostatních dokumentech. Více o možnostech a omezení skriptování v jádře EasyCom se lze dočíst

v kapitole 3.2.

2.7 Tenký klient

Tenký klient je novější verze EasyArchiv klienta vytvořená v jazyce C#. Je postaven na architektuře klient + server, takže se praktiky vzato jedná o dva programy: klient a server. Ke komunikaci mezi nimi využívá technologii WebService více popsanou v kapitole 3.4

Tento typ klienta má největší výhodu v prostředí, kde databáze a datový server nejsou rychle dostupné. Serverová aplikace má rozšířené možnosti cachování oproti silnému klientovi. V moment, kdy jakýkoli uživatel požaduje přístup k dokumentu, server si dokument a jeho metadata stáhne a až poté jej distribuuje uživateli. Tato schopnost se netýká jen dokumentů, ale také skriptů. Aplikace tak nemusí zdlouhavě kontaktovat databázi a datový server vždy, když chce otevřít soubor nebo spustit skript.

Serverová aplikace tenkého klienta využívá stejné jádro jako silný klient. C++ knihovna EasyCom se spouští spolu se serverem pomocí COM rozhraní (více v kapitole 3.9). Jádro samotné obstarává veškerou komunikaci s databází a datovým serverem, práci s daty a cachování. Server tenkého klienta primárně obstarává komunikaci mezi sebou a klientem.

Zobrazení formuláře pro vytvoření dokumentu probíhá tak, že se na požadavek serveru formulář inicializuje v jádře EasyCOM v čistě datové podobě (alokované objekty, ale žádné prvky, s kterými by bylo možné interagovat). Tento formulář se převede do XML podoby, která se předá serveru. Ten ji přepošle klientovi, který XML dokument zpracuje a vytvoří podle něj formulář pomocí ovládacích prvků z knihovny System.Windows.Forms. Uživatelské akce jako stisknutí tlačítka ve formuláři nebo změna hodnoty textového pole jsou přes server poslány do jádra, které odpoví XML dokumentem popisujícím změnu ve formuláři.

Na rozdíl od silného klienta, tenký klient není jakkoli připraven na spouštění skriptů v klientském prostředí. Současné řešení pošle informaci o uživatelské akci, například o stisknutí tlačítka, jádru, které vykoná skripty, které jsou s tlačítkem svázané a pošle klientovi informaci, jak se má formulář změnit. Na všechny prvky formuláře, které jsou schopné spouštění současných skriptů jsou však navěšeny funkce, které reagují na uživatelské akce a infor-

mují o nich jádro. Za předpokladu, že tenký klient dostane při inicializaci formuláře také informace o klientských skriptech a prvcích, na které jsou navázány, bylo by možné rozšířit současný systém reagující na uživatelské akce o obsluhu uživatelských skriptů.

Tenký klient definuje první omezení klientského skriptování. Jelikož klient samotný nemá přístup k jádru systému, skripty spuštěné v jeho prostředí k němu také přístup mít nebudou. Z tohoto důvodu přijdou o možnost přistupovat k jakýmkoli datům, která nejsou součástí formuláře v moment, kdy je odeslán z jádra. Klientské skripty budou potřebovat možnost dynamického získávání dat, schopnost doručit požadovaná data na klientskou aplikaci spolu se skriptem, nebo se budou muset smířit pouze s daty, které jsou statickou součástí skriptu nebo byla vyplněna ve formuláři.

2.8 Webový klient

Webový klient je založen na technologii Apache Tomcat (více v kapitole 3.5) v jazyce Java. Stejně jako u tenkého klienta se jedná o architekturu klient + server. V tomto případě se však vyvíjí pouze serverová část páru a jako klient slouží webový prohlížeč.

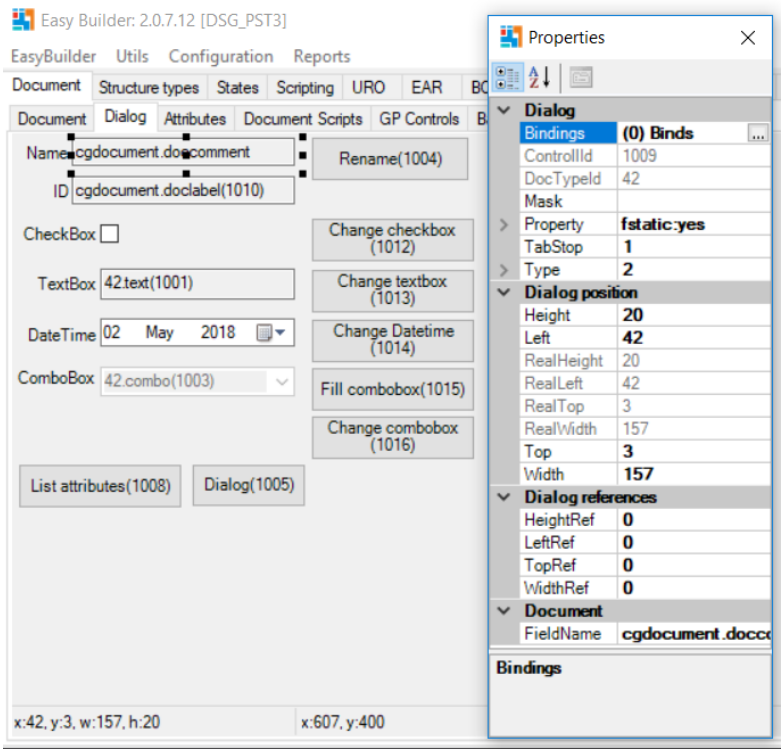
Vytváření formuláře pro správu dokumentu probíhá podobně jako u tenkého klienta, ale samotné zpracování XML definice formuláře probíhá jinak. Poté, co server od jádra obdrží XML definici, transformuje ji pomocí XSLT (více o XSLT v kapitole 3.7) na HTML dokument, který je následně poslán na klienta (prohlížeči) k zobrazení. HTML dokument se skládá ze standardních HTML prvků a JavaScript kódu (více o JavaScript v kapitole 4.3).

Webové prohlížeče mají z důvodu bezpečnosti mnoho omezení na to, jaký obsah jsou ochotné spustit. Toto klade velké omezení na množství skriptovacích jazyků, které bude možné ve webovém klientovi implementovat. Jak je popsáno v kapitole 4, různé skriptovací jazyky mají v prostředí prohlížeče různá omezení. Některá lze obejít a některá by implementaci neovlivnila, ale i tak je webový klient nejvíce limitujícím prvkem při volbě budoucí implementace, neboť zvolené řešení musí být funkční pro všechny klientské aplikace.

2.9 Easy Builder

EasyBuilder je administrativní nástroj pro správu systému EasyArchiv. Nejedná se však o správu dokumentů v systému. Ta probíhá přímo v klientských aplikacích. EasyBuilder upravuje záznamy v databázi týkající se podporovaných typů dokumentů, vytváří se v něm formuláře pro definici metadat dokumentů, vážou se atributy na kontrolní prvky, vytváří se skripty, spravují se akce, které je spouští a spravují se mnohé další konfigurovatelné části ovlivňující celkové chování systému. EasyBuilder tedy obstarává úpravy systému EasyArchiv tak, aby byl na míru přizpůsoben zákazníkům a umožňuje, aby jakékoli další mohly být rychle provedeny.

Tvorba uživatelských formulářů pro práci s metadaty dokumentu je řešena rozhraním, ve kterém si lze vytvořit téměř libovolný formulář pomocí jednoduchého drag&drop rozhraní. K takto vytvořeným prvkům je možné přiřadit atributy, které se tak stanou metadaty dokumentů vytvořených pomocí tohoto formuláře. K prvkům lze také přidat skripty spouštěné v jádře.



Obrázek 2.4: Definování formuláře v systému EasyArchiv

Vytváření spustitelných skriptů v současnosti probíhá nahráním celého C# projektu Visual Studia do databáze. EasyBuilder nemá schopnost upravovat skripty, ale umožňuje jejich nahrávání do databáze a z databáze pro případné úpravy. Po nahrání projektu je možné libovolnou statickou metodu v něm zaregistrovat jako použitelné skripty, které lze propojit s ovládacími prvky. Více o server-side skriptování systému EasyArchiv lze najít v kapitole 2.6.1.

EasyBuilder je naprogramovaný v jazyce C# 2.0 a dokáže pracovat s MySQL i Oracle databázemi.

Mezi jeho funkce patří:

- Vytváření a úprava formulářů metadat,
- vytváření atributů dokumentu,
- vytváření skriptů,
- správa uživatelů, uživatelských skupin a jejich práv,
- správa konstant v databázi,
- nastavení konstant systému,
- vytváření položek v menu systému,
- prohlížení historie změn v databázi
- a další...

2.9.1 Práce se skripty

Skripty jsou v databázi uložené celkem ve třech tabulkách:

- EASCRIPTPROJECTS – tabulka, ve které je uložen projekt a reference, které potřebuje.
- EASCRIPTFILES – tabulka, ve které jsou uloženy jednotlivé soubory jako text.
- EASCRIPTS – tabulka s jednotlivými volitelnými skripty (statické metody projektu).

Program EasyBuilder tak dokáže nahrávat do databáze téměř libovolný C# projekt Visual Studia a později ho opět exportovat pro možné úpravy.

Propojení skriptu a ovládacího prvku se provádí ve vlastnostech konkrétního prvku. Nastaví se zde akce, při které se má skript spustit (Message) a který skript se má spustit (Parameter). Tyto hodnoty se poté uloží do tabulky GPBinding, podle které se jádro rozhoduje, zda na uživatelskou akci reagovat a jak.

3 Použité technologie v EA

Systém EasyArchiv využívá ke své funkci množství různých technologií. V této kapitole jsou popsány ty důležitější a ty, které ovlivní průběh této práce.

3.1 COM rozšíření

„COM (Component Object Model) je platformně nezávislý, distribuovaný, objektově orientovaný systém pro vytváření binárních softwarových komponent které mohou interagovat.“ [2] (přeloženo z angličtiny)

COM určuje standardizovaný objektový model a programové požadavky, které umožňují COM objektům a komponentám interagovat mezi sebou. Nejedná se tedy o programovací jazyk a neurčuje ani implementační detaily aplikace. COM objekty mohou být uvnitř jediného procesu, ve více procesech nebo i na různých počítačích. Nemusí být vytvořené ve stejných programovacích jazycích, a proto se COM nazývá binární standard. Jedná se totiž o standard, který se aplikuje až poté, co byl program přeložen do binárního strojového kódu.

Používá se především k umožnění mezi-procesové komunikace mezi objekty v různých programovacích jazycích. To je hlavní vlastností COM rozšíření, vytváření objektů jazykově neutrálním způsobem použitelných i v prostředích, pro které nebyly stvořeny.

Proces, který využívá COM komponentu, nemusí být znovu zkompilován poté, co se implementace komponenty změní. V tomto ohledu je komponenta rozdělena na interface a implementaci. Při jakékoli změně, která nemění interface se mění pouze implementace. [12]

V mnoha oblastech byl COM do určité míry nahrazen .NET frameworkem a podporou webových služeb skrz WCF (O Windows Communication Foundation více v kapitole 3.4). COM objekty však mohou být stále použity všemi .NET jazyky pomocí .NET COM Interop.

Všechny COM komponenty jsou zaregistrovány do operačního systému při jejich instalaci. Když je chce programátor použít, musí vědět, kde je komponenta uložena a znát její GUID (Globally Unique IDentifier). GUID

je unikátní identifikátor, pomocí kterého může klient požádat systém o inicializaci komponenty. Registrované COM komponenty lze standardně najít v registrech na adrese HKEY_CLASSES_ROOT//CLSID.

Tenký a webový klienti přistupují k jádru systému (EasyCom) pomocí technologie COM.

3.2 C# skripty

Skripty, které může jádro systému spustit, jsou uloženy spolu s celým svým C# projektem, který je uložený v databázi. Když ho uživatel chce spustit, EasyCOM ho stáhne do lokálního úložiště, pokud je potřeba, tak ho sestaví (build), lokálně spustí a vrátí uživateli výsledek.

Projekt obsahující skripty je přeložen uvnitř skriptovací knihovny EAScripting vytvořené v jazyce C#. K přeložení projektu využívá knihovnu System.CodeDom.Compiler od společnosti Microsoft.

„Jmenný prostor System.CodeDom.Compiler obsahuje typy pro správu vytváření a kompilování zdrojového kódu v podporovaných jazycích. Generátory kódu mohou každý produkovat zdrojový kód v konkrétním programovacím jazyce podle struktury objektového model dokumentu s kódem (Code Document Object Model - CodeDOM), který se skládá z elementů poskytnutých jmenným prostorem System.CodeDom.“ [5] (přeloženo z angličtiny)

Jazyky, které lze zkompilovat pomocí System.CodeDom.Compiler, jsou C#, Visual Basic, C++ a JScript. Implementací rozhraní ICodeGenerator a ICodeCompiler lze vytvořit CodeDomProvider, který rozšiřuje podporu CodeDOM i na jiné programovací jazyky.

Jádro systému EasyArchiv v sobě obsahuje skriptovací knihovnu EAScripting, která má referenci na Interop.EasyCom, čímž získává přístup k rozhraním jádra. Je tedy možné C# skriptům předat reference na objekty jádra a umožnit skriptům přístup k jeho funkcím bez potřeby implementace wrapperu (viz. kapitola 3.3).

