VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INFORMAČNÍ SYSTÉM PRO SPRÁVU PROJEKTŮ

INFORMATION SYSTEM FOR PROJECT ADMINISTRATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JOSEF SYSEL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/20494/2017/xsysel07

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Sysel Josef**
Obor: Informační technologie
Téma: **Informační systém pro správu projektů**
Information System for Project Administration
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s principy tvorby informačních systémů na webu a existujícími řešeními IS pro správu projektů.
2. Analyzujte požadavky na informační systém správy projektů pro OSVČ. Zákazníci zde budou zadávat své objednávky, které budou přiřazovány řešitelům. Součástí bude také rezervační systém, kalendář úkolů, možnost organizace práce a času na projektech, možnost online konverzace a úložiště soukromých i sdílených souborů. Srovnejte požadavky s existujícími řešeními.
3. Navrhněte informační systém dle uvedených požadavků.
4. Navržený informační systém implementujte a ověřte jeho funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Literatura:

- Welling, L., Thomsonová, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Práce se zabývá tvorbou webového informačního systému pro správu projektů. Výsledná aplikace by měla pomoci OSVČ nebo menším podnikatelům s organizací práce, času a koordinací lidí, kteří s nimi spolupracují. Systém, umožňuje zákazníkům vytvářet objednávky v podobě nového projektu, k němuž potom administrátor přiřazuje hlavního řešitele, který pak následně rozděluje úkoly mezi jednotlivé řešitele. Nedílnou součástí systému je také možnost online konverzace mezi registrovanými uživateli, kalendář s probíhajícími projekty a úkoly, statistiky odvedené práce a nahrávání soukromých či sdílených souborů. Webová aplikace je implementována v prostředí ASP.NET s využitím Microsoft SQL Serveru a eXpress App Frameworku od společnosti DevExpress.

Abstract

This bachelor thesis deals with the creation of web-based information system for project management. The application should help a self-employed person or small businessman with the organization of the work, time, and coordination of people who cooperate with him. The system allows customers to create order in the form of a new project, to which then an administrator assigns a main worker who after that subsequently distributes tasks to other available workers. Also, an important part of the system is the option of online conversation between registered users, calendar with ongoing projects and tasks, statistics of work and recording of private or shared files. The Web application is implemented in the ASP.NET environment with using of Microsoft SQL Server and eXpress App Framework by DevExpress.

Klíčová slova

Informační systém, IS, webová aplikace, eXpress App Framework, XAF, DevExpress, C#, ASP.NET, XPO, Microsoft SQL Server

Keywords

Information system, IS, web application, eXpress App Framework, XAF, DevExpress, C#, ASP.NET, XPO, Microsoft SQL Server

Citace

SYSEL, Josef. *Informační systém pro správu projektů*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém pro správu projektů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Josef Sysel
15. května 2018

Poděkování

Chtěl bych poděkovat svému vedoucímu práce, Ing. Vladimíru Bartíkovi, Ph.D., za jeho odborné vedení během celého průběhu vývoje této bakalářské práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Informační systémy jsou nedílnou součástí každodenního života většiny lidí používajících internet a dokáží za nás vyřešit většinu administrativních záležitostí, které se ještě v nedávné minulosti musely archivovat v papírové podobě. Tyto systémy nacházejí uplatnění v téměř všech odvětvích podnikání, jelikož jejich zavedení obvykle přináší mnoho benefitů v podobě větší efektivity práce.

Informační systémy mohou být uživatelsky dostupné zpravidla buď instalací na vlastním elektronickém zařízení a nebo přístupem přes webový prohlížeč díky využití architektury klient/server, která se v posledních letech stává stále více populární. Důvodem jsou nízké nároky na vybavení klienta, kterému stačí vlastnit pouze počítač, tablet nebo jiné elektronické zařízení zpřístupňující internetové webové stránky, protože veškeré hardwarové nároky se nachází na straně serveru a tím pádem jsou nezávislé na operačním systému i hardwarovém vybavení klienta. Naopak nevýhodou může pro některé uživatele být absence dostupnosti systému v offline režimu a často značně velké množství přenášených dat mezi zařízeními, které znamenají i vyšší bezpečnostní riziko.

Využití architektury klient/server je případ i této bakalářská práce, která je zaměřena na vývoj webového informačního systému sloužícího ke správě projektů se zaměřením na pomoc administrace projektů osobám samostatně výdělečně činným, dále jen „OSVČ“, nebo menším podnikatelům. Tato práce je rozdělena do dvou hlavních částí, přičemž se tyto části navzájem prolínají. První částí je teoretický rozbor problematiky tvorby webových informačních systémů. Druhá část se zabývá popisem praktického návrhu řešení a implementace systému včetně jejího testování. Celkově je text členěn do 8 kapitol a dvou příloh.

V následující 2. kapitole se nachází stručné shrnutí studií obdobných webových aplikací kladoucí si podobný cíl, z nichž jsem uvedl čtyři, které si zasloužili větší pozornost. Na základě jejich nedostatků, zadání této práce a své vlastní úvahy jsem ve 3. kapitole shrnul požadavky kladené na systém ústící v diagram případu užití, neboli use case diagram, a podle něj navrhl modelový příklad užití. Před započetím vývoje už zbývalo jen vybrat vhodné technologie. Při výběru jsem se soustředil primárně na technologie, se kterými mám minimálně dobré uživatelské zkušenosti. Současně jsem se snažil zvolit co nejvhodnější framework, abych nemusel začínat tzv. „na zelené louce“, ale místo toho se mohl soustředit více na cílové požadavky kladené na systém. Rozbor výběru technologií, mezi nimiž jsem se rozhodoval před zahájením programování, je popsán v kapitole 4.

Další části již přestávají být čistě teoretické, ale spíše popisují praktické řešení problematiky. Celý rozbor vlastního návrhu řešení se nachází v kapitole 5, ze které se odvíjí implementace webové aplikace. Návrh řešení zahrnuje entity–relationship diagram, dále jen „ER diagram“, který je výsledkem konceptuálního modelování vycházejícího z předchozí analýzy dat. Na základě ER diagramu byl pak ucelen logický návrh, na jehož výstupu stojí logické schéma databáze. V 6. kapitole jsou rozebrány, z mého pohledu, nejtěžší či nejzajímavější části programové implementace, která byla časově nejnáročnější částí celého projektu. V průběhu implementace byla aplikace náležitě testována pomocí standardního postupu, který počítá s vlastním průběžným testováním vývojáře v kombinaci s testováním softwaru od nezávislých potenciálních uživatelů. Postup a závěry testování aplikace jsou zdokumentovány v předposlední kapitole 7. V závěrečné 8. kapitole jsou shrnuty dosažené výsledky této několikaměsíční práce a možnosti optimalizace či rozšíření aplikace v budoucnosti.

Kapitola 2

Informační systémy pro správu projektů

Ještě před zahájením realizace bylo žádoucí udělat analýzu již existujících webově založených informačních systémů, poskytujících do maximální míry řešení zadání této bakalářské práce, abych se nepustil do vývoje softwaru, který již existuje, a tudíž by byl zbytečný. V současné době existuje velká řada aplikací zabývajících se správou projektů, úkolů, organizací času, organizací práce apod., a proto bylo poměrně zdlouhavé zanalyzovat dostatečné množství softwaru dostupného na internetu. Většinou jsem se odkazoval na webové adresy vhodných aplikací z technických odborných článků zabývajících se recenzemi nebo porovnáváním aplikací cílených ke stejným účelům. Analýza zahrnuje pouze systémy nebo části systémů, které jsou zdarma, jelikož můj informační systém neplánuji provozovat jako komerční záležitost a bude tudíž dostupná zdarma. Vybrané webové aplikace jsem nejprve studoval z odborných článků, abych je následně sám otestoval jako uživatel a na závěr jsem nahlédl i do uživatelských recenzí. Analýza všech systémů včetně výčtu cen jejich placených variant proběhla během měsíce dubna roku 2018.

2.1 Trello

Jedná se o webově založenou aplikaci pro správu projektů, vyvinutou americkou firmou Fog Creek Software[1] v roce 2011. Trello je velmi jednoduchá na použití, což jí pomohlo k jejímu rozmachu po celém světě a k poměrně kladným hodnocením pohybujícím se okolo 90%. Systém údajně využívají kromě běžných lidí i světoznámé firmy jako Google, Adobe nebo RedHat.

Základem této aplikace je systém posuvných karet, obsahující všechny možné informace o dané činnosti. Aplikace je založena na systému nástěnek, sloupců a karet. Menu orientované vlevo nahoře obsahuje seznam s nástěnkami rozdělenými do kategorií pro větší přehlednost. Jednotlivé nástěnky jsou složeny ze sloupců mající určitý název a řádky, jenž jsou ve skutečnosti pouze názvy jednotlivých karet, které po otevření zobrazí pop-up okno s popisem, záznamem aktivit, přílohami, štítkem, členy (označení uživatelé), apod. V základní verzi je i možnost vytvoření týmu s přiřazením dalších účastníků a nástěnek.

Myslím si, že tato webová aplikace se stává použitelnou pro komerční účely až po dokoupení levnější varianty rozšíření v hodnotě 10$ měsíčně. Dražší varianta by stála ještě dva krát více. Trellu chybí chat a dokonce i grafický kalendář, který je obecně považován za zá-

[1] Více o společnosti Fog Creek Software: http://www.fogcreek.com/

kladní prvek při orientaci mezi termíny, a který je k dispozici až v placených variantách. Mimo jiné velikost souborů je limitována 10MB/soubor.

Tato aplikace je sama o sobě velmi kvalitní nástroj, ale jelikož není specializována výhradně pro malé podnikatele nebo OSVČ, tak její množina funkcí, které má implementované, nepokrývá všechny požadavky vyžadované k uspokojení této práce.[12]

2.2 Freelo

Freelo je česká služba, která organizuje úkoly v týmu, dává jim řád, upozorňuje na plnění/neplnění termínů, rozpočet projektů a umí i nahradit klasickou e-mailovou komunikaci. V určitých směrech se jedná v podstatě o zdokonalení Trella a samotní vývojáři se tomuto tvrzení nebrání, ba naopak popisují na svém webu výhody oproti Trellu, jako například: podpora češtiny a online chat pro rychlou možnost rady od uživatelské podpory.

Základní nezpoplatněná verze je opět ořezána o mnoho funkcí, bez kterých ji považuji za nedostačující pro naplnění mých požadavků. Například: k dispozici jsou pouze dva přizvaní uživatelé, dva projekty a chatovat lze pouze pomocí komentářů pod jednotlivými úkoly. Další omezení může být v české doméně, kde je sice možnost přepnout jazyk do slovenštiny a angličtiny, avšak pod jednotnou doménou .cz se obecně nepředpokládá celosvětové rozšíření. Zakoupení nejlevnější placené varianty začíná na částce 590 CZK za měsíc.[7]

2.3 Avaza

Webová aplikace s příjemným uživatelským prostředím, určená pro spolupráci na projektech/úkolech s týmovými kolegy a klienty. Avaza je cloudově založený nástroj umožňující organizaci času, spolupráci s týmovými kolegy a zákazníky a rovněž podporuje výdej faktur zákazníkům. Správa projektů je rozčleněna mezi úkoly s různými instrukcemi přiřazeným členům, včetně termínů odevzdání. Také umožňuje konvertovat e-maily na úkoly. Aplikace umožňuje dokonce i platby platebními kartami, jež bych sám ocenil ve svém systému.

Základní verze zdarma omezuje počet zákazníku na 10, 1 administrátora, aktivní projekty a faktury na 5 za měsíc. Komunikace mezi jednotlivými aktéry systému je opět řešena pouze formou komentářů pod úkoly. Nicméně tento software se blíží mé představě o výsledném systému z vyzkoušených aplikací nejvíce. Pro komerční využití si její použití umím představit až po zakoupení alespoň nejlevnější varianty za 10$.[1]

2.4 Yalla

Posledním nástrojem stojícím podle mého úsudku za zmínku je propracovaná online aplikace pro správu úkolů, zaměřující se spíše na mladou generaci, která se ještě více snaží o jednoduchost použití nežli konkurence. Disponuje vestavěnou komunikační platformou (podobnou Slacku), kalendářem, Ganttovým grafem, prioritizací úkolů, šikovným průvodcem pro kontrolu odváděné práce a vedením rozpočtu projektu. Celé grafické uživatelské rozhraní je prakticky neomezeně upravitelné, s možností výběru různých témat a pozadí.

Limitovaná bezplatná verze nabízí prostor pouze pro 3 uživatele, rovněž je omezen počet projektů a zákazníků. Prémium účet stojí 9$ měsíčně. Celkově se jedná o velmi dobře graficky zpracovaný produkt a jeho Prémium verzi bych hodnotil jako použitelnou k naplnění mých požadavků.[15]

Kapitola 3

Analýza požadavků

Prvním krokem při tvorbě informačního systému je analýza požadavků vycházející ze zadání této bakalářské práce a současně z mojí vlastní úvahy, která požadavky rozebírá podrobněji a současně je tím rozšiřuje. Moje úvaha bere v potaz cílovou skupinu uživatelů a jejich potřeby ve smyslu užívání této aplikace plus očekávané požadavky na systém od této cílové skupiny uživatelů a jejich klientů.

3.1 Cílová skupina

Důležitou otázkou návrhu je cílová skupina, která bude systém používat, a pro kterou bude systém nejúčelnější. Jisté je, že cílový uživatel by měl mít denně k dispozici internet a nějaké elektronické zařízení, na němž se bude moci přihlásit do svého systému, jinak by pro něj postrádal smysl. Zařízení musí být vybaveno displayem alespoň o velikosti standardního smartphonu, aby se mu responzivní vzhled systému dokázal přizpůsobit.

Předpokladem je, že systém budou využívat především OSVČ nebo drobnější podnikatelé, čímž nevylučuji nasazení systému do firemního prostředí. Takové cílové skupiny potřebují ke své činnosti organizovat práci, přijímat požadavky od svých zákazníků a často i spolupracovat na zakázkách s dalšími lidmi. Pro člověka, který ke své činnosti nevyžaduje žádné další řešitele, je systém také přínosný a použitelný, avšak správa systému se pro něj může jevit zbytečně komplikovaná, jelikož je cílena na více uživatelů v pracovních rolích.

Dále je předpokládáno, že každý, kdo bude chtít systém provozovat, přenese zdrojové soubory včetně SQL skriptu k vytvoření výchozí databáze, kde se ukrývá nastavení uživatelských oprávnění, na svůj webový server a bude v systému vystupovat v roli administrátora 3.3.1. Cílový server musí splňovat nezbytné základní parametry a konfigurační soubor `Web.Config` je vyžadováno přizpůsobit konfiguračnímu nastavení daného serveru, na který se chystá deployment. Postup při nasazení aplikace na server je popsán v příloze A a současně může posloužit jako uživatelská příručka k použití hotové aplikace. Ostatním uživatelům stačí pouze zadat příslušnou webovou adresu do svého prohlížeče a do systému se přihlásit pod přiděleným uživatelským loginem a dočasným heslem pro první přihlášení dodaným do poštovní schránky. Systém je napsaný v angličtině, tedy v jazyce, jenž by měl aplikaci udělat přístupnou pro největší možnou komunitu lidí na světě používající internet.

3.2 Požadavky na systém

Je nezbytné zahrnout v systému tři základní uživatelské role: **zákazníci**, **řešitelé** a **administrátoři**. Editaci či dokonce nové role je reálné přidat dodatečně v případě potřeby v rámci UI pod administrátorskou rolí. Uživatel v roli zákazníka má možnost zadávat svoje požadavky v podobě vytvoření nového projektu, který je v ten moment viditelný pouze pro administrátory systému, kteří pak zanalyzují zákazníkův požadavek, změní stav projektu a přidělí k němu hlavního řešitele a vedlejší řešitele. Mohou tak určit i sami sebe či jiného administrátora, v případě, že jich systém obsahuje více.

Odpovědnost k projektu přebírá hlavní řešitel, který k projektu vytváří úkoly přiřazené jednomu čí více řešitelům. Tímto rezervuje svým kolegům práci a všechny běžící projekty a úkoly je možné zobrazit v kalendáři. Při vykonávání určitého úkolu má řešitel možnost spustit odpočet a při ukončení činnosti ho zase ukončit, což bude znamenat jeho uložení do systému v podobě odpracovaného času. Na tento čas se váže i možnost předpokládané finanční odměny, která kromě času závisí ještě na hodinové mzdě definované administrátorem.

Každý uživatel musí mít k dispozici možnost nahrát do systému nový soubor a označit ho jako veřejný a nebo soukromý. Současně musí být zachována určitá ochrana osobních údajů, a proto by nahrané soubory neměly být viditelné uživatelům, kteří nemají s daným souborem žádnou spojitost. Systém musí poskytovat možnost online komunikace, kde bude komunikace otevřená mezi všemi účastníky kromě zákazníků mezi sebou, jelikož se nepředpokládá, že by k ní měli mít nějaký důvod. Obsahy konverzací musí být viditelné pouze účastníkům daného konverzačního vlákna, které obsahuje vždy dva účastníky.

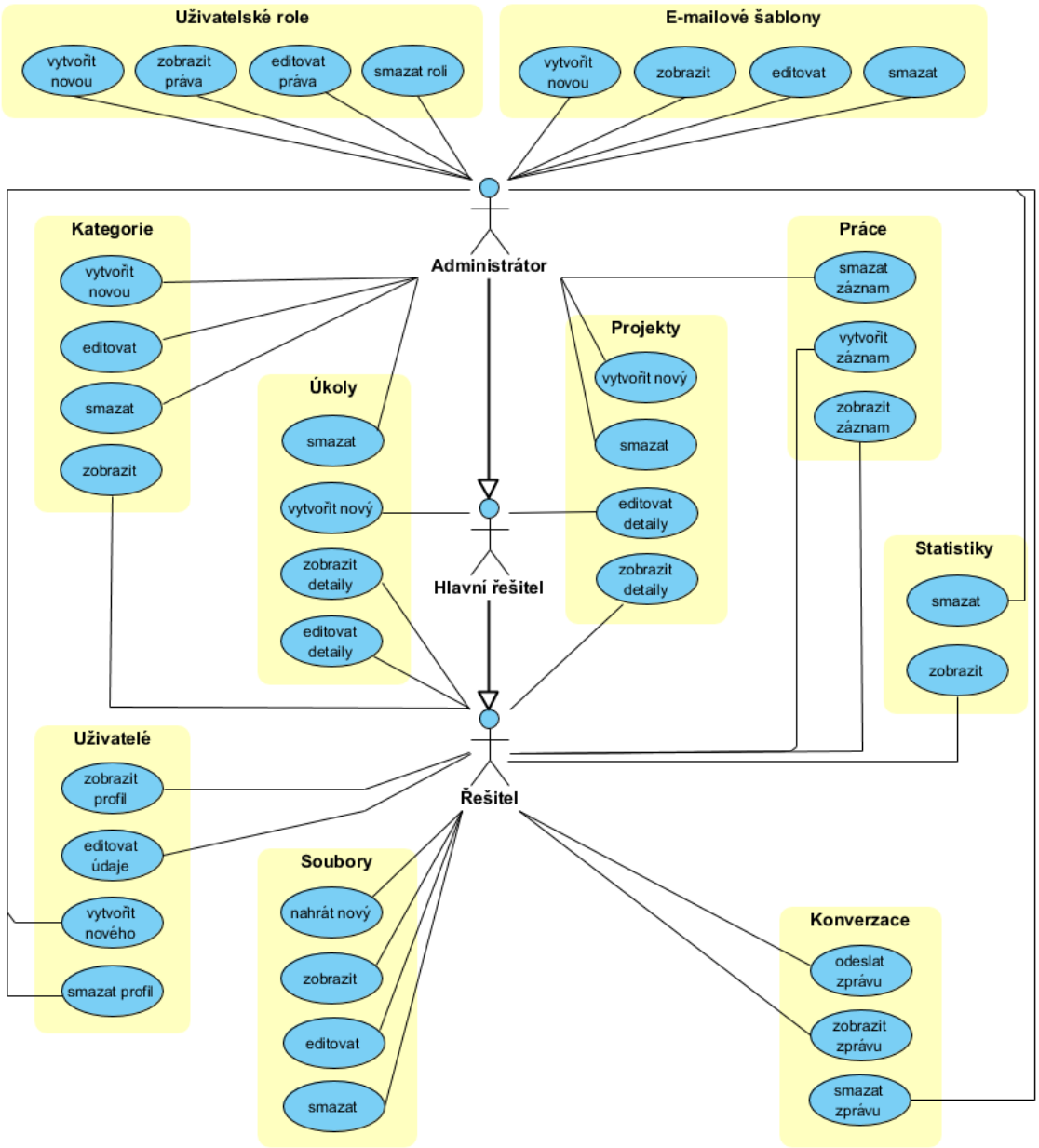
3.3 Use case diagram

Use case diagram byl vytvořen na základě modelovacího jazyka UML 2.0, což je jazyk nejčastěji používaný k tvorbě náčrtku diagramu. V tomto případě se jedná o náčrtek diagramu případu užití v rámci dopředného i zpětného inženýrství. U dopředného inženýrství je kód vytvářen na základě UML diagramu, kdežto u zpětného inženýrství je tento diagram tvořen až na základě již existujícího kódu.[6] Důvodem dvojnásobného načrtnutí diagramu bylo to, že jsem si během programování uvědomil, že některé systémy a případy užití je vhodné doplnit a jiné zase zrušit.

Výchozí nastavení definuje 3 základní uživatelské role, ze kterých vyplývají 4 aktéři. Prakticky neomezená práva má aktér Administrátor vystupující v programu pod rolí `Administrators`. Role tohoto aktéra je podrobněji rozepsána v podkapitole 3.3.1 a graficky znázorněna na níže uvedeném use case diagramu 3.1, kde je společně s aktéry Hlavní řešitel a Řešitel. Tito řešitelé spadají v aplikaci pod roli `Workers`, rozepsané v podkapitole 3.3.2. Aktéra Zákazník ukazuje use case diagram na obrázku 3.2. Zákazník vystupuje v programu pod rolí `Customers` 3.3.3. V případě potřeby je možné upravit či smazat stanovená výchozí práva nebo vytvořit zcela nová. Vše je možné editovat za běhu programu v administrátorské sekci Admin – Features v položce levého menu Rules.

3.3.1 Administrátor

Jedná se o roli vlastníka či správce systému, tedy pravděpodobně předpokládaného OSVČ, podnikatele nebo jimi pověřené osoby. Administrátoři mají povolení k přístupu ke všem položkám menu a mohou číst i editovat většinu záznamů. Výjimku tvoří editace práv jiného administrátora, pokud tedy v systému existuje alespoň jeden další administrátor. Další omezení musí být pochopitelně u historie online konverzace, kde nikdo nemá povoleno editovat záznamy o proběhnutých konverzacích, a záznamy, které aktuálnímu uživateli nenáleží, nemůže žádný uživatel ani číst. Rovněž nemá práva na editaci statistik, aby nebyly znehodnoceny jejich výsledky. Současně pro administrátory platí, že dědí všechny případy užití hlavního řešitele.



Obrázek 3.1: Use case diagram – Administrátor, Hlavní řešitel a Řešitel

3.3.2 Hlavní řešitel a Řešitel

Tato podkapitola podrobněji popisuje aktéry Hlavního řešitele a Řešitele, pro něž se předpokládá přiřazení ke všem řešitelům projektů, přidávaných do systému intervencemi administrátorů. Platí, že Hlavní řešitel dědí všechny případy užití Řešitele plus je rozšiřuje o další.

Projekty (Project)

- Zobrazit seznam i detailní náhled všech projektů, ve kterých je zahrnut mezi pracovníky (řešitele) pomocí vazební tabulky ProjectWorker na obrázku 5.6. Ke všem atributům projektu má právo pouze ke čtení.
- Pokud je jmenován administrátorem do pozice **Hlavního řešitele**, který je v aplikaci viditelný jako `Main Worker`, tak má tento řešitel posílená privilegia následujícím způsobem:
  - V levém sloupci detailního náhledu projektu může editovat atributy s předmětem projektu (`Subject`), stavem (`State`), předpokládanou cenou (`Price Prediction`), měnou (`Currency`), statusem (`Status`), časem upomínky na blížící se termín (`Remind In`), zaškrtnout pole k označení celého dne `All Day` a upravit popis projektu v poli `Description`.
  - V pravém sloupci detailního náhledu projektu smí editovat časový interval daného projektu pomocí atributů `Start On` a `End On`. Dále smí upravovat atributy s označením typu projektu `Label` a `Location` k eventuálnímu upřesnění lokace.
  - Použít tlačítko v horizontálním menu sloužící jako okamžité oddálení termínu odevzdání projektu o jeden týden.

Úkoly (Task)

- Zobrazit seznam úkolů a současně zobrazit i detailní náhled úkolů, ve kterých patří mezi jeho řešitele.
- V detailu konkrétního úkolu, v němž patří mezi jeho řešitele, smí editovat atribut `Completed` udávající informaci, zdali je již úkol hotov.
- Pokud je jmenován administrátorem do pozice **Hlavního řešitele** (`Main Worker`) v projektu, pak má tento řešitel posílená privilegia i v položce s úkoly následujícími rozšířeními:
  - Vytvořit nový úkol, aby k němu současně přiřadil řešitele, kteří mohou být vybíráni pouze z podmnožiny řešitelů projektu, do kterého je úkol zařazen a ve kterém je tento pracovník registrovaný jako hlavní řešitel.
  - Editovat všechny atributy úkolu, jehož je on sám autorem.
  - Použít tlačítko v horizontálním menu k oddálení termínu odevzdání úkolu přesně o jeden den.
  - Rovněž smí použít tlačítko v horizontálním menu k označení úkolu jako splněného.

Kategorie (Category)

- Zobrazit seznam všech kategorií.
- Zobrazit detailní náhled kterékoliv kategorie.

Soubory (File)

- Nahrát nový soubor nebo editovat již nahraný, což je ve skutečnosti operace smazání starého a nahrání nového souboru současně.
- Vybrat typ souboru v poli `Document Type`.
- Označit soubor jako soukromý či veřejný pomocí označení pole `Visibility`.
- Přidat popis souboru do pole `Description`.
- Číst detaily zahrnující atributy: datum vytvoření (`Created`), datum poslední modifikace (`Modified`), autor (`Created By`) a jméno a příjmení posledního uživatele, který soubor editoval (`Modified By`).
- Číst soubory, jichž je vlastníkem, a eventuálně je odstranit.
- Číst všechny soubory nahrané do projektu, ve kterém je daný zákazník zahrnut a současně je daný soubor označen jako viditelný pomocí pole `Visibility`.

Práce (Work)

- Zobrazit seznam všech prací včetně jejich detailů, v nichž je aktuální pracovník autorem. Autor záznamu o práci je automaticky systémem uložen do atributu `Worker`, který z uživatelského pohledu nelze editovat.
- Editovat v detailním náhledu práce pole pro komentář ke konkrétní práci (`Comment`).
- Číst v detailu práce atributy o autorovi záznamu (`Worker`), odpracovaném čase (`Work Time`), finanční odměně za odpracovaný čas (`Payment`),
- Vytvořit novou práci a přiřadit ji k úkolu, který musí obsahovat daného řešitele.
- Uložit záznam o zahájení nové práce, čímž je současně spuštěn časovač.
- Ukončit práci, které je autorem, použitím tlačítka `End` v seznamu záznamů prací.
- Pokud je jmenován administrátorem do pozice `Main Worker`, což v use case diagramu znamená Hlavní řešitel některého z projektů, potom má tento řešitel posílená privilegia i v položce Work a to následovně:
  - Zobrazit seznam prací včetně jejich detailních náhledů ostatních řešitelů, kteří jsou součástí projektu pod správou daného hlavního řešitele.

Uživatelé (ProjectUser)

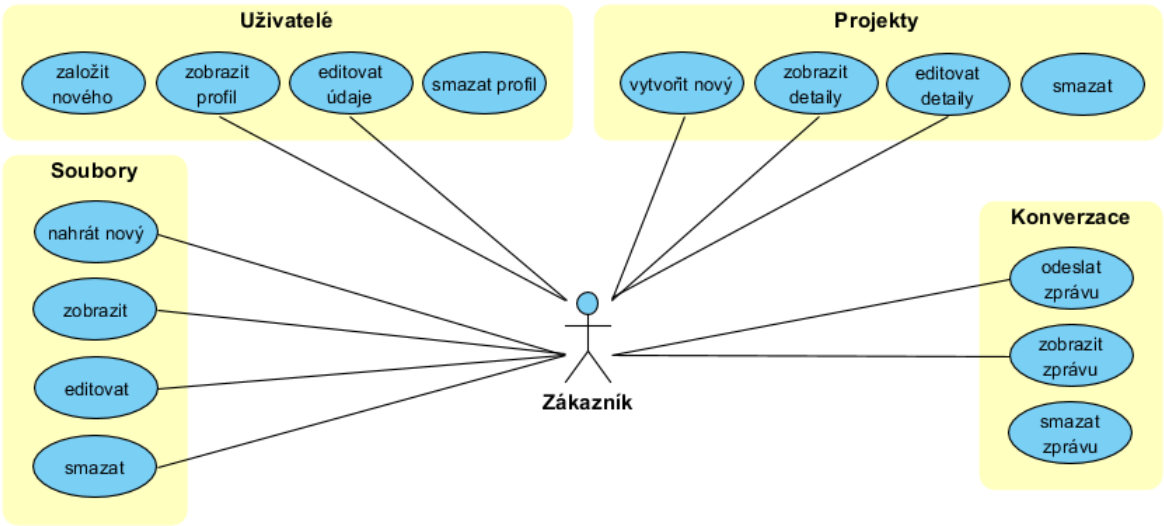
- Číst záznamy o všech administrátorech i o všech řešitelích.
- Číst a editovat svůj vlastní záznam obsahující jeho osobní údaje. Výjimku tvoří jen atribut `User Name` vyjadřující unikátní login uživatele, který přiděluje administrátor a je pouze pro čtení.
- Resetovat uživatelské heslo pomocí tlačítka `Reset Password`. Bezprostředně je po takové akci odesláno heslo příslušnému uživateli na jeho e-mailovou adresu uloženou v atributu `Email`.
- Nahrát svoji osobní fotografii do pole `Image` s využitím tlačítka `Browse`.

Konverzace (Chat History)

- Povoleno číst záznamy o proběhnutých konverzacích, ve kterých je obsaženo `OID` aktuálního uživateli, což je omezení na záznamy, ve kterých je aktuální řešitel odesílatelem (`Initiator`) a nebo příjemcem (`Invited user`).

3.3.3 Zákazník

Tato podkapitola odkrývá druhou část use case diagramu, která doplňuje první část o případy užití aktéra Zákazník. Pro připomenutí, první část diagramu je vyobrazena na obrázku 3.1 a zabývá se právy Administrátora, Hlavního řešitele a Řešitele. Use case diagram zákazníka se nachází na obrázku 3.2. Pod obrázkem diagramu jsou pak podrobněji popsány možnosti tohoto aktéra, který jinak v aplikaci vystupuje v roli `Customers`, pro níž se předpokládá přiřazení všem zákazníkům přidávaných do systému administrátorem.