Největší nevýhodou a zároveň důvodem pro vznik této práce je skutečnost, že se C# skripty spouští v jádře. Jedná se tedy čistě o server-side operace. Silný klient je klientská aplikace postavená kolem jádra, a tak se jí toto omezení netýká a skripty s grafickým rozhraním se na něm spouští v kli-

entském prostředí, které je zároveň i prostředím serveru. Tenký a webový klient standardně mají klienta a jádro na různých strojích, takže otevření dialogu pomocí skriptu vede k zamrznutí klientské aplikace, která čeká na to, až skript doběhne; až uživatel zavře dialog, který se otevřel na serveru, který ani nemusí mít grafický výstup.

3.3 Wrapping COM komponenty

Obalení (wrapping) COM komponenty probíhá za účelem poskytnutí vrstvy mezi klientem COM komponenty a komponentou samotnou. Umožňuje vytváření objektů a volání metod komponenty pomocí proměnných a struktur které fungují v jazyce klientského programu. Obalovací třída pak převede proměnné na takové, které vyhovují rozhraní COM komponenty a spustí požadovanou metodu.

Ke spouštění skriptů v jádře slouží C# knihovna (EAScripting), která je obalená c++ knihovnou (ScriptingWrapper), která překládá existující rozhraní C# knihovny na rozhraní kompatibilní s C++ kódem.

3.4 WebService Tenkého klienta

Webová služba, kterou tenký klient využívá pro komunikaci mezi serverem a klientem, je součástí Windows Communication Foundation (WCF).

„WCF je framework, sloužící k tvorbě servisně orientovaných aplikací, vyvíjený společností Microsoft. Jeho používání umožňuje asynchronní posílání zpráv z jednoho koncového bodu (endpoint) služby na jiný. Koncový bod služby může být součástí kontinuálně dostupné služby hostované pomocí IIS (Internet Information Services), nebo hostován jako součást aplikace. Zprávy posílané mezi koncovými body mohou být jednoduché jako například jeden znak nebo slovo poslané jako XML dokument, nebo může jít o komplexní proud binárních dat.“ [8] (přeloženo z angličtiny)

Jelikož WCF následuje principy servisně orientované architektury, lze pomocí něj implementovat systémy s distribuovanými výpočty a systémy obsluhující vzdálené uživatele. Klienti mohou přistupovat k mnoha službám a služba může být využívána více klienty.

Služby obvykle poskytují Web Services Description Language rozhraní (WSDL) umožňující jakémukoli WCF klientovi využívat služby nezávisle na platformě, na které jsou spuštěny.

Výhodou služeb je, že jsou obvykle tvořené jako univerzální přístupový bod k datům a funkcím, místo toho, aby vznikala jedna služba pro každý konkrétní účel nebo klienta. WCF SOA spoléhá na webové služby na přijímání a doručování dat. Tyto dvě vlastnosti ručí, že se ke službě může připojit jakýkoli klient splňující potřebné kontrakty. Klient je nezávislý na .NET platformě nebo jakýchkoliv proprietárních komunikačních protokolech. [12]

3.5 Tomcat

„Apache Tomcat, často nazývaný jako Tomcat Server, je open-source Java Servlet kontejner vyvíjený společností Apache Software Foundation.“ [6] (přeloženo z angličtiny)

Obrázek 3.1: Logo Apache Tomcat

Tomcat poskytuje jednoduchou, transparentní, hardwarově nenáročnou (oproti alternativním řešením) metodu vývoje Java Servletů a JavaServer Pages (JSP). Tomcat však neposkytuje vývojové prostředí pro vývoj webových stránek.

Velkou výhodou tohoto systému je jednoduchost jeho použití. I starší verze 7.0 využívaná webovým klientem byla připravena během několika minut. Po stažení archivu z oficiální stránky [6] ho pouze stačilo rozbalit, poskytnout „war“ archive s klientem a spustit Tomcat. Program samotný se o vše ostatní postará automaticky a publikuje webovou stránku na lokální doménu. Hned po spuštění poskytuje domovskou stránku, na které lze nastavit, jaké webové stránky by měly být aktivní. Také umožňuje nahrát webové stránky skrz webové rozhraní.

Celý webový klient je postaven na technologii Apache Tomcat a komunikuje s jádrem EasyCom pomocí programu Jacob.

3.6 Jacob - JAVA-COM Bridge

"Jacob je JAVA-COM přemostění umožňující volání COM komponent z Javy. Využívá JNI k provádění nativních volání COM a Win32 knihoven. Projekt započal v roce 1999 a je aktivně udržován a využíván tisíci vývojáři po celém světě. Jako open-source projekt benefitoval ze zkušeností svých uživatelů, mnozí z nichž přispěli s úpravou kódu nebo poskytli zpětnou vazbu."[7] (Přeloženo z angličtiny. Citát z roku 2004. Prohlášení o aktivní údržbě projektu již nejsou aktuální a projekt se zdá neaktivní od roku 2015. Je však stále využíván v rámci webového klienta.)

Jacob je aktuálně využíván v rámci webového klienta a umožňuje volání jádra jako COM komponenty serverem Tomcatu, který je psán v Javě.

3.7 XSLT

Extensible Stylesheet Language Transformations (XSLT) je jazyk pro transformaci XML dokumentů na jiné XML dokumenty, HTML dokumenty, text, XSL formátovací objekty a další formáty. Často se používá spolu s XSL formátovacími objekty, které lze dále transformovat do formátů jako je PDF, PostScript nebo SVG.

Zpracovávaný dokument není změněn, nový dokument z něj pouze vychází. Standardním vstupním dokumentem je XML soubor, ale jakákoli data, ze kterých dokáže procesor vygenerovat XQuery a XPath model lze použít jako vstup transformace. Příkladem takového vstupu je relační databáze.

Webový klient využívá XSLT k transformaci XML dokumentu, který reprezentuje formulář pro definici metadat dokumentu. Tento XML dokument je posléze upraven na HTML a poslán klientovi.

3.7.1 XPath

XPath lze využít k navigaci mezi elementy a atributy XML dokumentu a je důležitou součástí XSLT standardu.

K navigaci dokumentem XPath využívá syntax podobný navigaci adresá-

řovou strukturou operačních systémů. XPath dále obsahuje přes 200 funkcí sloužících k práci s daty, porovnávání hodnot, zpracování dat, manipulaci s pořadím a přesnější navigaci dokumentem.

3.7.2 XQuery

XQuery je dotazovací jazyk nad XML daty. Slouží k vyhledávání a získávání elementů a atributů z XML dokumentů.

Příklad použití XQuery:

```
doc("dialog.xml")/gpform/gpcontrols/gpcontrol[type=5]
```

Příklad vybere všechny ovládací prvky ze XML souboru dialog.xml, které mají element „type“, který obsahuje číselnou hodnotu rovnou 5ti (nejedná se o atribut „type“, ale vnitřní uzel). GPControls typu 5 jsou tlačítka.

3.8 Relační databáze (MySql/Oracle)

Relační databáze je sada datových položek zorganizovaných do podoby předem definovaných tabulek, které mohou být navzájem propojeny různými typy klíčů. V těchto tabulkách lze přistupovat k datům, vyhledávat v nich a extrahovat je v různých formách bez nutnosti úpravy databázové tabulky.

K vyhledávání v relačních databázích slouží strukturalizovaný dotazovací jazyk Structured Query Language (SQL). Pomocí SQL lze nejen vyhledávat v databázi, ale také aktualizovat a mazat záznamy, vytvářet a upravovat tabulky, přidávat omezení na data a další.

Velkou výhodou relačních databází je jednoduchost pochopení konceptu, z kterého vychází. To umožňuje relativní jednoduchost vytváření a přístupu k nim. Jsou také jednoduše rozšiřitelné bez potřeby úpravy existujících dat a aplikací, které s nimi pracují.

Systém EasyArchiv dokáže fungovat s MySql i Oracle databází. Obecně řečeno Oracle databáze má více funkcí a je vhodnější pro rozsáhle projekty a datové sklady, zatímco MySql je uživatelský přívětivější, v mnoha případech rychlejší a zadarmo. V kontextu tohoto projektu lze však říct, že jsou si oba

systémy více méně rovné. Jediným rozdílem, který tento projekt ovlivňuje, jsou drobné rozdíly v datových typech a syntaxi SQL příkazů.

3.9 Komunikační protokol EA

Data posílaná mezi jádrem a různými servery jsou ve formě XML dokumentu. Tento dokument obsahuje všechny informace ke konstrukci formuláře pro vytváření a správu metadat dokumentu. Spolu s názvy, rozměry, rozmístěními a typy jednotlivých ovládacích prvků formuláře také obsahuje identifikátory prvků, které umožní jádru v případě uživatelské akce rozpoznat, který prvek formuláře vyžaduje reakci jádra a jak tuto reakci vygenerovat (například spuštěním skriptu, poskytnutím dat nebo spuštěním vnitřní akce jádra).

XML dokument nesoucí data o formuláři může vypadat například takto:

```
<gpform>
   <gpstate>1</gpstate>
   <width>796</width>
   <height>278</height>
   <gpcontrols>
      <gpcontrol>
         <controlid>CID1003</controlid>
         <type>2m</type>
         <value>
                  TYPEID = 16: cgdocument.owner = 'none';
         </value>
         <position>
            <left>60</left>
            <top>115</top>
            <width>733</width>
            <height>160</height>
         </position>
         <title>Query</title>
         <tabindex>9</tabindex>
         <properties>
            <property name="style" value="10a110c4" />
            <property name="fstatic" value="yes" />
            <property name="fsw" value="61" />
         </properties>
```

```
            <events />
        </gpcontrol>
        <gpcontrol>
            <controlid>CID1008</controlid>
            <type>5</type>
            <value>Add</value>
            <position>
                <left>687</left>
                <top>90</top>
                <width>104</width>
                <height>20</height>
            </position>
            <title>Add</title>
            <properties>
                <property name="fstatic" value="no" />
                <property name="pseudo" value="yes" />
                <property name="style" value="50002300" />
            </properties>
            <events>
                <onclick />
            </events>
        </gpcontrol>
    </gpcontrols>
</gpform>
```

4 Skriptování

Skript je program napsaný pro prostředí aplikace, který automatizuje nebo ulehčuje provedení úkolů, které by jinak musel ručně vykonávat uživatel. Často se vytváří pro situace, kde lze automatizovat části procesu, které nebylo možné předpokládat při návrhu aplikace nebo které vznikly jako součást jiného procesu v rámci společnosti. Mohou provádět jednoduché akce jako předvyplňování hodnot, kontrolovat, zda uživatel správně vyplnil formuláře nebo mohou vytvářet celé databázové tabulky a spouštět další aplikace vykonávající vlastní cíle.

V rámci systému EasyArchiv skripty slouží dvěma hlavním funkcím:

- Usnadnění práce uživatelům – kontrolují například, zda jsou jména dokumentů podle firemní normy nebo z poskytnutých dat generují SQL příkaz.
- Zpřístupnění rozšířené funkcionality – lze pomocí nich přistupovat k funkcím jádra. Umožňují tak například získat dodatečné informace o jiných dokumentech, uživatelích nebo přístupových právech.

Klientské skriptování, které je zaměřením této práce, by se spíše mělo soustředit na uživatelskou část systému. Jelikož nebude mít přístup k funkcím jádra, budou se muset uživatelské skripty spokojit s předem připravenými daty a s uživatelským rozhraním, které současné skriptování postrádá.

V této kapitole jsou analyzovány některé skriptovací jazyky a metody jejich spuštění. Jsou zhodnoceny jejich silné a slabé stránky a je navržen skriptovací jazyk, který bude využit k implementaci rozšíření.

4.1 Požadavky

Na klientské skriptování se vztahuje několik požadavků, které vychází ze zadání a je potřeba je brát v potaz. Dále existuje několik požadavků, kterým by bylo vhodné vyhovět, aby bylo možné budoucí rozšíření řešení.

Základním požadavkem je bezproblémová integrace do současného systému EasyArchiv. Klientské skripty budou muset být spustitelné na všech třech klientských aplikacích a bude možné je vytvářet a upravovat v rámci

programu EasyBuilder. Zde vzniká asi největší omezení. Webový klient požaduje, aby bylo možné zvolené řešení spustit ve webovém prohlížeči. Skripty také musí být možné uchovávat v obou typech relačních databází (Oracle a MySql) a musí je být možné předávat mezi aplikacemi uvnitř XML dokumentu.

Řešení musí zároveň sloužit pouze jako rozšíření existujícího systému. Nesmí jakýmkoli způsobem omezit existující funkcionalitu. Uživatel by měl být schopen přejít na verzi EasyArchivu s klientskými skripty, aniž by jakkoli zaznamenal změnu. Datová úložiště by také neměla podstoupit žádnou změnu, snad s výjimkou přidání databázových tabulek pro uložení nových dat potřebných ke spuštění klientských skriptů.

Hlavním zaměřením tohoto rozšíření současné funkcionality je poskytnutí větších možností uživatelské interakce. Zvolený skriptovací jazyk musí být schopen vytvářet uživatelské dialogy nebo být schopen spolupracovat s jinou technologií, která by obstarávala grafické rozhraní.

V případě skriptovacího jazyka, který by neposkytoval možnost vytvoření uživatelského rozhraní, by bylo možné rozšířit současný systém EasyArchivu pro vytváření a zobrazování dialogů o možnost použít ho tak, aby šel využít spolu s klientskými skripty. Pokud by se takové rozšíření zobrazování dialogů ukázalo jako proveditelné, bylo by také možné místo přidávání dalšího skriptovacího řešení pouze rozšířit grafické rozhraní a použitím ho spolu se současným skriptovacím systémem. Spoluprací těchto dvou komponent by bylo možné dosáhnout stejného výsledku jako implementací dalšího skriptovacího rozšíření.

Důležitým prvkem řešení je nenáročnost na uživatelské straně i co se týče nasazení. Bylo by tedy velkou výhodou řešení, které nevyžaduje instalaci dodatečného softwaru, který by potřebovaly některé jazyky spouštěné na virtuálním stroji. Zároveň by byla výhodou implementace řešení, které nevyžaduje nákup dodatečného softwaru.

Další z nefunkčních požadavků, nebo spíše preferencí, je jednoduchost využitého skriptovacího jazyka nebo využití takového jazyka, který je obecně znám. Od uživatelů programu EasyBuilder, který bude pracovat s kódem skriptů, lze očekávat určitou úroveň počítačové a programovací gramotnosti, ale jednoduchý nebo známý jazyk sníží nároky na znalosti zákazníků a umožní jim tak rozšíření funkcionality celého systému.

4.2 Powershell

„PowerShell je textový interpretr příkazů a skriptovací jazyk postavený na .NET Frameworku od společnosti Microsoft. Je navržen, aby umožnil zkušeným uživatelům a administrátorům jednoduše automatizovat administraci operačních systémů (Linux, macOS, Unix, and Windows), procesů a aplikací.“ [3] (přeloženo z angličtiny)

Obrázek 4.1: Logo Microsoft PowerShell

Jelikož je PowerShell postavený na .NET Frameworku, poskytuje všechny funkce, které jsou v něm dostupné a to prostřednictvím cmdletů, což jsou specializované třídy implementující konkrétní funkce. Cmdlety lze seskupovat do skriptů s příponou `.ps1`, které pak poskytují rozšířené možnosti využití logických operací.

PowerShell byl zamýšlen jako náhrada příkazové řádky operačního systému Windows. Dokáže tedy pracovat s klasickými Windows aplikacemi, ale také vytvářet instance .NET tříd a COM objektů. Umí nejen pracovat v souborovém systému, ale má také přístup k registrům a certifikátům.

Jako skriptovací jazyk s dynamickým typováním dokáže PowerShell implementovat komplexní operace. Podporuje vytváření proměnných, funkcí, podmínek, cyklů, vyvolávání výjimek a používání lambda výrazů.

Velkou výhodou PowerShellu v rámci této práce je jeho schopnost spustit libovolný .NET kód bez potřeby ho nejdříve zkompilovat v rámci vývojového prostředí. Powershell dokáže libovolný .NET kód přeložit za běhu na CIL (Common Intermediate Language, jazyk spouštěný virtuálním strojem), který je uchován v paměti po dobu běhu skriptu. PowerShell také dokáže převzít objekty a funkce .NET knihoven. Je tedy možné využít například `System.Windows.Forms` knihovnu k vytvoření téměř libovolného uživatelského dialogu.

PowerShell skripty lze spustit na jakémkoli počítači s nainstalovaným operačním systémem Windows 7 nebo novějším nebo nainstalovaným programem PowerShell. Pokud jsou skripty spustitelné na daném zařízení, jakákoli aplikace s přístupem k příkazové řádce je dokáže spustit. Webové prohlížeče

standardně nemají přístup k příkazové řádce z důvodu bezpečnosti.

Ve snaze spustit PowerShell skript z webového klienta byl rámci této práce vyvinut prototyp add-onu pro prohlížeč Mozila FireFox. Ačkoli se povedlo spustit příslušný skript z prostředí webového prohlížeče, problémy spojené s vývojem a instalací add-onů pro více prohlížečů převážily výhody, které PowerShell poskytuje.

4.2.1 Zhodnocení

PowerShell jako skriptovací jazyk má mnoho výhod. Skripty by bylo možné psát v jakémkoli jazyce, který je součástí .NET, včetně C#, ve kterém se v systému EasyArchiv píší server-side skripty, a lze v rámci skriptů využít libovolné knihovny použitelné v jazycích .NET. Výhodou je také, že fungují prakticky na všech modernějších systémech a lze je spustit z příkazové řádky, aniž by uživatel potřeboval cokoli kromě kódu skriptu.

Kritickým problémem je komplikovaný způsob spouštění PowerShell skriptů z webového klienta. Kvůli tomuto problému je využití PowerShell skriptů v této práci velmi nepraktické.

4.3 JavaScript

JavaScript je široce rozšířený interpretovaný programovací jazyk, který pohání dynamické chování většiny webových stránek. Spolu s HTML a CSS je JavaScript jednou ze tří klíčových technologií webových stránek. Lze pomocí nich vytvářet grafické rozhraní interagující s uživatelem. Lze jej použít pro tvorbu formulářů, webových stránek nebo i her. JavaScript je tak rozšířený, že existuje mnoho JavaScriptových enginů a snad všechny současné webové prohlížeče dokáží JavaScript spustit bez nutnosti doinstalovávat plug-iny.

Obrázek 4.2: Logo JavaScriptu

JavaScript má dynamické typování, což znamená, že proměnná nemá od

začátku vykonávání programu pevně daný typ. Lze jí tak za běhu přiřadit různé typy dat. Ačkoli se JavaScript nazývá objektově orientovaným jazykem, nevyužívá koncept třída-instance stejně jako typické objektově orientované jazyky. Místo toho disponuje prototypováním. Dokáže tak napodobovat objektově orientované principy jako třeba dědičnost.

JavaScriptové programy se nejčastěji spouštějí v rámci webových stránek a důležitou vlastností je, že se spouští na straně klienta. Z toho důvodu se však potýká s určitými bezpečnostními omezeními, která zabraňují spouštění škodlivého kódu. JavaScript spuštěný ve webovém prohlížeči například standardně nemá přístup k datům na disku nebo k informacím o identitě uživatele.

4.3.1 Technologie obohacující JavaScript

Obrázek 4.3: Logo Bootstrapu

Díky své popularitě se JavaScript dočkal knihoven, které ho rozšiřují o potřebné funkce nebo upravují jeho funkcionalitu ke specifickému účelu. Mezi tyto technologie patří například Bootstrap nebo Node.js.

Bootstrap je open-source front-endová knihovna zaměřená na vývoj webových stránek a webových aplikací. Obsahuje JavaScript rozšíření, HTML a CSS šablony, písma, vzhledy, ovládací prvky a další elementy sloužící k umožnění, zpříjemnění a usnadnění uživatelské interakce s webovou stránkou.

Na druhou stranu Node.js je open-source systém vytvořený v jazyce JavaScript a slouží k tvorbě především serverových aplikací. Jeho silnou stránkou je dobrá škálovatelnost a jeho snaha o minimalizaci hardwarových nároků především využíváním asynchronních I/O operací.

Obrázek 4.4: Logo Node.js

Na rozdíl od obvyklého použití JavaScriptu, Node.js umožňuje vytvářet obsah dynamické webové stránky před tím, než je odeslána webovému prohlížeči. Node.js tedy umožňuje vytvářet webové stránky bez vy-

užívání zbytečně velkého množství různých programovacích jazyků. Místo používání PHP nebo ASP na straně serveru a JavaScript na straně klienta, s Node.js lze použít pouze JavaScript.

4.3.2 C# scripting engine

„Skriptovací engine, který je součástí .NET Frameworku, se nachází ve jmenném prostoru MSSciptControl. Jedná se o poměrně špatně dokumentovanou a zastaralou knihovnu.“ [4] (přeloženo z angličtiny)

Umožňuje předávání COM komponenty skriptu, takže JavaScript skript by tak měl přístup ke stejným třídám jako mají skripty spouštěné v jádře. Není však jednoduchý způsob, jak zpřístupnit data v těchto třídách klientskému programu nebo webovému prohlížeči.

MSSciptControl je menší, jednoduše použitelný JavaScript engine.

4.3.3 JavaScript.Net (Noesis)

”Knihovna JavaScript.net integruje V8 JavaScript engine (více v kapitole 4.3.4.1) a vystavuje ho CLI (Common Language Infrastructure). Kompiluje v runtimu a spouští skripty přímo z .NET kódu. Umožňuje CLI objektům, aby byly manipulovány JavaScript skriptem.”[9] (přeloženo z angličtiny)

Noesis je open-source knihovna dostupná na GitHubu a NuGetu. Je modernější než MSScirptControl, jednodušeji se používá a díky tomu, že se jedná o poměrně vyžívanou knihovnu, je větší šance, že se v případě nalezení problému půjde na někoho obrátit.

4.3.3.1 Common Language Infrastructure

„Common Language Infrastructure (CLI) je otevřený technický standard vyvíjený společností Microsoft a standardizovaný ISO a EMCA. Popisuje spustitelný kód a runtime prostředí, které popisuje použití vysoko-úrovňových jazyků na různých počítačových platformách bez potřeby úpravy pro různé architektury. Mezi implementace CLI patří například .NET Framework, .NET Core, DotGNU nebo Portable.NET.“ [5] (přeloženo z angličtiny)

Díky integraci s CLI je možné pro aplikace a knihovny sdílet mezi sebou funkce, vlastnosti, pole, události, objekty, typy, metody a další prvky .NET aplikací, i když nejsou vytvořeny ve stejném programovacím jazyce.

4.3.4 ClearScript

"ClearScript je knihovna, která umožňuje jednoduché přidání skriptování do .NET aplikací. V současnosti podporuje JavaScript (pomocí V8 a JScript enginů) a VBScript."[10] (přeloženo z angličtiny)

Silnou stránkou ClearScriptu je jednoduchost využití a integrace s CLI. Skripty mohou volat metody hostujícího programu s výstupními parametry, volitelnými parametry a poli parametrů. Aplikace také může přímo volat funkce skriptu a přistupovat k jeho objektům. Disponuje také plnou podporou debugování a podporuje tři různé skriptovací enginy (Google V8, Microsoft JScript a VBScript).

ClearScript je moderní, aktivně vyvíjená knihovna, která je velice jednoduchá na použití. Je díky V8 enginu vysoce výkonná a optimalizovaná pro více-vláknové využití.

4.3.4.1 Chrome V8

„Chrome V8 je vysoce výkonný open-source JavaScript engine od společnosti Google. Je napsán v C++ a využívá ho například Google Chrome, Node.js a další.“ [11] (přeloženo z angličtiny)

V8 kompiluje JavaScript přímo do strojového kódu a poté ho spouští. Zkompilovaný kód je dále optimalizován za běhu pomocí heuristik profilu běhu programu. Mezi optimalizační techniky patří Inline expansion, Copy elision a Inline caching.

Jako garbage collector využívá stop-the-world precizní generační garbage collector. Google udává, že právě ten je jedním z klíčů k výkonu V8.

V8 také umožňuje jakékoli C++ aplikaci odhalit své objekty a funkce JavaScript kódu.

4.3.5 AJAX

AJAX (Asynchroní JavaScript a XML) je přístup k množině technologií (HTML or XHTML, CSS, JavaScript, DOM, XML, XSLT a XMLHttpRequest), který umožňuje webovým aplikacím provádět rychlé inkrementální úpravy uživatelského rozhraní bez potřeby znovu nahrát celou webovou stránku. Stávají se tak rychlejší a více responsivní.

Ačkoli se nejedná o metodu spouštění skriptů, jedná se o užitečnou možnost, jak obohatit JavaScript skripty, která by mohla zlepšit flexibilitu výsledného řešení této práce.

Pomocí AJAXu lze:

- Upravovat webovou stránku, aniž by musela být celá znovu nahraná.
- Dotazovat se o data na serveru poté, co byla stránka nahraná.
- Přijímat data ze serveru poté, co byla stránka nahraná.
- na pozadí odesílat data na server.

V rámci klientského skriptování by bylo možné pomocí AJAXu dynamicky získávat data z jádra systému EasyArchiv i přes to, že skript běží na klientské straně a aniž by bylo nutné implementovat tři různé způsoby pro přístup k jádru (jeden přístup pro každého klienta).

4.3.6 Zhodnocení

JavaScript je široce rozšířený skriptovací jazyk, který lze díky množství enginů spustit na různých platformách. Je nejčastěji využíván pro spouštění skriptů na klientské straně, ale pomocí Node.js v něm lze implementovat i server. Díky knihovnám, které počítají s využíváním CLI objektů, lze skriptu zpřístupnit objekty a metody aplikace, ve které je spuštěn.

Důležitou vlastností v rámci této práce je, že JavaScript skripty je možné spustit na všech klientech bez větších potížích, protože webové prohlížeče obvykle počítají se spouštěním JavaScriptu.

JavaScript je dále poměrně rozšiřitelný. Jelikož se jedná o široce využívaný skriptovací jazyk, bylo pro něj vytvořeno množství knihoven jako napří-

klad Bootstrap nebo jQuery. Spolu s možností provádět asynchronní volání je JavaScript silným kandidátem pro tuto práci.