Obrázek 3.2: Use case diagram – Zákazník

Projekty (Project)

- Založit nový projekt a nastavit zde oficiální termín zahájení (`Start On`) a termín ukončení (`End On`) práce na projektu.
- Přidat do pole `Description` popis projektu, jakožto jeho zadání k realizaci.
- Stanovit cenu do pole `Price Prediction`, za kterou předpokládá zhotovení svého požadavku, včetně výběru své preferované měny (`Currency`).
- Dokud administrátor nezmění stav (`State`) projektu ze stavu *Not started* na stav jiný, zákazník má možnost editovat výše zmíněné atributy.

Soubory (File)

- Nahrát nový soubor nebo editovat již nahraný.
- Označit soubor jako soukromý/veřejný pomocí označení pole `Visibility`.
- Vybrat typ souboru v poli `Document Type`.
- Přidat popis souboru do pole `Description`.
- Číst seznam i detaily o souborů, které jsou pro daného zákazníka dostupné, což znamená viditelné soubory k projektu, v němž je zákazník zahrnut.
- Číst soubory, jichž je vlastníkem a eventuálně je i odstranit/editovat.

Uživatelé (Project User)

- Číst záznamy o všech administrátorech. Nahlížet do záznamů o řešitelích, kteří jsou přiřazeni do společného projektu se současným zákazníkem.
- Číst a editovat svůj vlastní záznam s výjimkou atributu `User name`, který nelze editovat a slouží jako jednoznačný login přidělený administrátorem.
- Resetovat uživatelské heslo pomocí tlačítka `Reset Password`. Nové heslo je následně vygenerováno a zasláno na e-mail.
- Nahrát svoji osobní fotografii s využitím tlačítka `Browse` do pole `Image`.

Konverzace (Chat History)

- Může chatovat se všemi aktéry, pokud s nimi má společný projekt. Výjimkou jsou ostatní zákazníci, jež nemůže v systému vidět. Povoleno číst záznamy o proběhnutých konverzacích, ve kterých je obsaženo `OID` aktuálního uživateli, což zahrnuje pouze záznamy, ve kterých je aktuální řešitel odesílatelem (`Initiator`) nebo příjemcem (`Invited user`).

3.4 Modelový příklad případu užití

1. Zákazník dostává do své poštovní schránky přihlašovací údaje do systému a rovnou ve zprávě kliká na odkaz směřující na webovou stránku provozovatele služby.

2. Zadává login a vygenerované heslo, které si při prvním přihlášení mění dvojitým potvrzením na své vlastní.

3. Zákazník vytváří nový projekt, kde uvede subjekt, časový interval, očekávanou cenu, popis a případně nahraje nějaké soubory. Eventuálně napíše administrátorovi do chatu zprávu, aby mu upřesnil své požadavky.

4. Administrátor zobrazí tento nově vzniklý projekt a v případě zájmu změní stav projektu na *In Progress*, v opačném případě vybere stav *Deffered*. Dále určí hlavního řešitele z řad uživatelů s rolí `Workers` či `Administrators` a stejně tak vybere řešitele, u kterých očekává jejich zapojení do projektu. Volitelně může nastavit lokaci, label a status. Eventuálně založí a nebo otevře již existující chatovací vlákno inicializované zákazníkem, který by se rád ještě na něčem domluvil. Například ještě potřebuje projednat cenu navrženou zákazníkem, aby ji ještě upravil na více vyhovující.

5. Pověřený hlavní řešitel založí několik nových úkolů, aby práci rozčlenil na více podproblémů, do kterých přiřadí jednoho nebo více dostupných řešitelů zahrnutých v projektu. Úkolům nechybí časový interval pro vypracování, popis, předmět, label, status, priorita a zaškrtávací pole typu `bool` vyjadřující, zdali již byl úkol dokončen.

6. Řešitel si zobrazí nově přidělený úkol, na který je upozorněn e-mailem a v momentě, kdy začíná pracovat, tak spustí odpočet práce, ke které píše komentář, aby bylo zřejmé, co zrovna dělá a proč to dělá. Po ukončení činnosti ukončí odpočet záznamu o práci, kterou může vidět hlavní řešitel projektu a administrátor. Všechny soubory týkající se výsledného řešení nahrává k projektu.

7. Po určité době řešitel již považuje svůj úkol za dokončený a proto mění stav úkolu v atributu `Is Completed` na `true`, na což je upozorněn hlavní řešitel e-mailem.

8. Proces přidělování úkolů a vykonávání práce se opakuje tak dlouho, než hlavní řešitel uzná projekt za hotový a po případné online konverzaci se zákazníkem mění stav projektu na *Completed*.

Kapitola 4

Porovnání technologií a frameworků

K programování informačních systémů či webových aplikací je v současné době standardem použití tzv. frameworku, což je nástroj řešící často se opakující problémy při vývoji softwaru, které tak akorát navyšují náročnost projektu na úkor zaměření se na požadavky konkrétního systému. Rozhodoval jsem se mezi frameworkem eXpress App Framework, dále jen „XAF“, Entity Frameworkem a Nette. Pokud vezmu v potaz pouze Nette a XAF, jedná se o poměrné odlišné technologie, XAF je vyvíjen v programovacím jazyce C#/Visual Basic a nad ASP.NET, zatímco Nette je implementováno ve skriptovacím jazyce PHP. Entity Framework je naopak XAF vcelku blízký. Rozdíly vychází z toho, že Entity Framework je implementován nad technologiemi rodiny ADO.NET zatímco XAF je vyvíjen nad ASP.NET.

4.1 Nette

Mezi nejrozšířenější frameworky v České republice patří Nette, které je současně i českým produktem s rozsáhlou uživatelskou komunitou. Jeho klíčovou vlastností je dobrý výkon a zabezpečení při vytváření webových aplikací. O zabezpečení se starají například technologie XSS, CSFR, session hijacking a další. Snahou je i eliminovat rizika a umožňovat znovupoužitelnost zdrojového kódu. Do tohoto frameworku jsou začleněny i moderní technologie jako SEO, AJAX/AJAJ, KISS, MVC, DRY nebo cool URL. Tento software je dostupný zdarma i v rámci komerčního využití.[10]

Osobně oceňuji přehlednost, věcnost a stručnost dokumentace, dostupnou i v českém jazyce. Nette je vhodné pro začátečníky i pro vývoj větších projektů. Další nespornou výhodou k vývoji webových aplikací je nástroj Tracy, kterému se v české komunitě říká „Laděnka“. Tento nástroj vytváří výpisy o konkrétních chybách přímo ve webovém prohlížeči a vývojář díky tomu nemusí hledat chyby v logu. Umí měřit čas a paměťové nároky. Součástí je také šablonovací systém Latte, překládající šablony na jednoduchý optimalizovaný PHP kód. Nette Framework je šířený jako svobodný software, aby jej mohl používat kdokoliv a to buď v licenci New BSD nebo GNU General Public License (GPL) ve verzi 2 nebo 3.[10]

Nette je postaveno na programovacím jazyce PHP, což je skriptovací jazyk speciálně navržený pro potřeby webových stránek. PHP se v porovnání s konkurencí pyšní vysokou výkonností, nízkými náklady a vynikající přenositelností. Má rozhraní pro mnoho druhů databázových systémů, z nichž je v současné době nejtypičtější spojení s MySQL, která je vhodná pro tvorbu dynamických aplikací pro internet. Jde o Open Source technologie.[14]

4.2 Entity Framework

Entity Framework (EF) posouvá způsob práce s daty databáze na vysokou úroveň abstrakce. Jedná se o technologii, která pracuje jako tzv. Object Relation Mapper (ORM), která dovoluje vývojářům pracovat s relačními daty jako s bussiness modely, což eliminuje potřebu práce s relačními daty. Framework je možné programovat v jazyce C# nebo ve Visual Basicu a současně je vyvinut nad ADO.NET frameworkem. Tento framework poskytuje obsáhlý, modelově založený systém dělající přístup k datové vrstvě velmi jednoduchý.

Entity Framework umožňuje více se soustředit na aplikační logiku aplikace, přičemž s pomocí grafického rozhraní `Entity Designer Database Generation Power Pack` dovoluje generovat databázové schéma tabulek a vývojáři se díky tomu nemusí na přístup k datům vůbec zaměřovat. Stačí jim pouze využívat LINQ dotazy ve svém zdrojovém kódu. Při vývoji se používá buď přístup Code First (Entity model je generován ze zdrojového kódu) a nebo Database First, kde se Entity model generuje z již existující databáze.[9]

ORM na druhou stranu i svoje nevýhody. Při práci s objekty totiž často dochází k dotazům do databáze, které nemusí být vždy efektivní, což může omezovat rychlost. Dotazy jsou generovány automaticky a orientace v nich se stává s narůstajícím rozsahem databázového schématu značně komplikovanou až dokonce nemožnou. Níže je sepsán seznam hlavních výhod a nevýhod této technologie, které jsem při její studii shledal významnými.

Výhody:

- Redukuje čas vývoje a jeho cenu (povinnost každého správného frameworku).
- Umožňuje vývojářům mapovat strukturu databáze s pomocí grafického rozhraní.
- Podporuje dotazovací jazyk LINQ.
- Hodně rozšířený s rozsáhlou uživatelskou komunitou.

Nevýhody:

- Pro nováčky má poměrně komplikovanou syntaxi i z důvodu vysoké míry abstrakce.
- Při změně schématu v databázi je nutné aktualizovat schéma v solution.
- Daní za pohodlnější a rychlejší vývoj je nízká rychlost, která může být někdy omezující.
- Lazy loading zajišťuje načítání relačních vazeb až v době, kdy je k nim přistoupeno.

4.3 eXpress App Framework

Jedním ze sady různých softwarových komponent od firmy Developer Express Inc., dále jen „DevExpress“, je framework XAF. Jedná se o technologii postavenou nad platformou ASP.NET. Komponenty jsou velmi dobře propracované a rozsáhlé, díky čemuž již vývojáři nemusí vyvíjet vlastní náročné komponenty a místo toho se mohou zaměřit na požadavky zákazníků. Framework je integrován do Visual Studia, takže k použití stačí pouze doinstalovat balík dostupný na webových stránkách společnosti DevExpress. K automatickému generování uživatelského rozhraní slouží tzv. Application Model, který je automaticky vytvářen na základě zdrojového kódu objektů tzv. business vrstvy.

Například standardní grid od Microsoftu, který má funkci jako prvek pro zobrazení tabulky, neumožňuje tak rozsáhlou práci s daty jako grid od společnosti DevExpress. Tento grid umožňuje například seskupování dat, různé úrovně třídění, umístění vlastních sloupců, zobrazení sloupců, skrytí sloupců a filtrování dat, na jejichž vyfiltrovaných výsledcích zvýrazňuje zobrazená data. Framework mimo jiné podporuje grafické prvky pro práci z grafy a detailně vyřešené možnosti tisku. Uživatelé si mohou definovat i svoje vlastní tiskové reporty.

eXpress Persistent Objects

Další zajímavou vlastností je podpora Object Relation Mapping Tool v podobě nástroje zvaného eXpress Persistent Objects (XPO) sloužící jako most mezi relační databází a objektově orientovanou softwarovou konstrukcí. Dokáže na základě definice tříd a jejich vlastností automaticky generovat databázovou strukturu ve formě relačních databázových tabulek. Programátor se díky tomu nemusí na přímo zabývat databázovou strukturou, jelikož ji XPO vytvoří sama ze zdrojových kódů business objektů.[2]

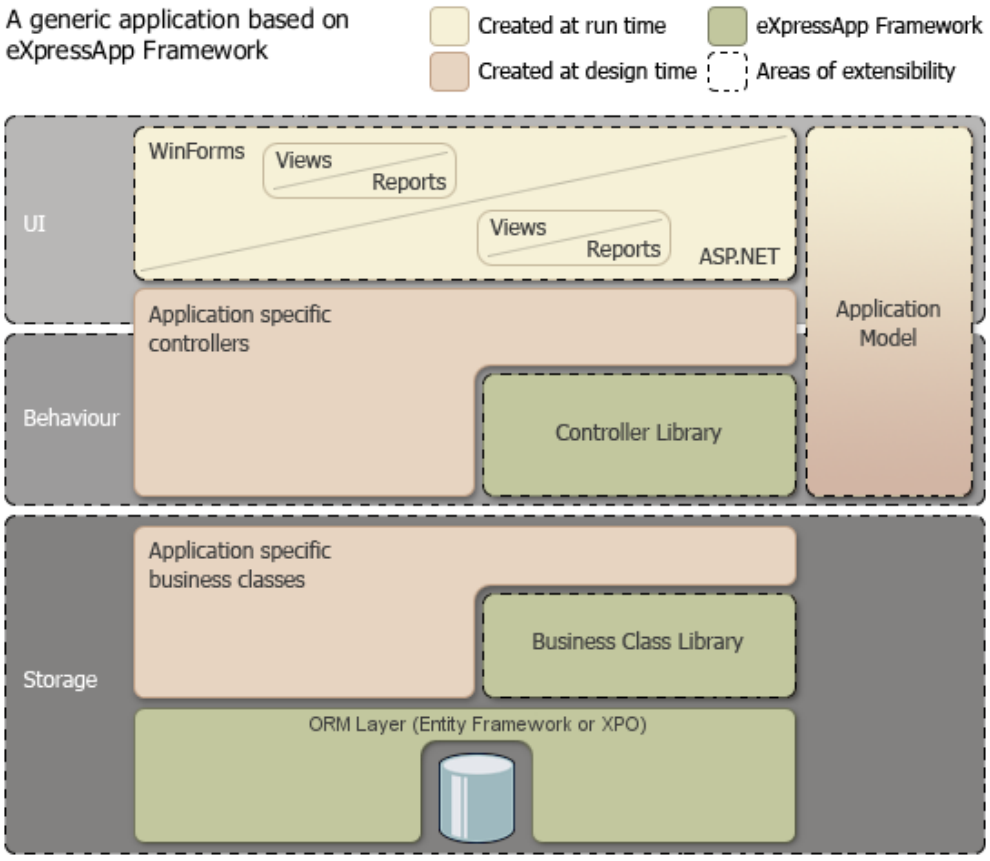
Výhody:

- Velmi propracovaný nástroj zahrnující vše potřebné, co by framework měl obsahovat. Jako dvě klíčové vlastnosti frameworku považuji:
  1. Urychlení vývoje a snížení nákladů na vývoj informačního systému.
  2. Zvýšení kvality výsledného produktu, případně umožnění jeho realizace, která se v některých případech může na počátku jevit jako neuskutečnitelná.
- Podpora dvou typů uživatelských rozhraní vystavěných nad společnou vrstvou logiky aplikace ASP.NET a Windows Forms.
- Podpora základních analytických funkcí a podpora tiskových sestav.
- Součástí je i podpora všech běžně známých dostupných relačních databází.
- Robustní bezpečnostní systém, který prošel testem bezpečnosti americké vládní organizace FIPS[1] a drží díky tomu normu RFC2898[2] týkající se kryptografie hesel.[3]
- Osvědčená modulární Model/View/Controller (MVC) architektura s podporou třístupňové architektury graficky znázorněné na obrázku 4.1
- Možnost vyvíjet na základě jednoho modulu aplikaci pro web i pro operační systém Windows.
- Přehledná dokumentace s velkým množstvím ukázkových příkladů a několik ukázkových aplikací, zahrnutých v instalaci frameworku.
- Možnost vyzkoušet si produkt zdarma po dobu jednoho měsíce.