Jednou z nevýhod JavaScriptu je, že sám o sobě neposkytuje grafické uživatelské rozhraní (s výjimkou volání metody `alert()`). JavaScript je navržen tak, aby byl využíván zároveň s technologiemi jako HTML a CSS. Toto řešení by tedy vyžadovalo spojení s další technologií, která by obstarávala vizuální část skriptování.

4.4 Visual Basic pro Aplikace (VBA)

VBA je implementace jazyka Visual Basic 6, jehož vývoj byl ukončen v roce 2008. Umožňuje vytváření uživatelských skriptů sloužících k automatizaci procesů v rámci aplikací a operačního systému. VBA má přístup k Windows API a jelikož je součástí .NET Frameworku, lze jej jednoduše zakomponovat do jiných .NET aplikací. Často je součástí sady aplikací Microsoft Office ve kterých umožňuje tvorbu skriptů k automatizaci práce, ale lze jej najít také například v programech společnosti Autodesk. VBA je proprietární software společnosti Microsoft a není tedy open-source.

VBA kód se kompiluje do mezijazyka (Microsoft P-Code), který je pak spuštěn virtuálním strojem spuštěným hostující aplikací.

Díky přístupu k Windows API lze v rámci VBA skriptů vytvářet grafické uživatelské rozhraní, ale tak tomu není v kontextu webového prohlížeče. Existují ale komerční řešení, které pomocí protokolu vzdálené plochy (Remote Desktop Protocol, RDP) a technologie HTML5 dokáží spustit libovolnou aplikace uvnitř prohlížeče.

4.4.1 Zhodnocení

Ačkoli je VBA populárním skriptovacím jazykem obzvlášť pro kancelářský software, kterým EasyArchiv je, se spouštěním na webovém klientu si elegantně neporadí. Řešení, která by to umožnila, jsou nejen komerční, ale také nepoměrně komplexní a rozsáhlá na úlohu, kterou by splňovaly v rámci této práce.

4.5 Další skriptovací jazyky

Mezi další rozšířenější skriptovací jazyky patří například PHP, Ruby, Lua, JCL, Tcl/Tk a Perl. Ačkoli mají tyto jazyky své vlastní kladné a záporné stránky, jedno mají společné. Bylo by obtížné nebo nemožné spustit je na klientské straně webového klienta. Webové prohlížeče jsou navrhované tak, aby zajistily co největší bezpečnost a zabraňují tedy webovým stránkám cokoli spouštět na hostujícím zařízení. Z toho vyplývá, že jediný jazyk, který bude možné použit pro klientské skriptování v rámci této práce je takový, jehož engine je součástí webového prohlížeče.

Jedním ze skriptovacích jazyků spustitelných ve webovém prohlížeči je ActionScript, který vychází ze standardizované verze jazyka JavaScript, nazvané ECMAScript, a byl po dlouhou dobu používán společně s technologií Adobe Flash. Právě jeho závislost na technologii Flash je nevýhodou, neboť ta je od vydání HTML5 na ústupu. Na rozdíl od JavaScriptu navíc nedokáže přistupovat k částem webové stránky, které nejsou součástí technologie Flash. Takže nejen, že je ActionScript z velké části podobný JavaScriptu, má oproti němu také zásadní nevýhody.

Další technologií pro skriptování ve webových prohlížečích je ActiveX od společnosti Microsoft. Disponuje alespoň částečnou kompatibilitou s frameworkem .NET, což je dobrou vlastností pro integraci s tenkým a silným klientem. Bohužel je ActiveX v současnosti defaultně podporována pouze v prohlížeči Internet Explorer 11 od společnost Miscrosoft (ActiveX není přítomný ani v jejich posledním prohlížeči Microsoft Edge).

5 Navržené řešení

Prozkoumáním různých možností implementace klientského skriptování v kapitole 4 se ukázalo, že zřejmě nejlepším skriptovacím jazykem pro účely této práce je JavaScript. Jedná se o široce rozšířený jazyk, který může pomocí CLI interagovat s objekty a metodami aplikace, ze které je spouštěn. Jeho největší výhodou je však spustitelnost v rámci webového klienta.

JavaScript sám o sobě neposkytuje grafické uživatelské rozhraní (s výjimkou metody alert()), takže spolu s obstarávání spouštění JavaScript skriptů je potřeba zajistit tvorbu uživatelského dialogu. K tomuto účelu dobře poslouží HTML. Detaily k tomuto řešení jsou blíže popsané v kapitolách 5.1 a 5.4.

Kvůli velkým rozdílům mezi technologiemi klientů je zapotřebí vytvořit dvě různá řešení. Silný a tenký klienti mohou spouštět skripty za pomoci knihovny, která bude obstarávat komunikaci mezi klientem a skriptem podobným způsobem, kterým je v současnosti implementováno skriptování na straně serveru. Webový klient tuto knihovnu využívat nebude, protože ji nebude možné zpřístupnit na klientské straně. Místo knihovny bude webový klient ke spuštění skriptů využívat funkcionalitu přítomnou ve webovém prohlížeči a komunikace mezi skriptem a klientem bude obstarávána pomocí JavaScriptu.

Systém EasyArchiv má mnoho různých triggerů sloužících ke spouštění skriptů. Tato práce se primárně zaměřuje na uživatelskou interakci, takže prioritně budou implementovány dva konkrétní:

- onInit – trigger spouští skript při každém otevření formuláře metadat dokumentu. Může sloužit například k předvyplnění údajů.
- onClick – trigger spouští skript při kliknutí na tlačítko. Takový skript může sloužit například k validaci vložených údajů, aktualizaci dat, nebo zobrazení nápovědy.

V systému existuje více událostí, se kterými lze skriptování spojit, ale tyto dvě patří mezi základní. Řešení může být v budoucnosti rozšířené o zbývající triggery.

5.1 EAClientScripting

EAClientScripting je název knihovny, která bude obstarávat spouštění skriptů pro silného a tenkého klienta. Knihovna bude vytvořena v jazyce C# a bude tak mít přístup ke knihovně System.Windows.Forms.WebBrowser. WebBrowser komponenta poskytuje prostředí, které dokáže interpretovat validní HTML kód a zobrazit grafické rozhraní stejně jako by to udělal webový prohlížeč. Obsahuje také JavaScript engine umožňující spouštění skriptů. Využitím knihovny System.Windows.Forms.WebBrowser nebude zapotřebí přidávat knihovny třetích stran k projektům EasyArchiv. A jelikož se jedná o knihovnu společnosti Microsoft, která vyvíjí .NET Framework, lze se spolehnout na pokračování její údržby a aktualizace.

Knihovna pro spouštění uživatelských skriptů bude do jisté míry podobná knihovně EAScripting, která v jádře obstarává práci se skripty. Bude se skládat z následujících částí:

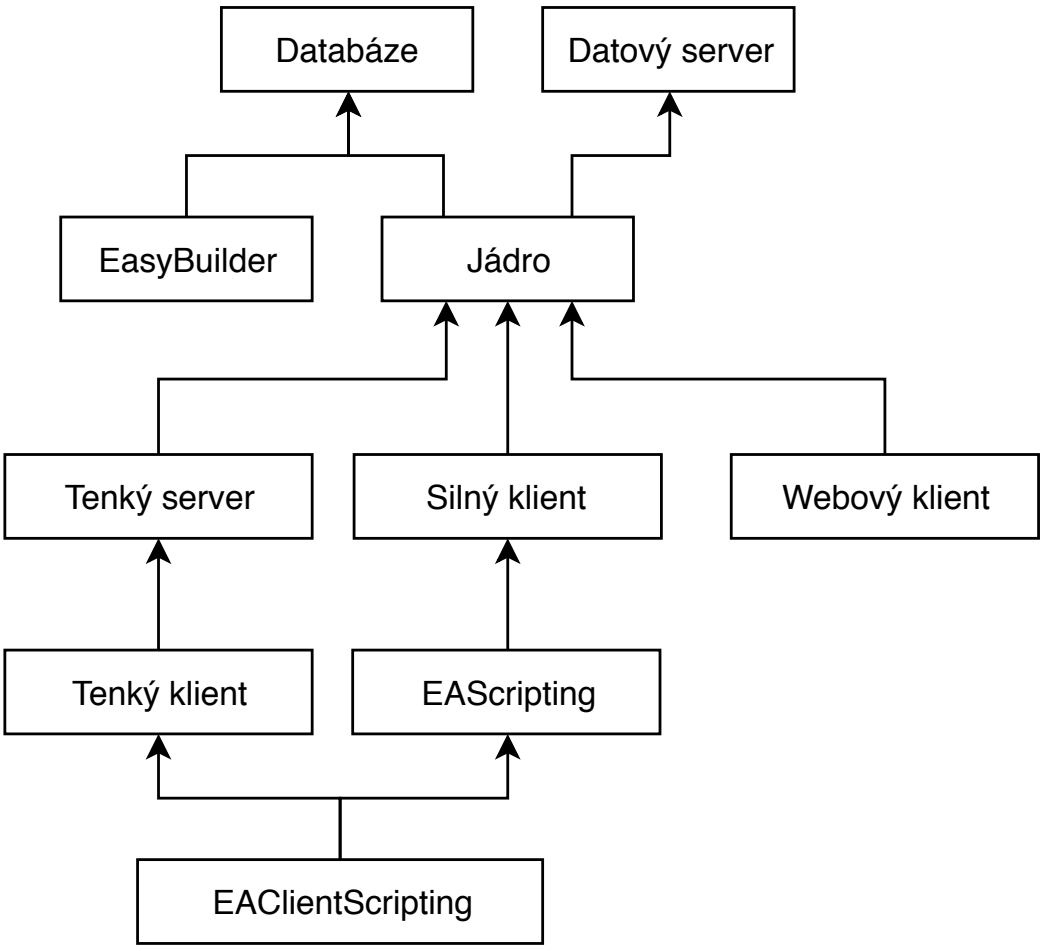
- Spouštění skriptů – tato část knihovny bude obstarávat zpracování skriptu a spuštění skriptu. Obstará předávání argumentů, vytváření dialogu, ve kterém bude umístěna komponenta WebBrowser, a obnoví běh klienta poté, co skript dokončí svou činnost.
- Argumentu skriptů – skriptu se budou předávat hodnoty formuláře ze kterého bude skript spuštěn. Třídy reprezentující argumenty budou vytvořeny, aby se ovládaly stejně jako argumenty skriptů spouštěných v jádře. Velká část proměnných a metod v těchto třídách nebude zpočátku využívaná, ale z uživatelského hlediska bude mezi používáním dvou různých způsobů skriptování menší rozdíl.

Tato knihovna bude sloužit pouze pro spouštění klientských skriptů v silném a tenkém klientovi, jak je vidět na diagramu 5.1. Webový klient bude spouštět skripty ve webovém prohlížeči.

Proměnné předávané skriptu jsou hodnoty formuláře, ze kterého je skript spuštěn. V okamžik, kdy uživatel vytvoří skript (v programu EasyBuilder), nebude ještě známo, z jakého formuláře bude skript spouštěn, bude možné spouštět ho i z více různých formulářů, a přístup k proměnným bude tedy čistě na uživateli. Bude to tedy uživatel, který si bude muset zajistit, že přistupuje pouze k existujícím proměnným formuláře. Vytváření nových proměnných v kontextu argumentů je možné, ale klientská aplikace nebude vědět, jak s nimi pracovat a bude je tedy ignorovat.

Jelikož se pro grafické rozhraní vytváří vlastní dialog, ve kterém je vložena komponenta WebBrowser, jsou zapotřebí dva typy skriptů:

- Skript s dialogem – do komponenty WebBrowser se nahraje skript (HTML i JavaScript), zobrazí se dialog a v něm se vykreslí grafické rozhraní definované pomocí HTML.
- Skript bez dialogu – ze skriptu se vyextrahuje jméno metody, která se má spustit, skript se vloží do WebBrowser komponenty a metoda se manuálně spustí pomocí funkce InvokeScript().



Obrázek 5.1: Navrhovaná úprava struktury systému EasyArchiv

5.2 Silný klient + jádro

Způsob integrace knihovny EAClientScripting do jádra klienta je prakticky identická způsobu, kterým je integrována knihovna EAScripting. V jádře systému jsou metody, které spouští staré skripty. Pro účely klientského skriptování je stačí do jisté míry zreplikovat. Jednalo by se o následující metody jádra:

- LoadScript – metoda z databáze načte skript a vloží ho do příslušné struktury.
- RunScript – vytvoří objekty pro předávání argumentů skriptu, naplní je daty a spustí skript. Poté, co skript dokončí svůj úkol, získá zpět argumenty a aktualizuje podle nich dialog.

EAClientScripting se nemusí nalézat v jádře systému, ale díky přítomnosti knihovny EAScripting je přidání knihovny přímo do jádra zásadně rychlejší a neškodné, nebot' nově přidané funkce nelze volat z jiných klientských aplikací.

5.3 Tenký klient

Tenký klient získává všechny informace o formuláři, který chce zobrazit ve formě XML dokumentu (popsán v kapitole 3.9). V tomto dokumentu jsou popsané všechny komponenty formuláře, jejich vlastnosti, hodnoty a zda mají spouštět jakékoli skripty. V tomto dokumentu tedy tenký klient od serveru obdrží kód klientského skriptu.