[1] Více o organizaci FIPS: https://www.nist.gov/itl/fips-general-information
[2] Více o normě RFC2898: https://tools.ietf.org/html/rfc2898

Nevýhody:

- Vysoká pořizovací cena – oficiální cena je v současné době 2200$. Částka je vysoká z důvodu, že samotný framework XAF je možné zakoupit jen jako součást univerzálního balíku všech komponent firmy DevExpress. Software je tedy cílený na firmy vyvíjející dlouhodobě velké projekty, kdy se pak počáteční investice stává zanedbatelnou.
- Vysoké požadavky na výkon způsobené značnou mírou abstrakce komplexnosti frameworku.
- Vývoj je prakticky reálný pouze na operačním systému Windows.
- Žádné zdroje informací v českém jazyce.



Obrázek 4.1: 3-stupňový model architektury XAF.[5]

4.4 Volba frameworku a datového úložiště

Rozhodl jsem se pro použití frameworku XAF z důvodu, že se mi jeho přínos pro můj projekt zdá efektivnější nežli v případě Nette a EF (s ADO.NET). Ačkoliv XAF a EF mají mnoho společného, XAF u mne nakonec zvítězil, protože na mě celkově udělal lepší dojem, věřím v jeho bezpečnost a líbí se mi dokonalé provázání komponent, které spolu potřebují spolupracovat jako například XPO a různé pomocné moduly.

Největší usnadnění v použití frameworku jsem shledal v generování prvotního uživatelského rozhraní a generování databázových tabulek na základě zdrojového kódu tříd tvořících kostru aplikace. Současně bych se do budoucna rád orientoval více cestou technologií od Microsoftu. Hlavní nevýhoda byla z mého pohledu vysoká cena, kterou jsem však vyřešil implementací systému nad zkušební verzí, jíž je možné využít pro předposlední i poslední verzi této služby, z čehož vyplývá celková dvouměsíční doba na programování aplikace. Zakoupení softwaru od společnosti DevExpress tedy pro tento relativně malý projekt nebylo nutné, ale bude zváženo do budoucna v závislosti na poptávce po optimalizaci této aplikace a nebo ke komerčnímu využití.

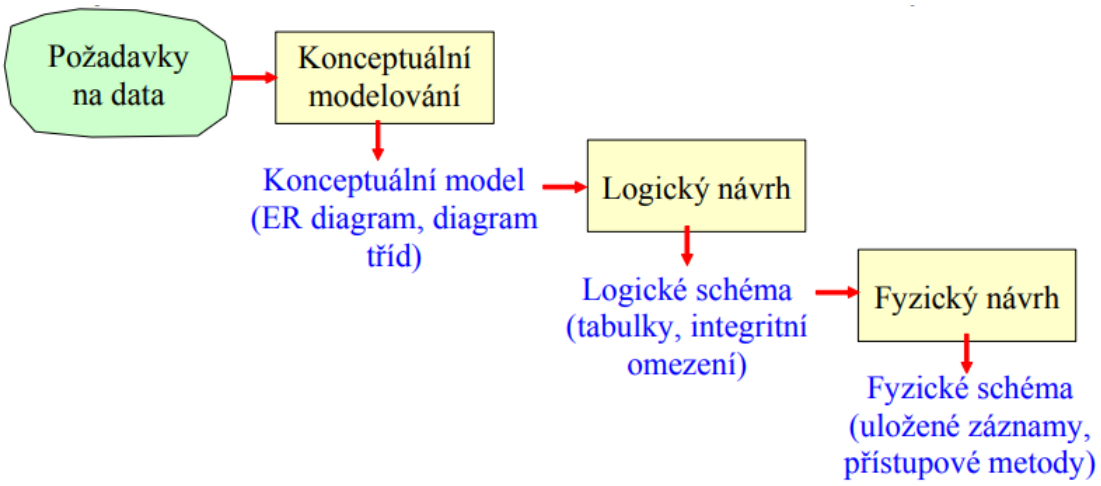
Jelikož informační systémy produkují ohromné množství dat, dalším rozhodnutím byl výběr datového úložiště, které nejčastěji bývá strukturované formou relační databáze. XAF podporuje většinu běžně používaných systémů řízení báze dat (SŘBD) jako například Oracle, MySql nebo MSAccess. Já jsem zvolil Microsoft SQL Server 2016, se kterým mám dobré zkušenosti a navíc bývá ve spojení s mými vybranými technologiemi standardem. Výpis kompatibilních SŘBD je k nalezení v příloze A.2.

Kapitola 5

Návrh řešení

Po vyjasnění požadavků přichází na řadu návrh řešení informačního systému, kde bude podrobně popsán postup návrhu informačního systému, a to jak z obecného teoretického hlediska, tak i z hlediska návrhu řešení tohoto konkrétního systému. Na základě analýzy požadavků z kapitoly 3 bude v této kapitole rozebrán konceptuální, logický a fyzický návrh. Z výsledku těchto návrhů bude možné zahájit implementaci.

„Mezi základní kroky, kterými procházíme při vývoji softwarového systému patří analýza požadavků, návrh, konstrukce (programování), testování a předání do užívání. Tyto kroky mohou po sobě bezprostředně následovat nebo mohou obsahovat cykly v závislosti na použitém modelu životního cyklu. Pokud se zaměříme pouze na kroky související s návrhem databáze, můžeme rozlišit tři základní kroky znázorněné na obrázku 5.1.“[16]



Obrázek 5.1: Základní kroky návrhu relační databáze.[16].

5.1 Konceptuální modelování

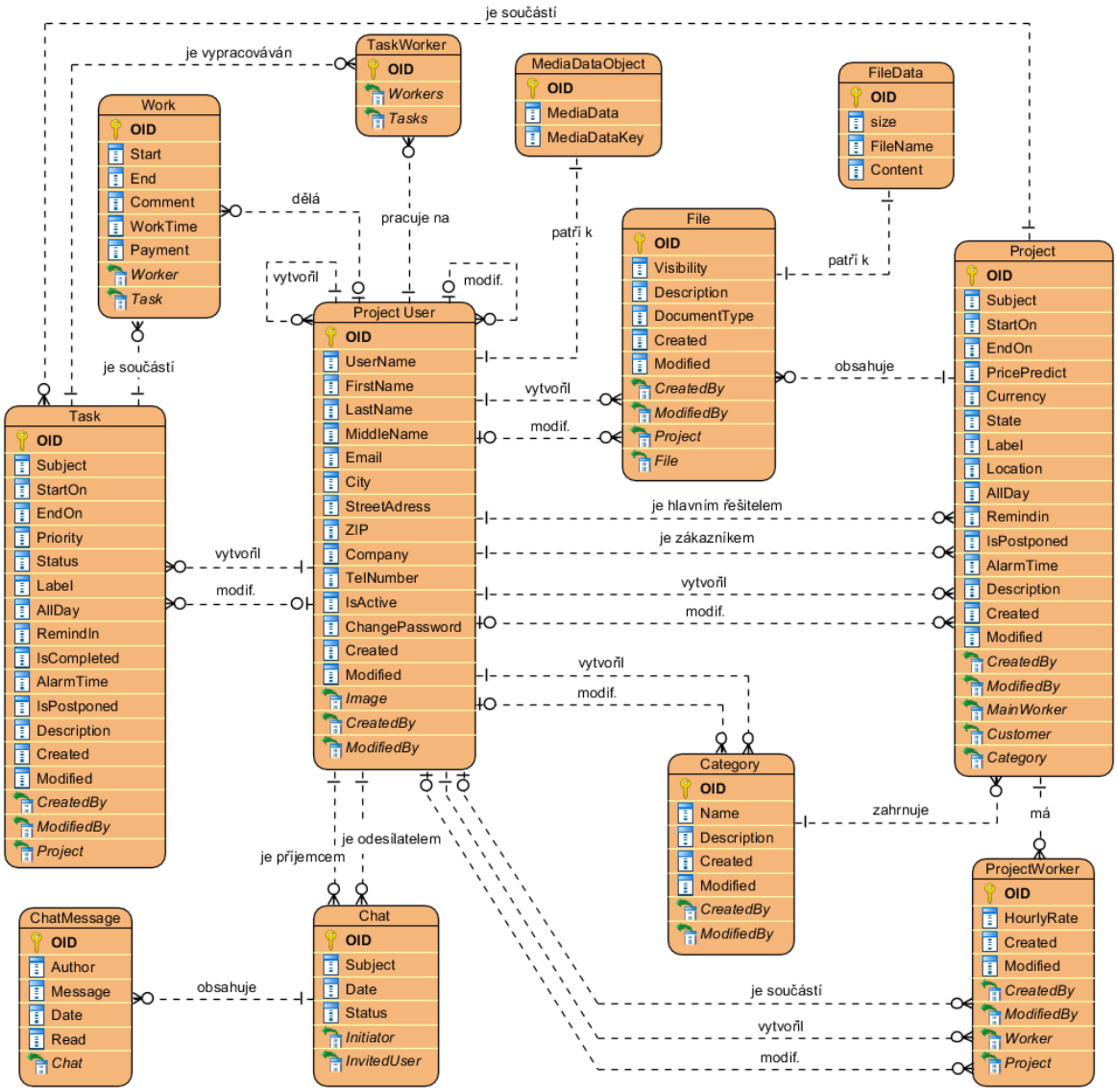
Konceptuální modelování je fáze datové, případně objektové analýzy využívající modelů založených na objektech aplikační domény. Jeho úlohou v procesu návrhu relační databáze je na základě analýzy požadavků na data od koncového uživatele a na základě pochopení domény, pro kterou je databáze navržena, vytvořit konceptuální model. Nejtypičtějším graficky znázorněným konceptuálním modelem je Entity–Relationship diagram, dále jen „ER diagram“, jenž je podrobněji popsán v následující podkapitole 5.1.1. Tento typ diagramu zároveň tvoří výstup konceptuálního návrhu.[16]

5.1.1 ER diagram

Jedná se o typickou techniku konceptuálního modelování vhodnou pro relační databáze, protože neodráží objektový pohled, ale strukturovaný přístup. Pro objektový přístup, spadající do kategorie funkčního modelování, by byl vhodný diagram tříd, který ale neobsahuje žádné operace.[16] V této práci je využíván pouze datový přístup, a proto zde bude dále rozebráno pouze datové modelování. Výsledný ER diagram je k dispozici na obrázku 5.2.

S ER diagramem jsou provázány i některé důležité pojmy:

- **Entita** – Jedná se o objekt z reálného světa, o kterém chceme uchovávat informace v databázi. Např.: projekt, uživatel či soubor.
- **Entitní množina** – Množina entit téhož typu, které sdílí tytéž vlastnosti.
- **Vztah** – Asociace mezi entitami.
- **Atribut** – Vlastnost entity a nebo vztahu mezi entitami, o které chceme ukládat informace. Např.: subjekt, příjmení, telefonní číslo, apod.
- **Primární a cizí klíč** – Primární klíč je atribut s unikátní hodnotou, zatímco cizí klíč je odkaz primárního klíče v jiné entitě používán pro spojování entit.
- **Kardinalita** – Udává, do kolika vztahů daného typu může jedna entita množiny na tomto konci vstupovat. Zpravidla nabývá poměru 0:N, 1:N, N:M nebo 1:1.
- **Členství** – Minimální počet vztahů daného typu, kterých se musí jedna entita účastnit. Členství bývá povinné (kardinalita 1:N) a nepovinné (kardinalita 0:N).



Obrázek 5.2: Entity–Relationship diagram.

5.2 Logický model

Druhým krokem návrhu relační databáze je logický model, jehož cílem je transformovat konceptuální model na schéma relační databáze. V této fázi tedy dochází k převodu ER diagramu na kolekci jednotlivých entit, která bude tvořit databázi. Musíme určit například: tabulky tvořící databázi, sloupce tvořící tabulky, datový typ hodnot jednotlivých sloupců, primární klíče, cizí klíče a jaká další omezení budou deklarována. Výstupem tohoto druhého kroku návrhu je logické schéma.[16]

5.2.1 Logické schéma

Tato podkapitola se zaměřuje na rozbor relací (tabulek) databáze. Tabulky, které jsou generovány frameworkem jsou popsány stručněji. S rozborem logického schématu souvisí i uživatelské role, jenž jsou detailně popsány v podkapitole 3.3.

Společný atribut – OID

Každá relace (tabulka) obsahuje **primární klíč**, jakožto atribut nesoucí název `OID` (Object Identifier), který je datového typu `GUID`, celým názvem Globally unique identifier. Tento atribut (sloupec) je automaticky generován frameworkem XAF, popsaným v podkapitole 4.3 pro každou tabulku. Jedná se o 128-bitové číslo vytvořené operačním systémem Windows nebo určitou Windows aplikací. Účelem je jednoznačně identifikovat specifickou komponentu, hardware, software, soubory, uživatelské účty, databázové entity a nebo jiné položky.[11]

Logovací atributy

Většina mnou vytvořených tabulek, respektive vygenerovaných komponentou XPO na základě mnou navržených tříd (business objektů), zahrnuje sloupce `Created`, `CreatedBy`, `Modified` a `ModifiedBy` Jejich účelem je logování základních informací o všech záznamech dané tabulky. `Created` označuje čas vytvoření daného záznamu a `Modified` vyjadřuje čas poslední modifikace záznamu. Oba časové logovací prvky jsou datového typu `TimeSpan`. `CreatedBy` je cizím klíčem tabulky ProjectUser 5.3 a vyjadřuje celé jméno[1] uživatele, který daný záznam vytvořil. `ModifiedBy` je rovněž cizím klíčem tabulky ProjectUser, aby uchovával informaci o uživateli, který naposledy modifikoval daný záznam.

OptimisticLockField a GCRecord

Všechny tabulky, s výjimkou ProjectUser, shodně obsahují následující 2 atributy typu `integer`: `OptimisticLockField` a `GCRecord`. `OptimisticLockField` řeší problém, kdy více uživatelů edituje jeden záznam současně. S každým dalším uložením záznamu je číslo tohoto sloupce inkrementováno o 1 a tato hodnota je porovnávána s aktuální hodnotou záznamu v databázi. Pokud se hodnota ukládaného záznamu nerovná hodnotě v databázi, znamená to, že záznam již není aktuální a záznam tím pádem není možné uložit.

`GCRecord` je atribut užívaný k označení objektů smazaných z databáze. Je to součást mechanismu nazývaného odložené vymazání.[2] Pokud má tento sloupec hodnotu NULL, znamená to, že daný záznam (řádek tabulky) není ve stavu odloženého vymazání.

[1] Celé jméno je ve formátu: jméno + třetí jméno + příjmení + (firma)

[2] https://documentation.devexpress.com/CoreLibraries/2103/DevExpress-ORM-Tool/Concepts/Deferred-and-Immediate-Object-Deletion

Project User

ProjectUser

OID	FirstName	LastName	MiddleName	Phone	Email	City	StreetAdress
ZIP	Company	Created	Modified	**Image**	**CreatedBy**	**ModifiedBy**	

Obrázek 5.3: Tabulka ProjectUser

Tabulka ProjectUser 5.3 obsahuje všechny potřebné informace o všech uživatelích systému. O každém uživateli je možné uložit jméno, příjmení, kontaktní údaje, adresu trvalého bydliště, PSČ, fotku a případně i název firmy a jeho třetí jméno. Sloupec `Image` je cizím klíčem z frameworkem vygenerované speciální tabulky MediaDataObject, která je k dispozici na následujícím obrázku 5.4.

MediaDataObject

OID	MediaData	MediaDataKey	OptimisticLockField	GSRecord

Obrázek 5.4: Tabulka MediaDataObject

MediaDataObject existuje za účelem uložení a zobrazení obrázku uživatele s využitím atributu `MediaData`, jakožto bytového pole objektu načítaného z databáze na vyžádání. Atribut `MediaDataKey` je klíčová hodnota typu `string`, která je automaticky aktualizována, pokud je bytové pole `MediaData` upraveno. Jedná se o unikátní hodnotu pro každý objekt v této tabulce.

Project

Project

OID	Subject	StartOn	EndOn	PricePredict	Currency	State	Label
AllDay	Location	Description	RemindIn	AlarmTime	Status	IsPostponed	Created
Modified	**CreatedBy**	**ModifiedBy**	**MainWorker**	**Category**	**Customer**	OptimisticLockField	GCRecord

Obrázek 5.5: Tabulka Project

V tabulce Project 5.5 jsou spravovány všechny informace o projektech. V prvním sloupci `Subject` se nachází údaj, datového typu `string`, o názvu konkrétního projektu. Dále je třeba ohraničit časový interval konkrétních projektů pomocí sloupců `StartOn` a `EndOn`. Hodnoty uložené v těchto sloupcích jsou datového typu `DateTime` a jsou regulovány podmínkou udávající pravidlo, že čas v `EndOn` musí být pozdějšího času nežli čas v `StartOn` a současně musí `StartOn` obsahovat pozdější a nebo stejný čas v porovnání s aktuálním časem s přesností na sekundy. `PricePredict` odhaluje předpokládanou cenu projektu zadanou hlavním řešitelem. S tím souvisí i `Currency`, datového typu `enum`, k určení konkrétní měny předpokládané částky. Na výběr jsou 4 měny: americký dolar, ruský rubl, euro a česká koruna.

Důležitým atributem je sloupec `State`, jenž je také datového typu `enum` a značí aktuální stav projektu, přičemž může nabývat následujících čtyř stavů:

- *Not Started*: Projekt je pro zatím pouze vytvořen zákazníkem a ještě mu nebyla věnována žádná pozornost ze strany administrátora.
- *In Progress*: Projekt byl administrátorem svěřen do rukou hlavního řešitele, byli mu přiděleni pracovníci a je možné na něm vykonávat práci asociovanou s úkoly daného projektu.
- *Deffered*: Projekt byl zamítnut a nebude řešen.
- *Completed*: Projekt byl označen za dokončený a již není možné ho nikterak editovat.

`Label` je další sloupec typu *enum* znázorňující jednoslovný popis daného projektu, který se následně promítne i v rozhraní kalendáře jako prvek dělající položky v kalendáři přehlednější. Sloupec `AllDay` je datového typu `bool` a slouží k jednoduché funkci – při hodnotě `true` vyznačí hranice časového intervalu projektu na celé dny. `Location` může být využita ke specifikaci geografické polohy vztahující se k projektu. `Description` obsahuje popis zadání či požadavku, který je očekáván od zákazníka.

Součástí projektu jsou i sloupce `RemindIn`, `AlarmTime` a `IsPostponed`, z nichž je v uživatelském rozhraní viditelný pouze ten první. Hlavní řešitel projektu si tím může nastavit, jak dlouho před termínem odevzdání si přeje být upozorněn na blížící se konec intervalu pro vypracování projektu. Funkce reminderu je podrobně popsána v implementační části práce v podkapitole 6.2.

Tabulka zahrnuje také nezbytné cizí klíče k ucelení vazeb s dalšími tabulkami. Atributy `Customer` a `MainWorker` jsou cizími klíči z předchozí tabulky ProjectUser a jsou nezbytné k identifikování informace o tom, kdo je zákazníkem a kdo hlavním řešitelem, kterého určuje administrátor. Podrobný popis uživatelských rolí je k nalezení v podkapitole 3.3. Cizí klíč `Category` z tabulky Kategorie nezbytný není, ale je doporučený k udržení přehlednosti uživatelských dat, protože přiřazuje danému projektu jednu konkrétní kategorii.

Project Worker

ChatMessage

OID	Message	Date	Read
Chat	**Author**	OptimisticLockField	GCRecord

Obrázek 5.6: Tabulka ProjectWorker

Tabulka ProjectWorker 5.6 se zobrazuje v detailu projektu a upravovat ji může pouze administrátor. Účelem této tabulky je vytvoření vazby M:N mezi tabulkami ProjectUser a Project. Kromě toho, obsahuje důležitý atribut `HourlyRate` k uložení hodinové mzdy pracovníka, který je zahrnut v daném projektu.

Task

Task

OID	Subject	StartOn	EndOn	Description	AllDay	Status
RemindIn	IsPostoponed	AlarmTime	Priority	**IsCompleted**	Created	Modified
Label	**CreatedBy**	**ModifiedBy**	**Project**	OptimisticLockField	GCRecord	

Obrázek 5.7: Tabulka Task

Tabulka Task 5.7 rozšiřuje předchozí tabulku Project o jednotlivé úkoly a tím umožňuje rozdělit složité zadání na jednotlivé části zadání a současně tak rozdělit práci mezi jednotlivé pracovníky. Pravomoc vytvářet úkoly a přidělovat je pracovníkům má vždy na starosti, a je k tomu oprávněn, pouze hlavní řešitel daného projektu.

Podobně jako Project i Task obsahuje sloupce `Subject`, `StartOn`, `EndOn`, `Description`, `AllDay`, `Label`, `AlarmTime`, `IsPostponed` a `RemindIn`. Tyto sloupce mají stejnou funkci jako v již zmiňované tabulce Project 5.5. Rozdíl je pouze v podmínce pro vytvoření intervalu kooperace hodnot sloupců `StartOn` a `EndOn`, kde platí odlišná podmínka: Hodnota atributu času `StartOn` v Task musí být větší nebo rovna aktuálnímu času a současně musí být větší nebo rovna časové hodnotě `StartOn` v tabulce Project a současně `EndOn` v tabulce Task musí být menší nebo rovno hodnotě `EndOn` v tabulce Project. Cizím klíčem tabulky je atribut `Project`, jakožto sloupec přiřazující konkrétní úkol ke konkrétnímu projektu.

File

File

OID	IsVisible	Description	DocumentType	Created	Modified
File	**CreatedBy**	**ModifiedBy**	**Project**	OptimisticsLockField	GCRecord

Obrázek 5.8: Tabulka File

Uživatelé často potřebují ukládat na webová úložiště různé soubory ke sdílení s ostatními či k ponechání pro soukromé účely a tato webová aplikace není výjimkou. Tabulka File 5.8 ve spolupráci s frameworkovou tabulkou FileData 5.9 zařizuje bezpečné nahrání, uložení, stažení či případné smazání souborů z databáze. File obsahuje stejnojmenný sloupec `File`, který je cizím klíčem z FileData, a tím propojuje tyto dvě tabulky. Jejich kardinalita je 1:1.

Každý soubor je fyzicky uložen ve FileData a rozšířen dalšími informacemi ve File. Uživatel si může nastavit viditelnost svého souboru použitím zaškrtávacího pole, jenž ukládá hodnotu ve formě datového typu `bool` do sloupce `IsVisible`. Pokud uživatel označí zaškrtávací pole, současně tím nastaví hodnotu na `true`, daný soubor je pak viditelný všem účastníkům daného projektu.

Sloupec `DocumentType` je typu `enum` a uživatel si může sám zvolit typ souboru. Pokud tak neučiní, systém určí typ souboru místo něj, ale pouze za předpokladu, že daný typ souboru v seznamu existuje. V případě absence daného typu souboru je typ určen jako `Other`. Atribut obsahuje seznam 10 nejpoužívanějších typů souborů. Nakonec každý soubor musí být přiřazen k jednomu projektu, k čemuž slouží cizí klíč `Project`.

FileData

OID	Size	FileName	Content	OptimisticLockField	GSRecord

Obrázek 5.9: Tabulka FileData

V prvním sloupci `Size` je obsažena fyzická velikost souboru. V dalším (`FileName`) pak jméno souboru, včetně přípony. Obsah souboru je ukryt ve sloupci `Content`.

Work

Work

OID	Start	End	Comment	WorkTime
Payment	**Worker**	**Task**	OptimisticLockField	GCRecord

Obrázek 5.10: Tabulka Work

Informace zahrnuté v tabulce Work 5.10 uchovávají záznamy o odvedené práci, kterou potřebují zaznamenávat pracovníci, aby bylo zjevné kolik svého času obětovali vykonáváním činnosti na úkolu, který jim byl přiřazen. Vazbu s tabulkou Task obstarává stejnojmenný cizí klíč `Task`, zatímco odkaz do tabulky s uživateli má na svědomí cizí klíč `Worker`. Doba strávená nad jednotlivými úkoly je ukládána do sloupce `WorkTime`, který je výsledkem jednoduchého výpočtu, kde se počítá rozdíl mezi hodnotou času ve sloupci `Start` a hodnotou času ve sloupci `End`. Na základě odpracovaného času a hodinové mzdy, kterou má každý pracovník stanovenou od hlavního řešitele v tabulce ProjectWorker, je počítána předpokládaná odměna za odpracovaný čas na konkrétním úkolu. Dále je možné zanechat k práci komentář a ten je zaznamenán v atributu `Comment`.

Category

Category

OID	Name	Description	Created	Modified
CreatedBy	**ModifiedBy**	OptimisticLockField	GCRecord	

Obrázek 5.11: Tabulka Category

Jednoduchou databázovou tabulkou je Category (na obrázku 5.11) sloužící pouze pro uložení názvu kategorie do sloupce `Name` a její popis do sloupce `Description`.

Chat

Chat

OID	Subject	Date	Status
Initiator	**InvitedUser**	OptimisticLockField	GCRecord

Obrázek 5.12: Tabulka Chat

Jednou z položek menu je také Chat History, ve které se po otevření rozkládají data z tabulky Chat 5.12 používající se pro zaznamenávání online konverzace mezi uživateli. Sloupec `Subject` označuje předmět zprávy, který udává odesílatel při odeslání první zprávy nového konverzačního vlákna. Ve sloupci `Date` je čas a datum odeslání a ve sloupci `Status` se nalézá stav vlákna, které může nabývat dvou stavů: Aktivní a nebo Ukončený. Poslední dva vyobrazené sloupce jsou cizí klíče z tabulky ProjectUser a jak již názvy napovídají, jedná se o označení uživatelů, kteří mezi sebou komunikovali. `Initiator` vystihuje iniciátora konverzačního vlákna a naopak `InvitedUser` ukládá adresáta konverzačního vlákna. Samotný obsah zpráv se nachází v detailu každého vlákna a je uložen v další tabulce popsané níže pod obrázkem 5.13.

ChatMessage

ChatMessage

OID	Message	Date	Read
Chat	**Author**	OptimisticLockField	GCRecord

Obrázek 5.13: Tabulka ChatMessage

Detail jednotlivých zpráv se skrývá v tabulce ChatMessage, která je spojena s tabulkou Chat pomocí cizího klíče tvořícího sloupec `Chat`. Druhým cizím klíčem je sloupec `Author` z tabulky ProjectUser, kde je uložen autor dané zprávy.

Nezbytný sloupec je zde `Message` s textem, který autor napsal a odeslal příjemci. Dále je třeba zaznamenat čas odeslání sloupcem `Date` a nakonec stav zprávy, zdali je přečtená příjemcem, či nikoliv. To se zapisuje do sloupce `Read`, jenž je datového typu `bool`.

Settings

Settings

OID	Url	SmtpHost	SmtpPort	SmtpUseSSL	SmtpUserName
SmtpPassword	SmtpFromMail	SmtpFromName	AdminAddress	OptimisticLockField	GCRecord

Obrázek 5.14: Tabulka Settings

K odeslání e-mailu je využívána metoda `Send()` třídy `EmailSender` používající ke své činnosti atributy z kolekce Settings, která je současně i tabulkou v databázi (viz 5.14). Při odesílání e-mailu je nezbytné určit URL/Port SMTP serveru, e-mailovou adresu odesílatele, jméno odesílatele, heslo e-mailové schránky odesílatele, uživatelské jméno a povolení použití šifrovacího protokolu SSL.

EmailTemplate

EmailTemplate

OID	Code	Text	Subject	Priority	OptimisticLockField	GCRecord

Obrázek 5.15: Tabulka EmailTemplate

Kromě nastavení služby k odeslání e-mailu je třeba vyplnit tělo elektronické pošty vhodným obsahem. K tomu fungují textové šablony tabulky EmailTemplate 5.15 s několika atributy. Subjekt obsahuje předmět zprávy, zatímco Text vyplňuje tělo zprávy. Mimoto zde používám 3-úrovňové nastavení priority a pomocný sloupec `Code`, který je potřebný k identifikaci konkrétní šablony. Podrobný Popis implementace k odesílání e-mailových šablon je dostupný v podkapitole 6.4.

Další tabulky generované frameworkem

Pro úplnost je třeba ještě zmínit zbývající seznam tabulek, které byly vegenerovány frameworkem XAF s pomocí ORM komponenty XPO již při inicializaci nového projektu ve Visual Studiu. Většinou se jedná o tabulky uchovávající informace o správě uživatelů a uživatelských rolí. Tyto tabulky jsou pro úplnost vyjmenovány v následujícím seznamu:

- `PermissionPolicyNavigationPermissionsObject`, `PermissionPolicyTypePermissionsObject`, `PermissionPolicyObjectPermissionsObject` a `PermissionPolicyMemeberPermissionsObject` – Obsahují nastavení uživatelských práv pro jednotlivé položky, objekty, atributy a typy omezení (např. Read only).
- `PermissionPolicyUser`, `PermissionPolicyUserUsers` a `PermissionPolicyRole` – Zahrnují registrované uživatele a jejich přidělené role.
- `ModelDifference` a `ModelDifferenceAspect` – Uchovávají informace o změnách UI.
- `XPObjectType` – Tato tabulka poskytuje informace o typu objektu pro komplexní hierarchii dědičnosti.

5.3 Normalizace

Je to proces, při kterém se relace rozkládají za účelem jednodušší práce s daty, jejich lepší manipulaci, zabránění redundance dat (dat, která se zbytečně opakují) a lepší konzistence dat, což do značné míry souvisí s redundancí.[8] V praxi se normalizace používala k návrhu databázového schématu spíše v minulosti, kdy ještě systémy nebyly tolik složité a tím pádem neobsahovali velké množství tabulek. V současné době se používá spíše jako doplněk při optimalizaci schématu relační databáze pomáhající odstranit nedostatky, které vznikly během transformace konceptuálního modelu (entitních množin s jejich vztahy) na schéma relační databáze. Přesně tak to je i v případě této bakalářské práce. Znalost zásad normalizace je rovněž užitečná při posuzování kvality návrhu ER diagramu. K ohodnocení kvality návrhu tabulky slouží tzv. normální formy, jež jsou popsány níže, přičemž platí, že čím vyšší číslo normalizační formy je dosaženo, tím je návrh schématu databáze kvalitnější.

Před rozborem normálních forem je třeba si definovat některé důležité pojmy:

- **Doména** – množina přípustných hodnot, kterých může nabývat daný atribut.
- **Kandidátní klíč** – sloupec nebo kombinace sloupců, v nichž mají všechny řádky tabulky unikátní hodnoty. Každý kandidátní klíč tak umožňuje jednoznačně identifikovat každý řádek tabulky. Jeden z kandidátních klíčů slouží jako primární klíč.
- **Funkční závislost** – situace, kdy hodnota jednoho atributu jednoznačně určuje hodnotu jiného atributu téže relace.
- **Tranzitivní závislost** – platí-li pro atributy X, Y a Z nějaké relace R funkční závislost $X \to Y$ a $Y \to Z$, pak platí také závislost $X \to Z$.