Metoda jádra vytvářející XML dokument formuláře bude muset být upravena tak, aby v XML dokumentu vznikaly dva nové uzly:

- onInitScript – element v uzlu GBForm (viz kapitola 3.9). Obsahuje název skriptu a kód skriptu. Vznikne pouze pokud formulář obsahuje klientský skript, který se má spustit při inicializaci.
- onClickClient – element v uzlu Action, který se sám nachází v GBElement. Obsahuje název skriptu a kód skriptu. Vznikne pouze pokud je akce stisknutí tlačítka spojena se spuštěním klientského skriptu.

Po inicializaci formuláře, ale před tím, než se zobrazí, se klient ujistí, zda je s ním svázán skript, který se má spustit po jeho inicializaci. Pokud tomu tak je, pošle se skript knihovně EAClientScripting, která ho spustí.

Akci stisknutí tlačítka ve formuláři již poslouchá metoda, která zajišťuje, že server spustí skripty v jádře. Není problém rozšířit tuto metodu o schopnost rozpoznávat, zda se má spustit klientský skript nebo skript jádra. Pokud je s tlačítkem svázán klientský skript, předá se knihovně EAClientScripting a ta ho pak vykoná.

5.4 Webový klient

Webový klient nemůže na klientské straně využít knihovnu EAClientScripting. Její funkcionalita je nahrazena odpovídajícími JavaScript metodami. Mezi tyto funkce patří primárně gettery a settery pracující s daty otevřeného formuláře. Kvůli neobvyklému způsobu práce s checkBoxy a comboBoxy bude také zapotřebí vytvořit funkce pracující s nimi.

Všechny klientské skripty musí být v HTML dokumentu přítomné již v moment, kdy dorazí ke klientovi. Skript, který se má spustit při inicializaci formuláře, je vložen do <script> uzlu HTML a spustí se sám hned po načtení.

Skripty s dialogy jsou součástí <body> uzlu dokumentu. Každý dialog je uvnitř uzlu, který je při inicializaci neviditelný. Po stisknutí tlačítka se uzel se skriptem zviditelní a deaktivuje ostatní prvky stránky, aby s nimi nebylo možné manipulovat. Pro ukončení běhu skriptu se k dialogu přidá tlačítko pro jeho opětovné skrytí.

Webový klient využívá k získání formuláře z jádra stejnou metodu jako tenký klient (GetXml()), není proto potřeba nadále upravovat jádro. Rozdíl je až ve zpracování XML dokumentu na serveru.

Webový klient generuje HTML stránku pomocí technologie XSLT. Převádí tak XML dokument, který obdrží, na webový dokument. XSLT transformace se rozšíří o možnost vytváření tlačítek, které po stisku volají novou metodu na spuštění skriptu nebo zobrazení dialogu skriptu. Dále přidají do HTML kódu JavaScript metody, které slouží jako náhrada knihovny EAClientScripting.

5.5 EasyBuilder

Program EasyBuilder slouží ke správě databáze a jako takový má nástroje pro práci s daty v ní. Rozšíření jeho možností o práci s klientskými skripty se skládá pouze z replikace toho, jak pracuje s jiným elementárním prvkem systému. Například práce se souborem C# projektu skriptu jádra obsahuje všechny potřebné metody pro vytvoření, úpravu a smazání záznamu v databázi.

Úprava programu EasyBuilder se skládá z následujících akcí:

1. Vytvoření třídy ClientScript.
2. Vytvoření šablon SQL příkazů pro práci s databází.
3. Vytvoření dialogu pro práci s existujícími skripty.
4. Úprava dialogu pro propojení triggeru se skriptem.

V databázi je skript uložen pouze pomocí následujících hodnot:

- NAME – Jméno skriptu. Primární klíč.
- SCRIPT – Obsah skriptu. Jeho HTML a JavaScript komponenty.
- SKIND – Typ klientského skriptu (prozatím nepoužíván)
- ACL – Přístupová práva
- LANGID – Kódování textu skriptu (Když není vyplněno předpokládá se UTF-8)
- DESCRITION – Komentář ke skriptu. Slouží pouze pro zvýšení přehlednosti v rámci EasyBuilderu.

Jméno skriptu je primárním klíčem převážně kvůli tomu, že u skriptů jádra tomu tak dosud je a tato práce se snaží držet se postupů a pravidel, kterými se řídí celý systém EasyArchiv, aby nebyl budoucí vývoj zkomplikován tím, že se každé rozšíření, kterým systém prošel, vyvíjelo jiným způsobem.

5.6 Klientské skripty

Klientské skripty by měly být prakticky nerozeznatelné od HTML stránky s JavaScript kódem. Jediným rozdílem bude volání metod pro interakci s formulářem EasyArchivu. Mělo by tedy být možné je spustit ve webovém prohlížeči a do určité míry otestovat jejich funkcionalitu i mimo klienta.

Vzorové klientské skripty jsou k prohlédnutí v příloze tohoto dokumentu.

6 Implementace

Implementace řešení proběhla celkem v šesti krocích:

1. Implementace knihovny EAClientScripting – knihovna v jazyce C#, která spouští a zobrazuje klientské skripty. Knihovna je zaregistrována jako COM objekt.

2. Rozšíření databáze – do MySql a Oracle databází se pomocí SQL přidaly potřebné tabulky pro uchování klientských skriptů.

3. Rozšíření silného klienta – zdrojový kód silného klienta v C++ je rozšířen o EAClientScripting COM komponentu, EAClientScriptingWrapper a metody spouštějící klientské skripty.

4. Rozšíření tenkého klienta – zdrojový kód tenkého klienta a jeho serveru v C# je rozšířen o knihovnu EAClientScripting a načítání, přeposílání, zpracování a spouštění klientského skriptu

5. Rozšíření webového klienta – zdrojový kód serveru v jazyce Java je rozšířen o zpracování a přeposílání kódu klientského skriptu klientské aplikaci.

6. Rozšíření programu EasyBuilder – zdrojový kód v jazyce C# je rozšířen o možnost pracovat s tabulkou klientských skriptů v databázi a je přidáno grafické rozhraní k usnadnění uživatelské interakce.

Pořadí kroků bylo určené především logickou návazností jednotlivých částí systému a jejich dostupností. První byla vytvořena knihovna EAClientScripting, aby bylo možné ověřit proveditelnost spouštění klientských skriptů a zjistit jejich omezení.

Po vytvoření databázových tabulek následovalo rozšíření silného klienta. Ten byl z klientů první na řadě, protože již obsahoval skriptovací modul. Aby se klientské skripty co nejvíce podobaly skriptům spouštěným v jádře, byla jejich struktura založena na již používaných strukturách.

Na pořadí implementace tenkého a webového klienta nezáleželo. Administrativní program EasyArchiv byl rozšířen jako poslední, neboť se jedná pouze o uživatelské rozhraní a jeho funkce neměla na průběh práce velký vliv.

Implementace proběhla s následujícími verzemi programů:

- EasyArchiv 2.27.16.5
- LightClient 1.0.39.0
- WebClient 2.6.4
- EasyBuilder 2.0.7.12

6.1 EAClientScripting

Knihovna EAClientScripting do určité míry napodobuje knihovnu EAScripting, která obsluhuje skriptování v jádře. Je to z důvodu, aby se jednotlivé části systému EasyArchiv od sebe nelišily na základě toho, v jaké době a jakým programátorem byly vytvořeny. Knihovna tedy bude rozdělena na přibližně stejné části, její metody budou dodržovat stejný způsob pojmenovávání a k celé knihovně se bude přistupovat stejně, jako se přistupuje ke knihovně EAScripting.

Knihovna obsahuje následující třídy pro předávání argumentů klientskému skriptu:

- ClientScriptDocument – tato třída obsahuje slovník párů skládajících se z jmen komponent a jejich hodnot. Pomocí metod poskytuje přístup k hodnotám vyplněným ve formuláři a umožňuje je měnit.
- ClientScriptGridToGP – v současnosti nevyužitá třída, která slouží k předání souřadnic v datové tabulce a hodnoty v ní.
- ClientScriptArgs – třída, která v sobě nese instance ClientScriptDocument a ClientScriptGridToGP. Kromě nich také umožňuje nastavení dalších argumentů. Ty zastávají různé funkce na základě toho, jaká akce daný skript spustila.

Komponenta WebBrowser sloužící ke spuštění skriptu z jmenného prostoru System.Windows.Forms.WebBrowser umožňuje vystavení třídy ClientScritArgs přímo JavaScript kódu uvnitř skriptu pomocí proměnné ObjectForScripting. Klientský skript tak může přistupovat k libovolným veřejným metodám a proměnným třídy.

Aby knihovna mohla sloužit jako COM objekt, je potřeba upravit AssemblyInfo.cs soubor a vytvořit pro knihovnu klíč silného pojmenování (Strongly Named Key, SNK). SNK klíč obsahuje veřejný a soukromý klíč, jednoduché jméno a verzi knihovny. Zabezpečuje, že bude volána správná knihovna, a že ji nebude možné za běhu nahradit jinou.

Nastavení hodnot v AssemblyInfo.cs a cesta k SNK klíči:

```
[assembly: ComVisible(true)]
[assembly: AssemblyDelaySign(false)]
[assembly: AssemblyKeyFile("..\\..\\KeyFile.SNK")]
```

K zaregistrování knihovny do operačního systém Windows je dále zapotřebí otevřít vývojovou konzoli integrovaného vývojového prostředí Visual Studio ("Developer Command Prompt for VS2015") a spustit následující příkaz, který vytvoří odpovídající registry a TLB soubor:

```
RegAsm.exe EAClientScripting.dll
        /tlb:EAClientScripting.tlb /codebase
```

Knihovna typu (TLB soubor) je binární soubor uchovávající informace o COM objektu a jeho vlastnostech a metodách. Umožňuje ostatním aplikacím za běhu přistupovat k informacím o COM objektech.

Knihovna dokáže spustit dva základní typy klientských skriptů:

- skript s dialogem,
- skript bez dialogu.

Pokud je následující hlavička rozpoznána, jedná se o skript bez dialogu a knihovna pomocí regulérního výrazu vyextrahuje kód skriptu z uvnitř uzlu <script>, vloží ho do komponenty WebBrowser a spustí.

```
<scriptOnly script="jmeno skriptu"/>
```

6.2 Rozšíření databáze

V databázi se klientské skripty budou ukládat do tabulky EAClientScriptFiles.

```
CREATE TABLE EACLIENTSCRIPTFILES (
    NAME varchar(50) NOT NULL,
    SCRIPT text NOT NULL,
    SKIND int NOT NULL,
    ACL int NOT NULL Default (0),
    LANGID int NOT NULL Default (0),
    DESCRIPTION varchar(256) NULL
);
```

Sloupec „NAME“ je primárním klíčem, nemohou tedy existovat dva klientské skripty stejného jména.

Ve sloupci „SCRIPT“ je uložen celý text scriptu, tedy JavaScript i HTML část. Sloupec „DESCRIPTION“ slouží uživateli, který skripty vytváří, aby se v nich mohl lépe orientovat pomocí krátkého popisu. „ACL“ značí přístupová práva ke skriptu, „SKIND“ značí, o jaký typ skriptu se jedná a „LANGID“ identifikuje programovací jazyk, ve kterém je skript napsán.

Propojení klientského skriptu a události, při které se má spustit, proběhne v tabulce GPBinding, ve které se doposud propojovaly skripty jádra a události.

Již existující tabulka GPBinding obsahuje ID prvku, na který má reagovat, a ID formuláře, ve kterém se daný prvek nachází (pokud se má skript spustit po inicializaci formuláře, ID prvku se ponechá prázdné). Aby systém mohl rozpoznat, zda má spustit klientský skript nebo ne, vznikl nový typ akce (více o ActionType enumerátoru v jádře v kapitole 6.3). Tato akce je jako celé číslo uložena ve sloupci Action.

```
CREATE TABLE GPBindings(
        ControlID int NOT NULL,
        DocTypeID int NOT NULL,
        Message numeric(10, 0) NOT NULL,
        Action int NOT NULL,
        RefControl int NOT NULL,
        Parameter varchar(256),
        NotifyMessage numeric(10, 0) NOT NULL,
        NotifyCode int NOT NULL,
        NotifyItems varchar(200) NULL
);
```

6.3 Silný klient

Jak je popsáno v kapitole 3.3, EAScripting je COM objekt, který je v jádru používán díky tomu, že existuje wrapper, který zprostředkovává komunikaci mezi COM objektem a jádrem. Stejný wrapper existuje pro EAClientScripting COM objekt. Konkrétně se jedná celkem o čtyři třídy, které slouží jako wrappery pro čtyři třídy EAClientScripting knihovny (EAClientScriptingMain, ClientScriptGridToGP, ClientScriptDocument a ClientScriptArgs). Tyto wrappery obsahují C++ metody pro všechny potřebné metody uvnitř C# knihovny a obstarávají potřebné typové konverze.