První normální forma Relace se nachází v první normální formě, právě když všechny její jednoduché domény obsahují pouze atomické hodnoty, čímž se eliminují opakující se skupiny v jednotlivých tabulkách. Při kontrole výchozí relace je třeba ověřit, zdali se v relaci nenachází jakýkoliv složený atribut. V mém návrhu se žádný takový atribut nenalézá, a proto mohu prohlásit, že 1NF je splněna.

Druhá normální forma Normální formy, dále jen „NF“, fungují hierarchicky a proto u 2NF platí, že je splněna i 1NF. Schéma relace R se nachází ve 2NF, právě když je každý její neklíčový atribut plně funkčně závislý na každém kandidátním klíči relace R. Zde je 2NF splněna.

Třetí normální forma Pro třetí normální formu platí, že je splněna 2NF a současně se ve schématu relace R nevyskytuje žádný neklíčový atribut, který je tranzitivně závislý na některém kandidátním klíči relace R. I 3NF je v mém návrhu úspěšně ověřena a splněna.[16]

BNFC Rozšířením 3NF získáme BCNF, celým názvem Boyce-Coddova normální forma, zahrnující předchozí normální formy, plus jednu rozšiřující definici. V drtivé většině případů však platí, že po dosažení 3NF se relace již nachází i v BCNF. Za určitých okolností však relace v BCNF být nemusí a to při průniku následujících tří podmínek:

1. Pokud v relaci existuje více kandidátních klíčů.
2. Když jsou všechny kandidátní klíče složené ze dvou či více atributů.
3. Pokud existuje takový atribut, jenž je společný pro všechny kandidátní klíče.

Definice BCNF říká, že: „Schéma relace R je v Boyce-Codově normální formě, právě když pro každou netriviální funkční závislost $X \rightarrow Y$ je X superklíčem. Superklíčem rozumíme každou nadmnožinu kandidátního klíče relace R.“[16]

Vzhledem k existenci primárních klíčů `OID` generovaných frameworkem do každé tabulky, neexistují v logickém schématu žádné složené kandidátní klíče, z čehož vyplývá, že i BNFC musí být v mém návrhu splněna. Existuje i 4NF a 5NF, které se ale v praxi příliš nepoužívají, a proto se jimi nebudu zabývat ani v tomto návrhu.

5.4 Fyzický model

Finální etapou je návrh fyzického modelu, který se týká organizace dat na fyzické úrovni, což znamená, že zde řešíme, kde budou uloženy tabulky vytvořené v logickém modelu. Cílem je dosáhnout maximální výkonosti při práci s databázovými tabulkami, které jsou uložené ve zvoleném SŘBD a přímo vycházejí z logického návrhu databáze.[16] V případě tohoto informačního systému se jedná o MS SQL server 2016. Kompletní databáze je dostupná na přiloženém CD v podobě SQL skriptu pro vytvoření výchozího stavu databáze, který je nezbytný k zajištění nastavení práv uživatelských rolí a vytvoření administrátorského účtu pro první přihlášení do systému. Struktura obsahu přiloženého CD se nachází v příloze B.

K optimalizování přístupu k datům databázových tabulek na fyzické úrovni se využívají tzv. databázové indexy sloužící ke zrychlení přístupu k datům a měly by se používat u všech sloupců, podle kterých se vyhledává, třídí, nebo podle kterých se spojují tabulky. Indexy se vytvářejí nad jedním nebo více sloupci tabulky, přičemž platí že každá tabulka může mít indexů několik. Index definovaný nad sloupcem tabulky umožňuje rychlý přístup k záznamům podle hodnot tohoto sloupce.[13]

Fyzické schéma tohoto systému bylo optimalizováno pomocí indexů s využitím frameworkové komponenty XPO, o které jsem se již zmiňoval v rozboru technologií v podkapitole 4.3. XPO umí automaticky generovat dva typy indexů. U primárních klíčů je využit clusterovaný unikátní index, zatímco pro cizí klíče existuje neclusterovaný index, který na rozdíl od clusterovaného indexu neseřazuje data, což ho dělá při vyhledávání časově náročnějším. Nevýhodou těchto indexů je vyšší časová složitost při vkládání záznamů nebo mazání dat z tabulky, což je daň za výrazné urychlení při vyhledávání ve velkém množství dat.

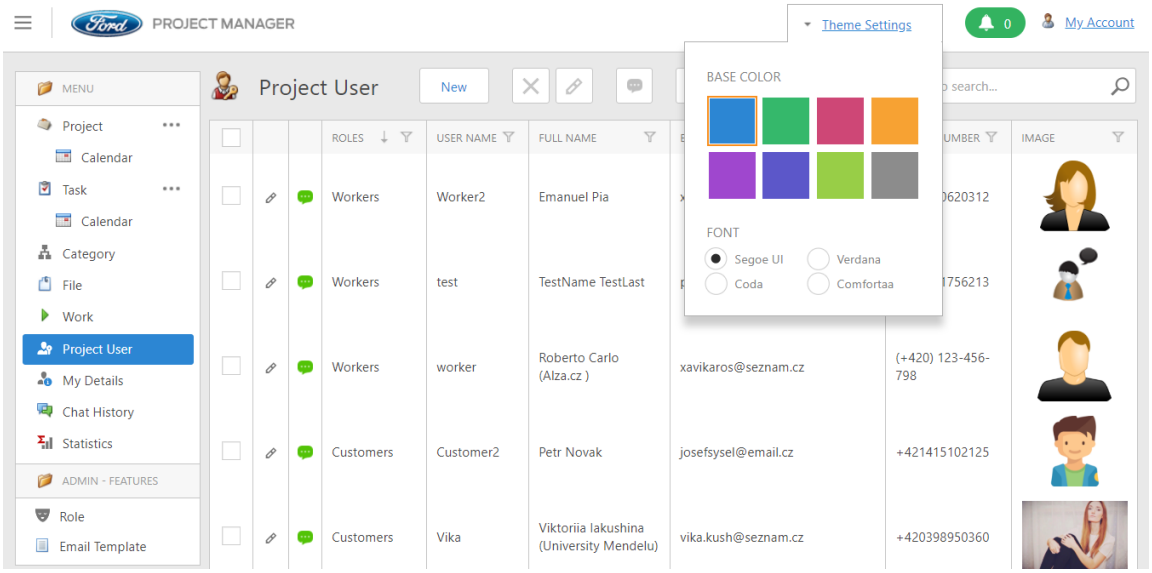
Kapitola 6

Implementace

K programování bylo použito Visual studio 2018, které pro mne bylo jasnou volbou, jelikož je aplikace vyvíjena na operačním systému Windows 10 a je použit programovací jazyk C#, webový framework ASP.NET a framework XAF. Všechny tyto zmíněné technologie jsou na vývoj ve Visual Studiu nejvíce určeny.

6.1 Uživatelské rozhraní

Uživatelské rozhraní (UI) je generováno automaticky podle zdrojového kódu tříd díky frameworku XAF, což je obvykle postačující na testování funkčnosti, avšak k uspokojení uživatelů je žádoucí vzhled aplikace vyladit s využitím nástroje zvaného *Model Editor*. Tento editor se pohodlně používá v prostředí Visual Studia, do kterého je po instalaci balíku komponent od DevExpressu integrován. Ukázka části UI ukazující sekci registrovaných uživatelů je na obrázku 6.1



Obrázek 6.1: UI – seznam testovacích uživatelů a rozbalený „theme picker“.

Model Editor ke své funkci používá tzv. Aplikační Model (*Application Model*), což je prostředek pro shromažďování informací z aplikačního kódu, které jsou reprezentovány jako metadata, tedy data definující databázovou strukturu a aplikační funkce skrz neutrální formát, který může být adaptován na kterýkoliv cíl platformy. Tento editor se stal v kombinaci s Aplikačním modelem na pozadí mocným nástrojem umožňujícím prakticky neomezeně přizpůsobovat uživatelské rozhraní.

Aplikační Model je formován z několika vrstev. První vrstva je generována ze zdrojového kódu. Následující vrstvy jsou jsou reprezentovány změnami uloženými v XafML souborech uloženými v odkazovaných modulech. XafML soubor je speciální frameworkový typ souboru určeného k zaznamenávání změn UI aplikace. Finální vrstva reprezentuje změny uložené v aplikačním projektovém XamfML souboru. Všechny tyto změny se navzájem překrývají, když je Aplikační Model načten. Mimo jiné, nachází se zde ještě `User.Model.xafml` soubor obsahující změny zapříčiněné koncovými uživateli v předchozím spuštění. Tento soubor představuje nejvrchnější vrstvu Aplikačního Modelu, když je načtena během provozu aplikace. *Model Editor* je možné využít k editaci kterékoliv vrstvy Aplikačního Modelu a ovlivňovat tak tím výsledný vzhled.[4]

Na obrázku 6.1 je dále patrné, že uživatelé si mohou vybrat svou oblíbenou barvu pozadí nebo jeden ze 4 fontů písma s pomocí funkcionality, která se ukrývá pod tlačítkem *Theme Settings*. Implementace je obsažena ve složce `BaseColorSelector`, kde se nachází *controller* `BaseColorSelector.ascx`, který v závislosti na nastavení aktivuje příslušný vzhled. Nastavení vzhledu je napsáno v souborech s kaskádovými styly a JavaScriptem ve stejné zdrojové složce. Výběr příslušného vzhledu je uložen i pro následující přihlášení.

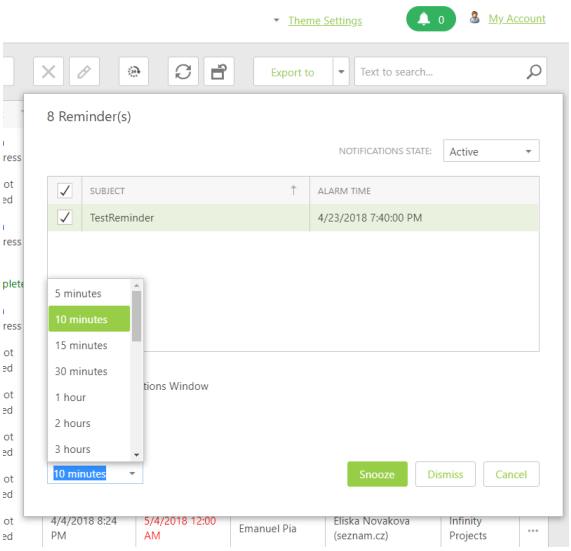
Správce systému má na výběr ze dvou log, přičemž první je univerzální a druhé (momentálně na obrázku 6.1) je navržené pro potřeby konkrétního uživatelského prostředí. Případně je možné na vyžádání nahrát logo jiné podle přání konkrétního správce systému, který si chce přenést zdrojové kódy na svůj server.

UI je plně responzivní a jeho vzhled vypadá uspokojivě na velké obrazovce monitoru, obrazovce tabletu a dokonce i na displeji chytrého telefonu. Při zúžení prohlížecího okna dochází nejprve ke skrytí levého základního menu a následně při dalším zmenšování okna začne postupné skrývání sloupců od méně prioritních napravo až po prioritnější sloupce nalevo. Jedná se o funkci „adaptive designu“, která byla nastavena v Model Editoru, kde každý sloupec dostal číslo určující jeho prioritu (čím menší číslo, tím větší priorita).

6.2 Reminder a kalendář

Reminder a kalendář jsou implementovány s využitím modulů frameworku XAF, které se používají pomocí dědičnosti příslušných rozhraní modulů. Tyto Moduly jsou využity ve třídách `Project` a `Task`.

6.2.1 Reminder



Obrázek 6.2: UI – Reminder

Reminder slouží k připomínání blížících se termínů týkající se dokončení projektů a úkolů. Funguje takovým způsobem, že příslušný uživatel nastaví očekávaný potřebný čas pro připomenutí během zakládání nového projektu/úkolu. Pokud žádný nenastaví, zůstává výchozí hodnota, která činí 1 hodinu pro úkol a 1 den pro projekt. Uživatel ukládá čas do pole `RemindIn`, který je datového typu `TimeSpan`, a je z toho počítán na pozadí čas atributu `AlarmTime`, což již je čas ke spuštění upomínky. Spuštění této události probíhá pomocí vyskakovacího okna ukázaného na obrázku vlevo 6.2. Připomínku je následně možné buďto zavřít (Cancel), zrušit (Dismiss) a nebo odložit (Snooze) o určitý čas, který je nabízený v rozbalovacím menu umístěném vlevo dole na zmiňovaném obrázku.

Implementace využívá modulu zvaného `NotificationsModul`, který je naprogramován za použití dědičnosti z frameworkového rozhraní `ISupportNotification`[1]. Z rozhraní byly do tříd `Project` a `Task` implementovány atributy `AlarmTime`, `UniqueId`, `NotificationMessage` a `IsPostponed`.

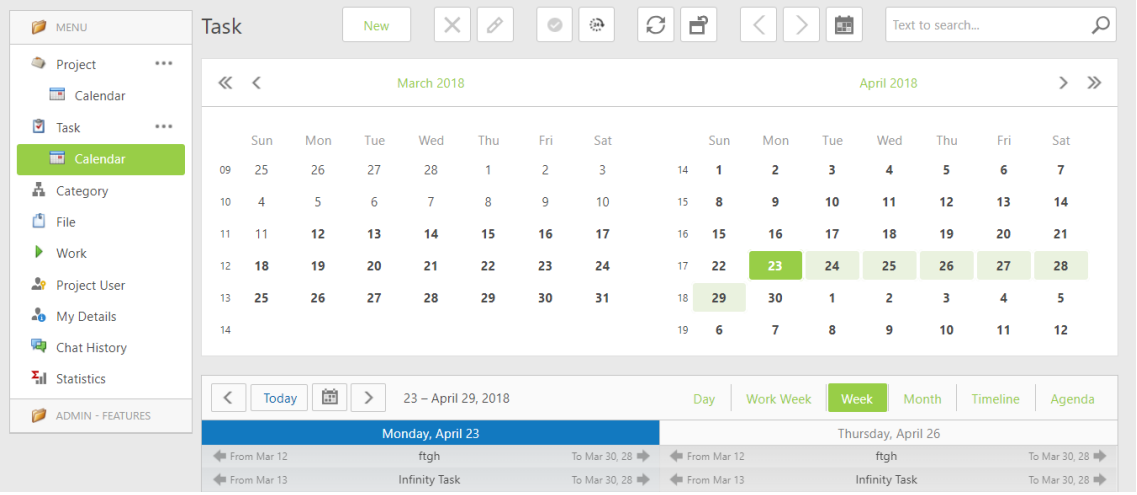
6.2.2 Kalendář

Kalendář je součástí tzv. `Scheduler Modulu`[2], což je modul využívající ke své funkci dvě komponenty. První z nich je `WinForms Scheduler`, která emuluje vzhled, atmosféru a schopnosti Microsoft Outlook Scheduleru. Jeho hlavními funkcemi je denní, týdenní, měsíční, timeline a agenda náhled. Druhou komponentou je `ASP.NET Scheduler` zajišťující vyhovující zobrazení kalendáře ve webovém prostředí.

Celý modul kalendáře je do aplikace připojen s použitím dědičnosti rozhraní `IEvent`, ze kterého jsem implementoval následující atributy: `Subject`, `Description`, `StartOn`, `EndOn`, `AllDay`, `Location`, `Label` a `Status`. Každý atribut je současně vlastností objektů třídy `Project` a `Task`. Kalendáře jsou implementovány nezávisle na sobě, což odděluje vykreslení obsahu projektů od úkolů a naopak. Obsah je následně možné číst v UI kalendáře. Ukázka grafického vzhledu kalendáře se nalézá na obrázku 6.3 na další straně.

[1] https://documentation.devexpress.com/eXpressAppFramework/DevExpress.Persistent.Base.General.ISupportNotifications.class

[2] https://documentation.devexpress.com/eXpressAppFramework/112811/Concepts/Extra-Modules/Scheduler/Scheduler-Module



Obrázek 6.3: UI – Kalendář úkolů.

6.3 Statistiky

Pro lepší přehlednost jsem se rozhodl implementovat třídu se statistikami o odvedené práci a zasloužených finančních odměnách za počet odpracovaných sekund. Všechny potřebné atributy jsou implementovány ve třídě `Statistics`, kde je obsažena instance třídy `ProjectUser`, `Task` a `Project`. K uložení hodnoty celkového času a celkové platby slouží atributy `TotalTime` a `TotalPayment`. Se zmíněnými atributy pracuje další třída nazývající se `StatisticsResults`, která se s jejich pomocí dotazuje pomocí SQL dotazu znázorněném na obrázku 6.4, aby vyhledala v databázových tabulkách správné hodnoty, na jejichž základě spočítala statistiky. Tato položka je k dostupná jenom k nahlédnutí a pouze administrátorům.

```
@"SELECT SUM(w.Payment) AS [TotalPayment], SUM(w.WorkTime) AS [TotalWorkTime], pu.Oid AS Worker, p.OID as Project, t.OID as Task
FROM
ProjectUser pu JOIN TaskWorker tw ON (pu.OID = tw.Workers)
JOIN PermissionPolicyUser ppu ON (ppu.Oid = pu.OID AND ppu.GCRecord is NULL)
JOIN Task t ON (tw.Tasks = t.OID AND t.GCRecord is NULL AND ISNULL(t.IsCompleted, 0) = 0)
JOIN Work w ON (t.OID = w.Task AND w.GCRecord is NULL AND w.WorkTime > 0 and w.Worker = pu.Oid)
JOIN Project p ON (t.Project = p.OID AND p.GCRecord is NULL)

GROUP BY p.OID, t.OID, pu.Oid"));
```

Obrázek 6.4: SQL dotaz nad databází ve třídě `StatisticsResults`

Tento SQL dotaz postupuje tím způsobem, že spojí databázové tabulky pomocí JOIN, ze kterých je žádoucí získat hodnoty atributů. Jedná se o tabulky: `ProjectUser`, `TaskWorker`, `PermissionPolicyUser`, `Task`, `Work` a `Project`. Každá z těchto tabulek má i speciální atribut `GCRecord`, který využívám k vyfiltrování pouze platných nesmazaných záznamů s pomocí podmínek ověřujících, zdali tento záznam obsahuje hodnotu `NULL`, což automaticky znamená, že tento řádek je platný, respektive není označen jako smazaný. Pro výpočet celkového času a platby slouží vestavěná SQL funkce SUM. Nakonec je k agregaci hodnot do správných skupin použita konstrukce GROUP BY, která tak vykonává podle hodnot primárního klíče `OID` z tabulek projektu, úkolu a uživatele.

6.4 E-mailové notifikace

Součástí aplikace je možnost automatického odesílání e-mailů za určitých okolností. Jedná se o následující situace:

- Založení nového uživatele – Každý nově vytvořený uživatel obdrží e-mail s přístupovými údaji ke svému účtu. Heslo si musí při prvním přihlášení změnit dvojnásobným zadáním stejného (nového) hesla.
- Obnova nebo změna hesla – Pokud uživatel zapomněl své stávající heslo, má šanci si nechat zaslat heslo nové, které si však musí při následujícím přihlášení změnit opět na vlastní.
- Nový projekt – S uložením nového projektu je odesílán notifikační e-mail všem řešitelům obsaženým v daném projektu.
- Nový úkol – S uložením nového úkolu je odesílán notifikační e-mail všem řešitelům zahrnutým do daného úkolu.
- Odstranění projektu – Při smazání projektu dostanou všichni přidělení uživatelé e-mail s upozorněním na tuto událost.
- Odstranění úkolu – Při smazání úkolu dostanou všichni přidělení uživatelé e-mail s upozorněním na tuto událost.
- Nové konverzační vlákno – Jakmile kterýkoliv uživatel obdrží zprávu v rámci nově založeného chatovacího vlákna, dostává na e-mailové upozornění.
- Právě dokončený projekt – Pokud hlavní řešitel označí stav projektu jako dokončený, řešitelé daného projektu dostávají e-mailovou zprávu o této události.

K implementaci jsem zvolil postup, který zde bude podrobněji vylíčen. Prvně ze všeho je potřeba uložit textovou šablonu e-mailové zprávy, určenou k odeslání do uživatelských schránek elektronické pošty. Obsahy všech e-mailových šablon jsou uloženy v databázové tabulce `EmailTemplates`, do které se dostávají s pomocí třídy `Updater`. Tato třída obsahuje frameworkovou metodu `UpdateDatabaseAfterUpdateSchema()`, která má za úkol aktualizovat data v databázi. Jednou z prováděných operací je i zavolání metody `CreateTemplates(IObjectSpace os)` s parametrem instance rozhraní `IObjectSpace`[3], která volá další metodu `CreateTemplate(IObjectSpace os,string code,string subject,string text)` s parametry zahrnujícími, kromě instance `ObjectSpace`, i identifikační řetězec šablony (například: new_user) uložený ve sloupci `Code` tabulky s šablonami, řetězec předmětu e-mailu a řetězec obsahu těla e-mailové zprávy. Volaná metoda `CreateTemplate()` převezme parametry a použije je k přidání obsažených hodnot do kolekce třídy `EmailTemplate`. Tímto jsou data přidána do databáze.

Druhou částí je samotné načtení správné šablony a odeslání e-mailu. Pro každou šablonu existuje ve třídě `EmailSender` jedna metoda, která musí být zavolána, pokud nastává některá ze sedmi situací uvedených na začátku této podkapitoly. Příkladem je metoda `SendNewUserPasswordMail()`, znázorněná na obrázku 6.5. Tato metoda vyžaduje parametr instance rozhraní `IDataLayer`[4], instanci třídy `ProjectUser` kvůli atributům objektu

[3] https://documentation.devexpress.com/eXpressAppFramework/DevExpress.ExpressApp.IObjectSpace.class
[4] https://documentation.devexpress.com/CoreLibraries/DevExpress.Xpo.IDataLayer.class

jednotlivých uživatelů a řetězec obsahující heslo, které je náhodně generováno v šestiznakovém formátu metodou `GeneratePassword(6,0)` z třídy náležící k .NET frameworku a jmenného prostoru `System.Web.Security`. Uvnitř těla metody je poté potřeba udělat novou instanci rozhraní `UnitOfWork`[5]. Unit Of Work reprezentuje logické připojení do databáze, sledování změn, které se propíší do databáze a je vstupním bodem pro sestavení dotazů nad databázemi a je součástí nástroje XPO 4.3. Dále je třeba přistoupit s pomocí `UnitOfWork` k datům tabulky `EmailTemplate` a použitím LINQ[6] dotazu na XPO vyhledat požadovaný řádek tabulky obsahující vyžadovaný typ šablony.

```
public static void SendNewUserPasswordMail(IDataLayer dl, ProjectUser pu, string password)
{
    if (pu == null)
        return;

    using (UnitOfWork uow = new UnitOfWork(dl))
    {
        EmailTemplate et = new XPQuery<EmailTemplate>(uow).FirstOrDefault(a => a.Code == "new_user");
        if (et == null)
            throw new Exception("No template found.");

        var rp = new Dictionary<string, string> //init a new dictionary
        {
            { "password", password }
        };
        string subject = ReplacePlaceholders(et.Subject, pu, rp);
        string text = ReplacePlaceholders(et.Text, pu, rp);
        Send(uow, pu.Email, subject, text, et.Priority);
    }
}
```

Obrázek 6.5: Tělo metody `SendNewUserPasswordMail()` ve třídě `EmailSender`

Potom je nezbytné přetransformovat proměnnou s vygenerovaným heslem do speciální kolekce `Dictionary`, aby se s ní následně mohlo lépe pracovat v pomocné metodě `ReplacePlaceholders(string text,object obj,Dictionary<string,string> runtimeParameters=null)`, která je odsud volána, aby nahradila tzv. „placeholder“[7] konkrétní hodnotou. Placeholdery jsou v textových řetězcích šablon ohraničených symboly složených závorek: „{“ (začátek) a „}“ (konec). `ReplacePlaceholders` přijme a validuje správnost dat, aby zavolala další pomocnou metodu `Replace` z třídy `CommonUtilities`, která se postará o správné nahrazení placeholderů za konkrétní hondnoty (jméno, datum, stav projektu,...) a současně i odstranění již nadbytečných složených závorek. Výsledné textové řetězce jsou vráceny v proměnné `result` typu `string` zpátky do `SendNewUserPasswordMail()`. Zde je volána konečná metoda (obsažena ve stejné třídě) `Send()`) s parametry viditelnými na již zmíněném obrázku 6.5.

`Send(UnitOfWork uow,string emailAdr,string subject,string text,EmailTemplatePriority priority)` slouží ke správnému poskládání e-mailové zprávy a k nastavení vlastností SMTP serveru. Kromě instance `UnitOfWork`, předmětu a těla zprávy přijímá i e-mailové adresy uživatelů, kterých se akce týká, a prioritu zprávy. Tato metoda dále používá

[5] https://documentation.devexpress.com/CoreLibraries/2138/DevExpress-ORM-Tool/Feature-Center/Connecting-to-a-Data-Store/Unit-of-Work

[6] https://documentation.devexpress.com/CoreLibraries/4060/DevExpress-ORM-Tool/Feature-Center/Querying-a-Data-Store/LINQ-to-XPO

[7] Určitý řetězec znaků určen k nahrazení konkrétní hodnotou.

pomocnou instanci třídy `Settings`, ze které využívá atributy k nastavení všech potřebných parametrů týkající se nastavení síťových služeb, jako například adresa URL, SMTP Host, SMTP Port, e-mailová adresa odesílatele a jeho jméno.

Obsah těla, předmět a priorita e-mailové zprávy nejsou pevně určeny, nýbrž je možné je zcela přizpůsobovat v UI pod administrátorským přístupem. Tělo zprávy využívá značky jazyka HTML i CSS, které je také možné libovolně vkládat i editovat. Přístup k těmto atributům je realizovatelný v podobě detailního náhledu položky Email Template v sekci menu Admin – Features.

6.5 Chat

Pro vyvolání funkce chatu je nutné kliknout na zelenou ikonku, která je nepřehlédnutelná již v listovém náhledu uživatele, jenž se má stát adresátem. Bezprostředně poté vyskočí aplikační pop-up okno, kde se zadává subjekt a první zpráva nového konverzačního vlákna. Po potvrzení odeslání první zprávy tlačítkem *Submit* je otevřena nová záložka v prohlížeči ve formě chatovacího okna. Pro dokončení tohoto procesu je z uživatelského pohledu často ještě potřebné povolit prohlížeči otevírání vyskakovacích oken.

Základem chatu je soubor `ActiveChats.aspx`, který byl upraven podle jedné z Bootsnip[8] šablon. Soubor obsahuje CSS definující vzhled chatovacího okna a JavaScript, který mimo jiné obsahuje volání metod několika handlerů[9] z dalších tříd umístěných v adresáři AJAX podílejících se na implementaci chatu. Principem je každé 3 sekundy volat funkci `UpdateContent()`, která kontroluje, zdali pro ni má server nová data. Pokud byly nalezeny nová data ke stažení, jsou předávána do jednotlivých HTML divů, aby následně došlo k překreslení již neaktuálních dat tvořící chatovací okno.

V souboru `availableChats.ashx.cs` je třída `availableChats`, kde jsou implementovány dva parametry: ID uživatele (`uid`), který chatuje a ID konkrétního chatu (`activeChat`). Metoda používá tyto parametry k vyhledání aktivních chatů a vytvoří z nich HTML kód, který se nejprve zkonvertuje do JSONu a pak je vykreslen v UI chatovacího okna s jednotlivými zprávami.

V `chatMessages.ashx.cs` se nachází handler `ProcessRequest(HttpContext context)` zobrazující všechny zprávy do HTML kódu chatu, který je té chvíli otevřený. Metoda `FormatDate(DateTime dt)` formátuje čas podle toho, zdali byla zpráva přijata do 1 minuty (v UI se zobrazí textový řetězec „Just Now“), v intervalu 1 až 2 minut („Minute ago“) a při starších zprávách vypisuje počet minut (do 1 hodiny) nebo počet hodin od obdržení určité zprávy.

V souboru `storeChatMessage.ashx.cs` je implementováno ukládání proběhlé komunikace do kolekce `ChatMessage`. Tímto způsobem dochází k zaznamenávání historie konverzace do databáze. V případě úspěšného uložení je vynucen reload, což znamená opětovné načtení celého chatu aktuální stránky.

Přístup dotazování se serveru každé 3 sekundy není z pohledu programovacích paradigmat ideální scénář, avšak pro požadavky této menší aplikace je plně vyhovující. Řešením pro rozsáhlý projekt volající po profesionální optimalizaci by bylo kupříkladu využití ASP.NET knihovny SignalR[10] umožňující oboustrannou komunikaci mezi klientem a serverem, což serverům umožňuje předat nové dostupná data připojeným klientům v okamžiku, kdy jsou serverem zaznamenány.

[8] Šablona chatu je dostupná na: https://bootsnipp.com/snippets/0e3Ma
[9] Handler vyjadřuje funkci, která je propojena s nějakou událostí.
[10] Více o technologii SignalR na: https://www.asp.net/signalr

Kapitola 7

Testování

Testování aplikace je klíčová záležitost, protože během programování je nutné ověřovat funkčnost programu a rovněž kontrolovat, do jaké míry je současný stav odrazem analýzy požadavků. Testování probíhalo dvěma způsoby: Prvním bylo průběžné testování s pomocí debuggeru vývojového prostředí Visual Studio Enterprice[1]. Druhý způsob byla zpětná vazba od potenciálního koncového uživatele systému.

7.1 Testování programátorem

V průběhu programování byla aplikace průběžně testována s pomocí debuggeru Visual Studia, které umožňuje po kompilování zdrojových kódů aplikaci spustit ve webovém prohlížeči a simulovat systém prakticky v reálné podobě s pomocí spojení s lokálním databázovým systémem. Lokální databáze je udržována díky softwaru Microsoft SQL server 2016[2]. Do databáze byly v průběhu vývoje vkládány různé testovací hodnoty podle tehdejší potřeby. Především bylo nutné velmi důkladně otestovat možnosti jednotlivých uživatelských rolí, které mají různá práva a tím pádem zodpovídají za zobrazený obsah a možnosti jeho editace. O rolích bylo více napsáno v kapitole o analýze požadavků 3.

Součástí průběžného testování bylo vylaďování uživatelského rozhraní. Vzhled aplikace je sice generován s pomocí XAF automaticky, ale pro dosažení nejlepší naplnitelné uživatelské přívětivosti bylo vhodné do vzhledu průběžně zasahovat s přičiněním tzv. Model Editoru, což je speciální frameworkový editor sloužící k úpravě výchozího uživatelského rozhraní. Osobně jsem stav UI testoval nejen na displeji svého laptopu, ale i na velkém monitoru, tabletu a chytrém telefonu. Ve všech případech jsem výsledný vzhled nakonec dostal do uspokojivého stavu.