Velká část rozšíření silného klienta replikuje již přítomné metody a volání. Cesta programem je stejná pro klientské skripty a skripty spouštěné v jádře, jedná se tedy především o přidání rozhodovací větve při rozhodování, jaký skript spustit a přidání enumerátorů identifikujících typy akcí a skriptů.

Při vyvolání akce se zavolá metoda OnCommand třídy GenericPropPageClient, která načte všechny skripty související s prvkem, který danou událost vyvolal, a v závislosti na typu skriptu zavolá odpovídající metodu. Enumerátor sloužící k identifikaci typu skriptu se nazývá ActionType. Jeho hodnota je uložená v databázi v tabulce GPBinding a pro dva typy skriptů nabírá hodnot atUserAction (s hodnotou 5) a asUserClientAction (s hodnotou 8).

Při volání skriptů je další metodou UserAction nebo UserClientAction. Tyto metody připraví data rozhodující o tom, jaký konkrétní skript se spustí a v jakém kontextu (který formulář ho zavolal). Proces pak pokračuje do metody DoExtWork, která inicializuje potřebné knihovny a zavolá dodispatch metodu. Ta zavolá Run nebo RunOnClient metodu, která již obstarává samotné získání skriptu, vytvoření instancí s argumenty a zavolání odpovídající knihovny, která skript vykoná. Poté, co klientský skript doběhne, má také za úkol nahrát skriptem upravené hodnoty zpět do formuláře.

Jelikož je knihovna EAClientScripting volána přímo z jádra jako COM objekt, je potřeba nastavit cestu k ní v konfiguračním souboru EasyArchivu.

```
<configuration>
  <runtime>
    <assemblyBinding
        xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity
```

```
          name="EAClientScripting"
          publicKeyToken="a78b7c88d57e1af6" />
        <codeBase
          version="1.0.0.1"
          href="file:///C:/EAClientScripting.dll"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

6.3.1 Úprava jádra pro servery

V jádře systému se nalézá metoda GetXML, která vytváří XML dokument, ve kterém se serverům předávají vlastnosti formuláře. Metoda postupně prochází všemi elementy formuláře a ukládá jejich vlastnosti ve formě XML.

Tato metoda je upravena, aby ke každému prvku přidala jednu vlastnost, kterou dříve neměly. Jedná se o FieldName prvku. Tato hodnota se musela přidat do XML dokumentu, protože v něm dříve nebyla, ale skripty přistupují s datům pomocí jmen prvků. Bylo by možné donutit skripty přistupovat k datům například pomocí jejich ID, ale opět se jedná o snahu, aby byly klientské skripty co nejpodobnější skriptům spouštěným v jádře, alespoň co se uživatelské obsluhy týká.

Do metody je dále přidána možnost vložení elementu <oninit>, který v sobě obsahuje klientský skript, který se má spustit při inicializaci formuláře. Element <oninit> vznikne pouze, pokud formulář obsahuje skript ke spuštění.

Během vytváření XML dokumentu se volají GetXML metody jednotlivých prvků formuláře. Mezi nimi je i volání ve třídě ControlerItemWebButton, které vytváří část XML týkající se tlačítek. Pokud je k tlačítku přiřazen klientský server, vznikne v XML dokumentu uzel <onclinckClient>, ve kterém je klientský skript a jeho jméno.

Skripty se do XML dokumentu vkládají zakódované ve formě Base64, aby nepůsobily problémy se strukturou dokumentu.

6.4 Tenký klient

Knihovna EAClientScripting je vložena do tenkého klienta jako reference. Lze tedy vytvářet instance jejích tříd a volat jejich metody bez potřeby implementace jakéhokoli wrapperu.

Ihned poté, co klientská aplikace získá XML dokument se strukturou formuláře od serveru, vytvoří podle informací v něm formulář. Klientské skripty jsou během tohoto procesu uloženy do slovníku, kde je lze najít podle jména prvku, který je spustí. Vždy, když se stiskne tlačítko, které má spustit skript, se prohledá slovník a všechny odpovídající skripty se spustí.

Klientský skript, který se má spustit hned po inicializaci formuláře, se automaticky spustí poté, co se dokončí zpracování dokumentu a inicializace formuláře.

Jak bylo již zmíněno, prvky v klientských aplikacích jsou adresovány pomocí jejich ID v systému EasyArchiv. Proto byla do XML dokumentu přidána položka, která nese jméno (FieldName) prvku. Prvky, se kterými tenký klient vytváří formuláře, jsou ovládacími prvky knihovny System.Wondows-.Forms a nemají tedy místo pro další identifikátor. Jednou možností je zdědit všechny možné použitelné prvky a v jejich potomcích vytvořit nový identifikátor, pomocí kterého by je bylo možné rozpoznat. Místo toho je v každém formuláři slovník, který se v době zpracování XML dokumentu naplní ID prvků a odpovídajícími FieldName hodnotami. Je tak možné kdykoli získat jméno prvku pro klientské skripty.

6.5 Webový klient

Webový klient, stejně jako tenký, obdrží data od jádra ve formě XML dokumentu. Místo toho, aby se zpracovávaly na klientské straně, se však XML rozebírá na straně klienta. Klientská strana webového klienta je webový prohlížeč a jeho možnosti jsou omezené, proto server vytvoří z XML dokumentu HTML dokument a až ten přepošle klientovi.

Server vytváří HTML dokument prvek po prvku včetně neviditelných formulářů klientských skriptů. Klientské skripty, které nemají formulář jsou také uchovány uvnitř neviditelného prvku a jejich kód se spustí spolu s akcí zviditelnění prvku.

Stejně jako u předchozího klienta, i zde je potřeba přidat FieldName jméno všem prvkům, které by mohly chtít adresovat klientské skripty. V HTML je toto jednoduché a všechny takové prvky mají přiřazen nový atribut s názvem FieldName, ve kterém je jméno uloženo a podle kterého lze prvky vyhledávat.

V tomto klientovi je knihovna EAClientScripting kompletně nepoužitelná. Je tedy potřeba její kód zreplikovat v JavaScriptu, který bude spustitelný ve webovém prohlížeči. Jedná se o vytvoření metod, které získávají data z aktuálně otevřeného formuláře a metod, které mění data formuláře. V okamžik, kdy je ve webovém prohlížeči spuštěn klientský skript, otevřený formulář je uživateli nedostupný, ale je stále otevřený na pozadí. Metody pro manipulaci s ním jsou tedy jen metody, které získají data z prvků aktuálně otevřeného HTML dokumentu. Konkrétní prvky najdou podle hodnoty FieldName.

JavaScript kód, který získá prvek dokumentu podle hodnoty jeho FieldName atributu, je následující:

```
document.querySelector('[FieldName="element name"]');
```

6.6 Easy Builder

Program EasyBuilder pracuje s databází za pomoci třídy System.Data.Odbc.OdbcCommand. Instance této třídy slouží jako SQL příkazy. Vytvoří je jako šablony ,které jsou později naplněny daty a spuštěny. Šablony pro práci vypadají takto:

```
UPDATE EACLIENTSCRIPTFILES SET SCRIPT=?, SKIND=?,
        ACL=?, LANGID=?, DESCRIPTION=? WHERE NAME=?

INSERT INTO EACLIENTSCRIPTFILES (NAME, SCRIPT,
        SKIND, ACL, LANGID,     DESCRIPTION)
        VALUES (?,?,?,?,?,?)

DELETE FROM EACLIENTSCRIPTFILES WHERE NAME=?
```

Program si po připojení k databázi stáhne do paměti celou tabulku EACLIENTSCRIPTFILES a uživatel pak může s daty pracovat. Pokud dojde k jakýmkoli změnám a uživatel se je rozhodne uložit, vytvoří se výše zmíněné šablony, naplní se daty a změny se uloží do databáze.

Klientské skripty jsou v programu EasyArchiv reprezentovány třídou, která dědí od třídy DataObject. Získává tak další funkce, které jsou dále zmíněné v kapitole 9.4.

Program byl také rozšířen o grafické rozhraní umožňující uživateli vytvářet, upravovat a mazat klientské skripty.

Dialog sloužící k vytváření vztahu mezi skriptem a ovládacím prvkem, který ho spustí, umožňuje navázat klientské skripty na prvky a události.

6.7 Překážky ve vývoji

Hlavní překážkou ve vývoji byla rozmanitost využívaných technologií a mohutnost systému EasyArchiv. Nemluvě o použitých jazycích (C++, C#, Java, JavaScript), systém se skládá z různých technologií (Tomcat, COM, XSL, Webové služby), které byly zakomponovány do systému v různých obdobích různými lidmi.

Práci dále často znepříjemňovaly různé způsoby používání identické funkcionality. Toto se primárně týkalo webového klienta, který svou podstatou funguje jinak než ostatní klientské aplikace.

6.8 Zhodnocení

Řešení splňuje cíle ustanovené v kapitole 5 Navržené řešení. Na všech třech klientských programech se spouští klientské skripty v reakci na uživatelské akce (otevření formuláře nebo stisknutí tlačítka). Tyto skripty dokáží manipulovat s daty v otevřeném formuláři.

V programu EasyBuilder lze klientské skripty vytvářet, navázat na ovládací prvky formuláře a uchovat v databázi.

6.8.1 Možnosti implementovaného řešení

- Dva typy skriptů – lze vytvořit skript, který upraví data uvnitř formuláře a poté se ukončí, nebo skript, který otevře vlastní dialog umožňující

uživatelskou interakci.

- Přístup ke všem datům ve formuláři – pomocí skriptu lze upravit/vyplnit/smazat jakákoli data uvnitř formuláře, ze kterého byl spuštěn (s omezením prvku ComboBox viz. níže)
- Navázání na onInit nebo onClick akce – spustit skripty lze v reakci na dvě různé akce. Na otevření formuláře a na stisknutí tlačítka uživatelem.
- Široké možnosti upravení dialogu skriptu – technologie HTML a CSS spolu umožňují tvorbu dialogu z velkého výběru ovládacích prvků a vizuálních úprav.
- Vkládání obrázků a diagramů do dialogu skriptu – do HTML dialogu lze vložit obrázky, diagramy nebo grafy zakódované v Base64 řetězci znaků. Lze tak do skriptu vkládat například návody na vyplňování formulářů, postupy pro zpracování dokumentů nebo kontaktní informace. To vše bez potřeby obrázek jakkoli adresovat nebo kopírovat na každé klientské zařízení, které ho chce zobrazit.
- Přístup k JavaScript funkcím - lze přistupovat k funkcím JavaScriptu a jednoduše tak získat například aktuální čas, hodnotu Π nebo používat matematické funkce.

Výhodou JavaScriptu je, že ho prakticky každý programátor ve své praxi využil, není proto potřeba učit uživatele obskurní skriptovací jazyky. Jeho syntaxe je také velice podobná jazykům jako C#, takže i méně zkušený programátor by neměl mít problém se naučit základy.

6.8.2 Omezení implementovaného řešení

Jelikož nejsou klientské skripty spouštěné v jádře, nemají přístup k jeho funkcím. To znamená, že nemohou požadovat od systému data, která již nejsou obsažena ve formuláři, měnit vlastnosti formuláře nebo jakkoli manipulovat se systémem. Sice se jedná o omezení schopností klientských skriptů, ale jelikož je možné ve webovém klientovi skripty před spuštěním upravovat, jedná se zároveň o bezpečnostní opatření.

Standardně se skripty spouští každý ve vlastním kontextu. Jen v rámci webového klienta je zapotřebí, aby byly všechny od momentu vygenerování

ve stejném HTML dokumentu. Toto může zapříčinit problém, pokud existují skripty s konfliktními názvy. Vytváření názvů metod a proměnných jsou však na uživateli, a pokud se tak nezařídí, může nastat situace, že bude volána jiná metoda, než kterou uživatel zamýšlel. Řešením by bylo pojmenovat funkci, která spouští skript, stejně jako celý klientský skript, jehož jméno je unikátní, a zbytek JavaScript kódu skrýt uvnitř jmenného prostoru. Toto je však volba pouze autora skriptu a nebylo by praktické ji programově vynucovat.

Ze stejného důvodu může nastat konflikt CSS. Proto je lepší nepoužívat CSS styly, které by mohly způsobit konflikty se styly používané Webovým klientem.

Klientské skripty dokáží měnit hodnoty uvnitř prvku ComboBox, který nabízí výběr z několika předem nastavených hodnot. Jelikož jsou ale tyto hodnoty uložené v databázi a změna výběru probíhá uvnitř jádra, nelze pomocí skriptů změnit hodnotu ComboBoxu tak, aby po uložení zůstala změněná.

6.8.3 Možnosti dalšího rozšíření

Jedním z možných rozšíření by bylo přidání AJAX funkcionality do klientských skriptů. Bylo by tak možné zpřístupnit funkcionalitu jádra skriptům spolu s určitými omezeními, aby nebyla narušena bezpečnost. Klientské skripty byt tak mohly dynamicky získávat data ze systému a měnit vlastnosti formulářů.

Řešení by potencionálně mohlo využívat knihoven jako Bootstrap nebo jQuery. Bootstrap by umožnil ještě více uživatelské svobody v oblasti vytváření vizuálně zajímavých a flexibilních dialogů, ale kvůli rozmanitým vzhledům by pak dialogy nevypadaly že jsou součástí systému. Bootstrap by navíc neposkytoval užitečnou funkcionalitu. Knihovna jQuery by ulehčila práci s HTML prvky v rámci klientského dialogu, ale zároveň by zjednodušila přístup k jiným prvkům webového klienta, což je nežádoucí.

Stejně jako klientské skripty dokážou pracovat s hodnotami prvků formulářů, mohli by také pracovat s vlastnostmi těch prvků. Bylo by tak možné měnit velikosti, pozice a styly ovládacích prvků i bez přidání AJAX funkcionality. Toto rozšíření by však vyžadovalo vlastní implementaci pro každého klienta zvlášť a změny by zmizely po uzavření formuláře. Jelikož se nejedná o obzvlášť užitečnou funkci s poměrně obtížnou implementací, není prioritou.

Poměrně užitečným rozšířením by bylo přidání možnosti spouštět skripty v jádře pomocí klientských skriptů. Bez potřeby velkého rozšiřování současného řešení by byly zpřístupněny funkce jádra, které nejsou aktuálně dostupné. Jednalo by se nejspíš o zavolání serveru, pravděpodobně pomocí AJAXu, který by poté v jádře spustil odpovídající skript. Ten by byl však vykonán a upravil by formulář. Neposlal by žádná data klientskému skriptu. Nebylo by tak možné získávat data, nebo je nastavovat, protože vstupními parametry skriptů v jádře je vždy formulář. Teoretickou možností by bylo spuštění klientského skriptu, který by se ukončil, spustil skript v jádře a po jeho ukončení opět spustil sebe, ale v takové situaci by bylo pravděpodobně jednodušší implementovat rozšíření s AJAXem.

Program EasyBuilder umožňuje uživatelům vytvářet formuláře pouze umisťováním ovládacích prvků do prostoru a volbou jejich atributů a vlastností z předem stanovené množiny. Tato funkce by mohla být rozšířena o možnost vytvářet HTML dokumenty. Bylo by tak možné jednoduše vytvářet grafickou část klientských skriptů. Jedná se však o rozšíření pochybné užitečnosti.

7 Testování

Cílem testování je ověření správného fungování nových funkcí na všech třech klientech a v EasyBuilderu. Ověřeny musí být následující funkce:

- Vytvoření, úprava a smazání klientského skriptu v programu EasyBuilder.
- Vytvoření, úprava a smazání propojení klientského skriptu s formulářem a prvkem v programu EasyBuilder.
- Spuštění klientského skriptu s dialogem.
- Spuštění klientského skriptu bez dialogu.
- Ověření, že za běhu klientského skriptu nelze manipulovat s formulářem, ze kterého byl skript spuštěn a že po skončení skriptu opět lze.
- Načtení, úprava a smazání dat prvků formuláře (vyjma prvku ComboBox) pomocí klientského skriptu.
- Uložení formuláře s upravenými daty.
- Ověření, že uživatel může svými akcemi ovlivňovat výsledek běhu klientského skriptu.

Testy míněné pro klienty musí být spuštěné na všech třech typech klienta. Testování musí být provedeno pro oba následující případy:

- Data systému jsou uložena v MySql databázi.
- Data systému jsou uložena v Oracle databázi.

7.1 Metodika testování

Systém EasyArchiv v současnosti nevyužívá žádný framework k automatizaci testování a žádně automatické testy neobsahuje. Z tohoto důvodu, a protože automatizace testů, které vyžadují uživatelskou interakci, by byla v tomto případě komplikovaná, budou testy prováděny do určité míry manuálně.

Celý testovací proces bude spuštěn dvakrát. Jednou na systému s MySql databází a jednou na systému s Oracle databází. V obou případech budou testy muset ověřit všechny testované vlastnosti.

7.1.1 EasyBuilder

Testování týkající se programu EasyBuilder nelze jakkoli automatizovat. Tester bude muset manuálně provést následující kroky:

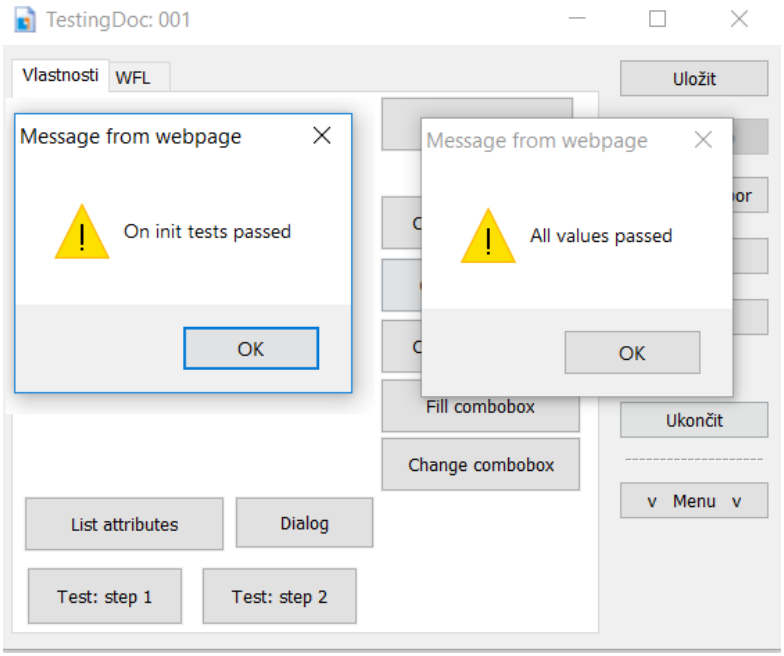
1. Vytvořit klientský skript a ověřit v databázi jeho korektní vznik.
2. Upravit klientský skript a ověřit v databázi, že se změnil.
3. Nastavit spuštění klientského skriptu na inicializaci testovacího formuláře a ověřit, že v databázi vznikla odpovídající hodnota.
4. Změnit akci spouštějící klientský skript na spuštění po stisku tlačítka a ověřit v databázi, že změna proběhla.
5. Smazat akci spouštějící skript a ověřit v databázi, že skript nebude spouštěn.
6. Smazat klientský skript a ověřit v databázi, že byl smazán.

Mnohé kroky testování lze místo v databázi ověřit spuštěním skriptu pomocí klientské aplikace. To však zavádí možnost falešně pozitivního výsledku (test je označen, že neprošel, ačkoli program EasyBuilder funguje správně) chybou v klientské aplikaci. Ověřování správných hodnot v databázi je preferované.

7.1.2 EasyArchiv

Testování klientských aplikací lze do určité míry automatizovat pomocí klientských skriptů. Do formuláře se všemi testovanými ovládacími prvky se vloží skripty, které automaticky provedou testované akce a zároveň je vyhodnotí. Testování bude probíhat následovně:

1. Tester otevře testovací formulář.
2. Pokud se úspěšné spustí onInit skript, zobrazí se informační dialog.
3. Formulář obsahuje popis, jak postupovat. Uživatel vyplní do ovládacích prvků požadované hodnoty a stiskne tlačítko. Tento test spouští klientský skript bez dialogu a testuje načítání a zapisování hodnot do kontrolních prvků. Po jeho skončení se zobrazí informační dialog.
4. Další tlačítko, které uživatel stiskne, zobrazí skriptovaný dialog, ve kterém uživatel dle pokynů vyplní požadované hodnoty a dialog uzavře.
5. Poslední tlačítko spustí skript, který ověří, že všechny hodnoty byly správně načtené ze skriptovaného dialogu.
6. Posledním krokem je uložení dialogu a jeho opětovné otevření, aby byla ověřena perzistence změn.



Obrázek 7.1: Dialogy splněných testů

7.2 Průběh testování

Testování bylo prováděno na počítači Dell Latitude E5520 s verzí systému EasyArchiv 2.27.16.5. Databáze systému byly starší zálohy, které slouží k testování.

Během vývoje testovacích skriptů se osvědčilo pojmenovávat funkcí a proměnných skriptů pomocí prefixů nebo sufixů se jménem skriptu. Minimalizuje se tak šance, že dojde na webovém klientovi ke kolizi jmen.

Parametry testovacího počítače jsou následující:

- 4,0 GB DDR3 RAM
- INTEL Core i5 Sandy Bridge
- 64bitový Microsoft Windows 7 Professional

7.3 Výsledky testování

Všechny testy proběhly úspěšně. Hodnoty v databázích se měnily přesně podle očekávání, skriptovací dialogy byly interaktivní a měnily obsah formuláře dle očekávání a klientské skripty bez dialogů proběhly bez problémů a s neznatelnou odezvou.

8 Závěr

Cílem této diplomové práce bylo rozšíření systému EasyArchiv o možnost vytváření a spouštění skriptů, které by byly schopné uživatelské interakce. Tento cíl byl splněn.

Prvním krokem byla zběžná analýza systému EasyArchiv. Jejím cílem bylo seznámení se se systémem, prozkoumání použitých technologií a omezení, která kladou na budoucí rozšíření. Následoval průzkum metod skriptování. Byly zváženy jejich silné a slabé stránky a to, jak by celkově fungovaly v systému. Hlavním cílem tohoto projektu bylo skloubení nového rozšíření skriptování s existujícími programy, pokud možno tak, aby nebyla jakkoli ovlivněna současná funkcionalita a aby bylo nasazení nového systému bezproblémové. Program je tedy navržen tak, aby nenutil uživatele ho využívat, ale tak, aby uživateli poskytoval rozšířené možnosti jak interagovat s celým systémem nezávisle na tom, z jakého programu k němu přistupuje.

Výsledkem této práce jsou 4 rozšířené programy. Program EasyBuilder, který umožňuje tvorbu klientských skriptů a jejich propojení s formuláři, a tři klientské aplikace, každá schopná spouštění klientských skriptů ve formulářích.

Klientské skripty dokáží pracovat s obsahem formuláře a zobrazovat vlastní, uživatelem definované dialogy se spustitelným JavaScript kódem.

Cíl této práce byl splněn a rozšířený systém lze využít, aniž by byla omezena dřívější funkcionalita systému.

Literatura

[1] TD-IS, EasyArchiv/EasyPLM® - Charakteristika, dne 28.4.018, http://www.td-is.cz/cs/produkty/easyarchiv/charakteristika.htm

[2] Microsoft, Windows Dev Center - Component Object Model (COM), dne 28.4.018, https://msdn.microsoft.com/en-us/library/windows/desktop/ms680573(v=vs.85).aspx

[3] Microsoft, Windows Dev Center - PowerShell), dne 28.4.018, https://docs.microsoft.com/en-us/powershell/scripting/

[4] Microsoft, Windows Dev Center - Using the ScriptControl, dne 28.4.018, https://msdn.microsoft.com/en-us/library/aa227633(v=vs.60).aspx

[5] Microsoft, Windows Dev Center, dne 28.4.018, https://developer.microsoft.com/en-us/windows

[6] The Apache Software Foundation, Apache Tomcat, dne 28.4.018, http://tomcat.apache.org

[7] Dan Adler, The JACOB Project: a JAva-COM Bridge, dne 28.4.018, http://danadler.com/jacob/

[8] Microsoft, .NET Documentation, dne 28.4.018, https://docs.microsoft.com/en-us/dotnet/

[9] Javascript .Net, JavaScript.net Documentation, dne 28.4.018, https://github.com/JavaScriptNet/Javascript.Net

[10] ClearScript, ClearScritp GitHub page, dne 28.4.018, https://github.com/Microsoft/ClearScript

[11] Google, Chrome V8 engine, dne 28.4.018, https://developers.google.com/v8/

[12] Joseph Albahari, C# 6.0 in a Nutshell: The Definitive Reference, šestá edice

[13] Kyle Simpson, You Don't Know JS: Up & Going, první edice

[14] Max Ogden, JavaScript For Cats, dne 28.4.018, `http://jsforcats.com/`

9 Přílohy

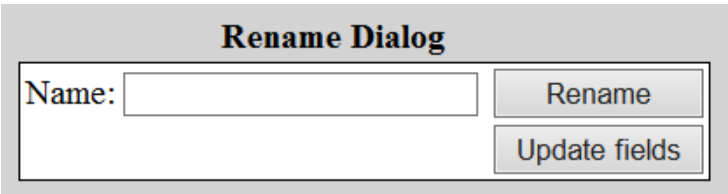
9.1 Slovník

- AJAX – Způsob jak mohou webové stránky získávat data ze serveru, aniž by se museli obnovit.
- C# – Vysokoúrovňový objektově orientovaný programovací jazyk.
- C++ – Multiparadigmatický programovací jazyk.
- CSS – Jazyk pro popis způsobu zobrazení elementů na webových stránkách.
- Component Object Model – Systém pro vytváření binárních komponent schopných interakce.
- Common Language Infrastructure – Technický standard popisující spustitelný kód a prostředí umožňující spuštění na různých platformách.
- Document Management System – Systém pro správu dokumentů.
- EasyArchiv – Systém pro správu dokumentů od společnosti TD-IS.
- EasyCom – Jádro systému EasyArchiv.
- EasyBuilder – Administrativní nástroj systému EasyArchiv.
- GUID – Globální, unikátní identifikátor.
- HTML – Značkovací jazyk pro tvorbu webových stránek.
- Java – Objektově orientovaný programovací jazyk.
- JavaScript – Multiplatformní, objektově orientovaný, skriptovací jazyk.
- MySQL – Systém řízení báze dat.
- Oracle – Systém řízení báze dat.
- Open-source – Program s otevřeným zdrojovým kódem.
- Powershell – Shell příkazové řádky sloužící k administraci systému.
- Product Lifecycle Management – Proces řízení životního cyklu výrobku.

- Relace (session) – Trvající síťové spojení mezi klientem a serverem.
- Skript – Program sloužící k automatizaci procesu v rámci jiného programu.
- SQL – Standardizovaný strukturovaný dotazovací jazyk pro práci s daty.
- Tomcat – Open-source webový server.
- Workflow – Pracovní nebo technologický postup.
- Wrapping – Obalování objektu třídou k usnadnění manipulace s ním.
- WebService – Systém umožňující interakci dvou strojů po síti.
- Webový prohlížeč – Počítačový program, který slouží k prohlížení webových stránek.
- XSLT – Jazyk sloužící k transformaci XML dokumentů.
- XML – Obecný značkovací jazyk.

9.2 Příklad klientského skriptu

Vzorový skript s nadefinovaným dialogem. Tlačítko „Update fields“ načte jméno dokumentu z formuláře a vloží ho do textového pole „Name“. Tlačítko „Rename“ změní jméno dokumentu na „NEW NAME“.



Obrázek 9.1: Okno vzorového klientského skriptu.

```
<head>
<title>Rename dialog</title>
<style>
div.sized {
width: 360px;
```

```
height: 50px;
}
</style>
</head>
<body style="background-color:lightgrey;">
<div class="sized" >
<center style="margin-bottom: 3px;">
<b>Rename Dialog</b>
</center>
<div style="border: 1px solid black;"
        style="background-color:white;">
<table>
<tr>
<td>Name:</td>
<td><input type="text" name="allInputs"
        id="cgdocument.doccomment_script"
        onchange="updateValue_demo1()"/>
</td>
<td><button type="button"
        style="margin:0px,0px,0px,5px;width:120px;"
        onclick="RunRename_demo1()">Rename
</button>
</td>
</tr>
<tr>
<td></td>
<td></td>
<td><button type="button"
        style="margin:0px,0px,0px,5px;width:120px;"
        onclick="updateFields_demo1()">Update fields
</button>
</td>
</tr>
</table>
</div>
</div>
</body>
<script>
function GetValue(name) {
return window.external.Document.
        GetValueInternal(name); }
```

```
function SetValue(name, newValue) {
window.external.Document.
        SetValueInternal(name, newValue); }
function GetComboBoxValues(name) {
return GetValue(name + '.list'); }
function SetComboBoxValues(name, newValue) {
SetValue(name + '.list', newValue); }
function AddValueToComboBox(name, newValue) {
SetComboBoxValues(name, GetComboBoxValues(name)
         + ", " + newValue) }
function ClearComboBox(name) {
SetComboBoxValues(name, '') }

function updateFields_demo1() {
document.getElementById
        ('cgdocument.doccomment_script').value
        = GetValue('cgdocument.doccomment');
}
function updateValue_demo1() {
SetValue('cgdocument.doccomment',
document.getElementById('cgdocument.
        doccomment_script').value);
}
function RunRename_demo1() {
SetValue('cgdocument.doccomment','NEW NAME');
updateFields_demo1();}
</script>
```

9.3 Rozsah úprav

V rámci této práce byla vytvořena knihovna EAClientScripting a byly provedeny úpravy v níže vyjmenovaných třídách a projektech.

Změny v rámci silného klienta a jádra jsou následující:

- přidána třída EAClientScriptingWrapper
- a upraveny třídy EAScriptingEngineImpl,
- EAScriptingEngine,

- GenericPropWebPage,
- GenericPropPageClient,
- baseConfig,
- ControlItems,
- ControlItemsClient,
- ControlItemWeb,
- ControlItemWebButton
- a ControlsInterface.

Změny v rámci tenkého klienta jsou následující:

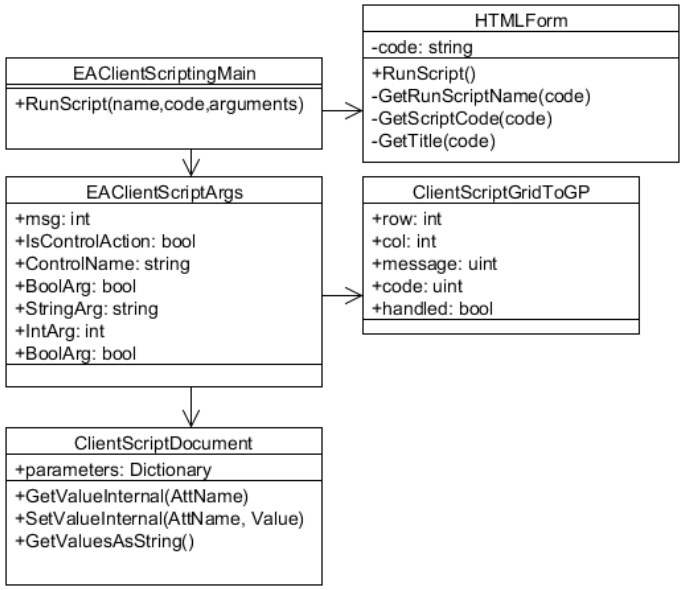
- upraveny třídy frmEADocument
- a EACommons.

Změny v rámci webového klienta jsou následující:

- upravena třída EADocGenPageTag
- a upraven soubor _genpage.xsl.

Změny v rámci programu EasyBuilder jsou následující:

- přidány třídy AddClientScriptForm,
- ClientScript
- a SqlEaClientScripting
- a upraveny třídy GPBindings,
- GPBindingsEditor,
- GPBindingsSetForm,
- Scripting
- a SqlGPBind.

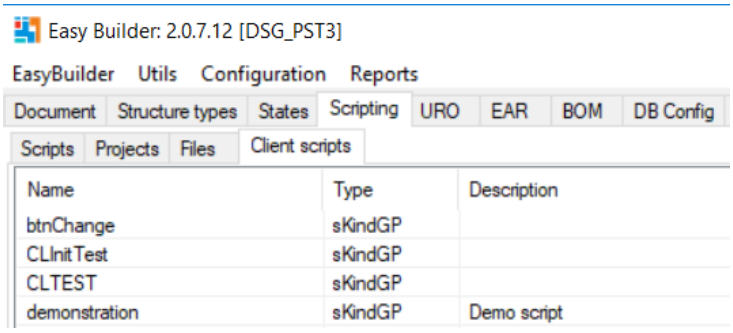


Obrázek 9.2: Diagram tříd knihovny EAClientScripting

9.4 Uživatelská dokumentace

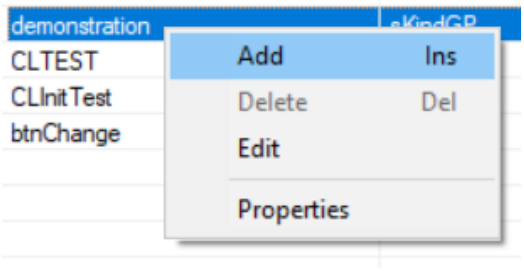
Uživatelská dokumentace se zaobírá pouze vytvářením klientských skriptů, nikoli jejich spouštěním, nebo obecně používáním systému EasyArchiv.

Po spuštění nakonfigurovaného programu EasyBuilder a nahrání dat z databáze (klávesa F3) uživatel zvolí záložku „Scripting“ a pod-záložku „Client scripting“. Zde jsou v seznamu vypsány všechny klientské skripty nezávisle na tom zda jsou použité, nebo ne.



Obrázek 9.3: Seznam klientských skriptů.

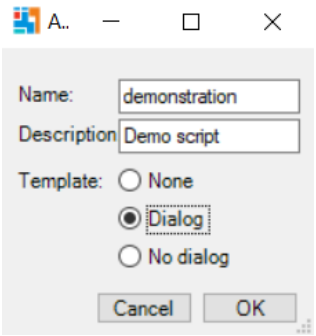
Poté, co uživatel klikne pravým tlačítkem myši do seznamu (na skript, nebo do prázdného prostoru), objeví se nabídka na obrázku 9.4. Tato nabídka umožňuje uživateli skript upravit („Edit“) nebo vytvořit nový („Add“). Možnost „Properties“ zobrazí dialog se všemi hodnotami skriptu a umožní je upravit.



Obrázek 9.4: Kontextové menu seznamu klientských skriptů.

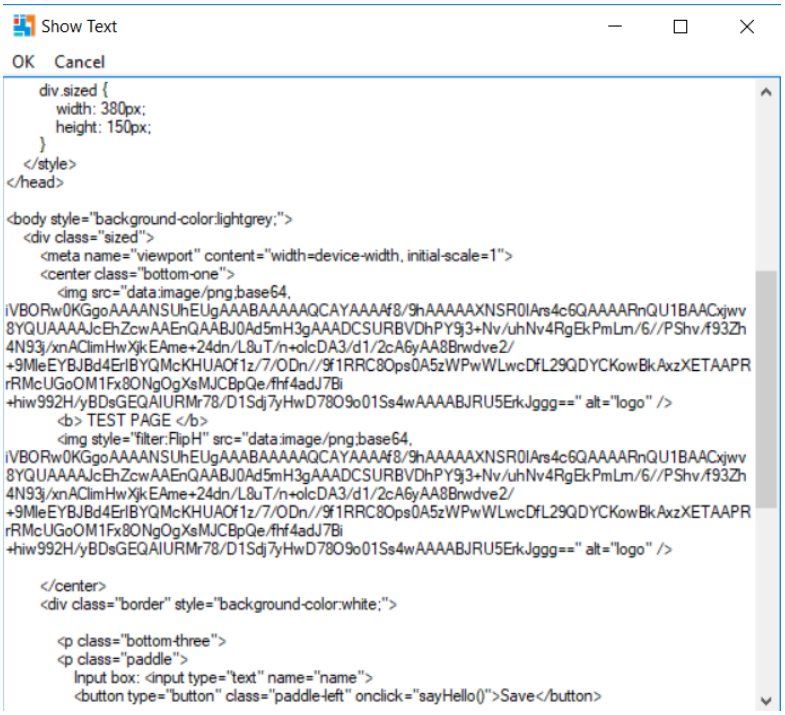
Když uživatel zvolí vytvoření nového skriptu, zobrazí se dialog na obrázku 9.4. Zde dochází k pojmenování skriptu („Name“), vytvoření popisku skriptu („Description“) a volba šablony („Template“) předvyplní obsah skriptu, aby uživatel nemusel sám vytvářet některé základní, nezbytné metody.

Na obrázku 9.4 je vidět editor vytvořeného klientského skriptu. Jedná se o jednoduchý textový editor a od uživatele se očekává, že skript vytvoří v libovolném vývojovém prostředí a poté ho pouze zkopíruje do editoru EasyBuilderu.



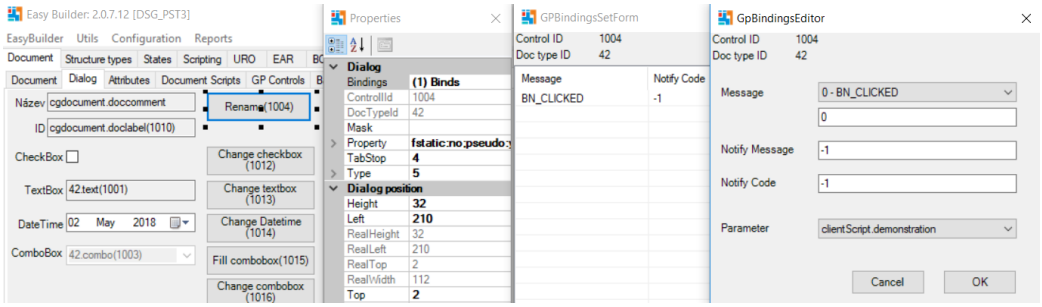
Obrázek 9.5: Dialog vytváření klientského skriptu.

Po vytvoření klientského skriptu se uživatel přepne do záložky „Document“, zvolí ze seznamu formulář, ke kterému chce skript připojit a dvojklikem ho otevře. Zobrazí se editační prostředí formuláře. Uživatel pravým tlačítkem klikne na prvek formuláře, nebo formulář samotný, a zvolí „Properties“. V menu, které se objeví je položka „Bindings“, která po otevření zobrazuje všechny skripty spojené se zvoleným prvkem. Obsahuje také tlačítka „Add“ pro přidání skriptu k prvku a „Remove“ pro jeho odstranění. Uživatel stiskne tlačítko „Add“ a otevře se dialog pro vytvoření vztahu mezi prvkem a skriptem („GpBindingsEditor“). Hodnota „Message“ značí typ akce, na kterou se jako reakce má spustit skript. Hodnota „Parameter“ je konkrétní skript, který se má spustit.



Obrázek 9.6: Editor kódu klientského skriptu.

Poté, co uživatel tyto hodnoty zvolí, může svou volbu potvrdit tlačítkem „OK“. Dialog se seznamem skriptů zavře stejným způsobem. Dialogy použité v tomto odstavci lze vidět na obrázku 9.4.



Obrázek 9.7: Dialogy Jednotlivých kroků propojování skriptu a prvku formuláře.

V tomto stádiu vytváření a propojení skriptu s prvkem formuláře již stačí jen uložit změny (např. stiskem klávesy F2).