7.2 Testování uživatelem

Čím více se blížila finální podoba webové aplikace, tím potřebnější byla interakce s potenciálními koncovými uživateli k získání zpětné vazby, ujasnění požadavků a odhalení nedostatků. Již v první čtvrtině celkové doby programování jsem získal dvě osoby, které projevily zájem o nasazení aplikace do praxe pod svou administrativou a současně splňovaly kritéria cílové skupiny charakterizované v podkapitole 3.1. Hlavním rysem těchto dvou testovacích uživatelů bylo, že vyvíjeli výdělečnou činnost pro více zákazníků ve spolupráci

[1] Podrobnosti o vývojovém prostředí Visual Studia na: https://www.visualstudio.com
[2] Podrobnosti o vývojovém prostředí SQL Serveru: www.microsoft.com/en-us/sql-server/sql-server-2016

s dalšími osobami, což mělo zaručit, že budou mít dostatečnou motivaci systém zkoušet a přemýšlet nad tím, v čem by jim mohl ještě výrazněji přispět. Prvním testovacím uživatelem byl muž středního věku pracující jako soukromý vývojář webových stránek, který na projektech obvykle spolupracoval se svým kamarádem. Druhou vhodnou osobou byl muž mladšího věku, pracující jako grafický designer v malé soukromé firmě se čtyřmi zaměstnanci.

Uživatelé zkoušeli do systému vkládat různé hodnoty v roli administrátora a odhalovali nedostatky. Testování proběhlo v několika nepravidelných iteracích. Iterace byly vykonávány tak, že po získání zpětné vazby jsem se vždy snažil sám otestovat a najít chyby nahlášené testery, abych je sám posoudil a rozhodl se, jestli se opravdu o chybu jedná, a zdali je vůbec možné ji opravit. Například jedním z návrhů na zkvalitnění produktu bylo implementovat automatické generování faktur podléhající obchodnímu zákoníku a zákonu o účetnictví. Tento návrh jsem ale z časových důvodů ponechal pouze jako předmět budoucí optimalizace. Naopak, upozornění týkající se příliš strohé e-mailové šablony jsem akceptoval a text v šablonách více rozvinul. Aplikace byla následně opravena a vrácena uživatelům k opětovnému vyzkoušení.

7.3 Testování na vlastní doméně

Výrazné usnadnění testování znamenalo nasazení webové aplikace na webový server pod svou doménovou adresou bp.josefmanagerapp.eu, která slouží i pro demonstraci výsledků této bakalářské práce. Aplikace se tak stala snadno dostupnou i pro další osoby, které jsem nechal si aplikaci vyzkoušet. Nejednalo se však o potenciální cílové uživatele, takže jsem ani nevyžadoval zpětnou vazbu.

Nejtěžší na testování se nakonec překvapivě stalo řešení nastavení uživatelských práv, které byly implementovány jako poslední část. Role `Administrators`, kterou nebylo nutno moc omezovat právy, již byla téměř kompletní před nasazením na webový server, ale zbývající dvě role `Workers` a `Customers` byly komplikovanější. Proto jsem společně s dvěma testovacími uživateli vytvořil v systému několik fiktivních uživatelských profilů a snažil se vžít do potřeb lidí ve zbývajících dvou rolích, což mi dopomohlo k vyladění práv pro všechny potenciální uživatele.

7.4 Testování firmou

Po dokončení aplikace do stavu, kdy jsem ji již považoval za připravenou pro nasazení do ostrého provozu byla umístěna na server české pobočky firmy Ford, kde byla dostupná pro zaměstnance v rámci vnitřní sítě v data centru společnosti. Před nasazením systému bylo pouze vytvořeno další logo přizpůsobené značce této automobilky, a které zůstalo nastaveno až do nynějšího stavu.

Motivací ze strany českého Fordu bylo zajištění koordinace reklamních týmů z reklamních agentur zajišťující Fordu marketingovou realizaci jeho obchodních plánů. Pro koordinaci je žádoucí sladit kampaň externích agentur s požadavky společnosti Ford. Jedná se o externí firmy spolupracující s Fordem a někdy současně i mezi sebou na různých projektech, jako například vytvoření webových stránek, reklamních bannerů, novinových článků a dalších propagačních materiálů.

Současný stav vypadá tak, že komunikace a většina přenosů materiálů probíhá pomocí e-mailové komunikace, což sice není ideální, ale zároveň pohodlnější cesta, která však způsobuje občasná nedorozumění a zmatky. Největší chaos nastává tradičně na konci každého

kvartálního období, kdy je třeba ke každému modelu automobilu zaktualizovat ceníky a prodejní plány. Pro všechny subjekty by mohlo být výhodnější pracovat se stejnými daty uloženými na jednom místě a nahradit e-mailovou koordinaci centrálním systémem.

Prakticky to vypadalo tak, že zákazníky se stali někteří z marketingových manažerů Fordu, zatímco řešiteli jsou členové externích týmů a administrátorem je osoba zodpovědná za deployment aplikace na server, která založila okolo 20 uživatelských účtů. Jednalo se pouze o pokus, zdali by se tento systém mohl stát populárním, aby pak případně mohl být více přizpůsoben a optimalizován potřebám firmy.

Testování trvalo doposud 1 týden a stále probíhá (květen 2018) a výsledky jsou zatím pouze omezené ohlasy, které se týkají funkcionality. Uživatelé postrádají ukazatel pracovního postupu, jenž by se jim líbil, vyjádřený v procentech. Funkce položky Work a Statistics se k seriozním účelům vůbec nepoužívá, jelikož jsou zaměstnanci v tomto případě placeni klasickou hodinovou mzdou. Hojně se využívá chat, soubory, projekty, kategorie a úkoly. Řešitelé si taktéž žádali o více práv, jako například k vytváření nových kategorií nebo mazání proběhlé konverzace. Taktéž mi bylo řečeno, že UI není dost intuitivní, moderní a některé operace v něm jsou pomalé. Zřídka některým uživatelům chyběla absence českého jazyka.

Kapitola 8

Závěr

Záměrem první části této bakalářské práce bylo seznámit se s již existujícími řešeními informačních systémů pro správu projektů dostupných na internetu a zanalyzovat a stanovit požadavky na svůj vlastní informační systém správy projektů se zaměřením pro OSVČ. Druhá část byla na rozdíl od té první praktičtější a zabývá se návrhem vlastního řešení na základě ucelených požadavků vyplývající z teoretické analýzy.

Závěrem lze konstatovat, že zadání této bakalářské práce bylo splněno. Systém splnil požadavky, které na něj byly kladeny a je použitelný v praxi, což dokazuje i jeho provizorní nasazení na serveru firmy, která aspiruje na to, aby se stala prvním zákazníkem. Výsledek dokonce v některých částech lehce překročil svůj původní rozsah, ve kterém jsem například nepočítal s implementací funkce reminderu, e-mailových notifikací, přepínače vzhledu pozadí a fontů v UI a nebo generování statistik o odvedené práci za určitou hodinovou sazbu.

Demonstrační verze webové aplikace je plně funkční bez žádných omezení a dostupná pro všechny uživatele s platnými přihlašovacími údaji na webové adrese bp.josefmanagerapp.eu. Postup k nasazení webové aplikace na webový server, včetně konfiguračních a výkonnostních požadavků na cílový server, je zdokumentován v příloze A.

Během vývoje jsem narazil na myšlenky o budoucí optimalizaci, pokud by se stal tento software úspěšným. Potenciálním rozšířením je automatické generování faktur, které by využívalo již existující informace k výpočtům, uspořádání a nakonec vytištění legitimní faktury ve formátu PDF. Z testování bylo zjištěno, že některým uživatelům chybí měřítko pracovního postupu. Dalším námětem bylo vyladění UI do modernější podoby, což je záležitost, která se s časem mění podobně jako móda nebo trendy a nikdy proto není úplně dokončená. Funkcionalita pro možnost nastavení pravidelného opakování úkolů po určitou časovou periodu by mohla být také užitečná a zamyslet bych se mohl i nad rozšířením do dalších jazyků, abych rozšířil potenciální uživatelskou komunitu. A nakonec jedním z běžných standardů dnešní doby je propojování služby s aplikacemi třetích stran. Mohlo by se například jednat o služby Google, GitHub, Slack či DropBox.

Literatura

[1] Avaza: *Avaza*. 2018, [Online; navštíveno 25.04.2018].
URL https://www.avaza.com/

[2] Developer Express Inc.: *DevExpress ORM Tool*. [Online; navštíveno 14.04.2018].
URL https://documentation.devexpress.com/CoreLibraries/1998/DevExpress-ORM-Tool

[3] Developer Express Inc.: *FIPS – compliance changes*. 2018, [Online; navštíveno 30.04.2018].
URL https://www.devexpress.com/Support/Center/Question/Details/T501305/fips-compliance-changes-to-the-devexpress-persistent-base-passwordcryptographer-and

[4] Developer Express Inc.: *Model Editor*. 2018, [Online; navštíveno 30.04.2018].
URL https://documentation.devexpress.com/eXpressAppFramework/112582/Concepts/Application-Model/Model-Editor

[5] Developer Express Inc.: *XAF Architecture*. 2018, [Online; navštíveno 17.04.2018].
URL https://documentation.devexpress.com/eXpressAppFramework/112559/Fundamentals/XAF-Architecture

[6] Fowler, M.: *Destilované UML*. Grada Publishing, a.s., 2009, ISBN 978-80-247-2062-3.

[7] Freelo: *Freelo*. 2018, [Online; navštíveno 26.04.2018].
URL https://app.freelo.cz

[8] Kulhan, J.; Lehocký, Z.: *Normalizace relačních databází*. 2008, [Online; navštíveno 28.03.2018].
URL http://programujte.com/clanek/2008071900-normalizace-relacnich-databazi/

[9] Microsoft: *Entity Framework Documentation*. 2016, [Online; navštíveno 02.05.2018].
URL https://docs.microsoft.com/en-us/ef/

[10] Nette Foundation: *Nette Framework*. 2018, [Online; navštíveno 17.04.2018].
URL https://nette.org/cs/

[11] Techopedia Inc.: *Globally Unique Identifier (GUID)*. [Online; navštíveno 31.03.2018].
URL https://www.techopedia.com/definition/1208/globally-unique-identifier-guid

[12] Trello: *Trello*. 2018, [Online; navštíveno 25.04.2018].
URL https://trello.com/

[13] Vrána, J.: *Využití databázových indexů*. 2003, [Online; navštíveno 10.05.2018].
URL https://www.root.cz/clanky/vyuziti-databazovych-indexu/

[14] Welling, L.; Thompsonová, L.: *PHP a MySQL – rozvoj webových aplikací*. SoftPress, s.r.o., 2004, ISBN 80-86497-60-7.

[15] Yalla: *Yalla*. 2018, [Online; navštíveno 25.04.2018].
URL https://www.yallahq.com/

[16] Zendulka, J.; Rudolfová, I.: *Databázové systémy (IDS), Studijní opora*. FIT VUT v Brně, Červenec 2006, [Online; navštíveno 27.03.2018].

Příloha A

Deployment

Tato kapitola popisuje nezbytné požadavky k úspěšnému nasazení aplikace na webový server, popis konfigurace a slouží současně i jako návod potencionálním zájemcům o provoz této služby na svém serveru.

A.1 Požadavky

Pro nasazení (deployment) celé platformy do ostrého provozu je vyžadován server s minimálně 2GB RAM a 1GB volného místa na disku. Platforma pro svůj běh vyžaduje Microsoft Windows Server 2008R2 a vyšší s následujícími rolemi:

1. **Web Server (IIS) – Web Server**
   - **1.1. Common HTTP Features**
     - 1.1.1. Default Document
     - 1.1.2. Directory Browsing
     - 1.1.3. HTTP Errors
     - 1.1.4. Static Content
     - 1.1.5. HTTP Redirection
   - **1.2. Health and Diagnostics**
     - 1.2.1. HTTP Logging
     - 1.2.2. Logging Tools
   - **1.3. Performance**
     - 1.3.1. Static Content Compression
     - 1.3.2. Dynamic Content Compression
   - **1.4. Security**
     - 1.4.1. Request Filtering
     - 1.4.2. Basic Authentication
     - 1.4.3. Centalized SSL Certificate Support
     - 1.4.4. IP and Domain Restrictions

1.5. Application Development

1.5.1. .NET Extensibility 4.5

1.5.2. ASP.NET 4.5.

1.5.3. ISAPI Extensions

1.5.4. ISAPI Filters

1.6. Management Tools

1.6.1. IIS Management Console

1.6.2. IIS Management Scripts and Tools

1.6.3. IIS Management Service

Aplikace by měla běžet na IIS ve vlastním aplikačním poolu, s nastavením .NET CLR na verzi 4.0 a Managed Pipeline Mode na hodnotu *Integrated*.

A.2 Konfigurace

Aplikace se konfiguruje pomocí standardního konfiguračního mechanizmu .NET Frameworku a to úpravou souboru `web.config` v rootu webu. Přehled nastavitelných parametrů je k nahlédnutí v tabulce A.1.

Tabulka A.1: Web.Config

Konfigurační parametry	Popis
ConnectionString	Connection string pro připojení do databáze.
Languages	Jazyková mutace systému. Nyní nastaveno na: En
TraceLogLocation	Umístění logovacího souboru – None, ApplicationFolder. Nastaveno na ApplicationFolder.
Diagnostics/Switches/XAF	Úroveň logování aplikačního frameworku: 0–Off, 1–Errors, 2–Warnings, 3–Info, 4–Verbose. Zde nastavena úroveň 2
Diagnostic/Switches/XPO	Úroveň logování databázové vrstvy: 0–Off, 1–Errors, 2–Warnings, 3–Info, 4–Verbose. Zde nastavena úroveň 2
ErrorReportEmail	E-mailová adresa pro odeslání chybových zpráv.
ErrorReportEmailServer	Název SMTP serveru pro odesílání chybových zpráv.
ErrorReportEmailSubject	Subjekt e-mailu.
ErrorReportEmailFrom	E-mailová adresa systému.
ErrorReportEmailFromName	Jméno od koho je adresován e-mail.

Connection string pro konfiguraci s databází vyžaduje následující parametry:

- XpoProvider (název poskytovatele, např. mssqlserver)
- Server = localhost
- Password (heslo k vaší databázi)
- User Id (vaše uživatelské ID (login))
- Database (název konfigurované databáze)
- Encoding = UNICODE

A.3 Seznam podporovaných SŘBD

Tato XAF ASP.NET aplikace podporuje vícero systémů řídící báze dat, jejichž seznam je vyjmenován v tabulce A.2.

Tabulka A.2: Podporované SŘBD

Poskytovatel	Verze systému
Advantage	Advantage Database Server v9.1, Advantage Database Server v10.1, Advantage Database Server v11.1.
ASA	SQL Anywhere 11, SQL Anywhere 12
ASE	Sybase Adaptive Server 12, Sybase Adaptive Server 15.5
DB2	DB2 9.5.2
Firebird	Firebird 1.5, Firebird 2.1.3
MSAccess	Microsoft Jet
MySql	MySQL Server 4.1, MySQL Server 5.0, MySQL Server 5.1 a MySQL Server 5.5
MSSqlServer	MS SqlServer 7.0, MS SqlServer 2000, SQL Azure™ Database, MS SQL Server 2000 Desktop Engine (MSDE 2000), MS SQL Server 2005, SQL Server 2005 Express Edition, SQL Server 2008, SQL Server 2008 R2, SQL Server 2008 R2 Express, SQL Server 2012, 2014 a 2016, SQL Server 2012, 2014 a 2016 Express
MSSqlServerCE	MS SqlServer 2005 Mobile, SQL Server 2005 Compact Edition
Oracle	Oracle 9i, Oracle 10g, Oracle 11g
Pervasive	Pervasive PSQL 9, Pervasive PSQL 10, Pervasive PSQL 11
Postgres	PostgreSQL 7, PostgreSQL 8, PostgreSQL 9
SQLite	SQLite 3
VistaDB	VistaDB 4, VistaDB 5

A.4 Postup instalace

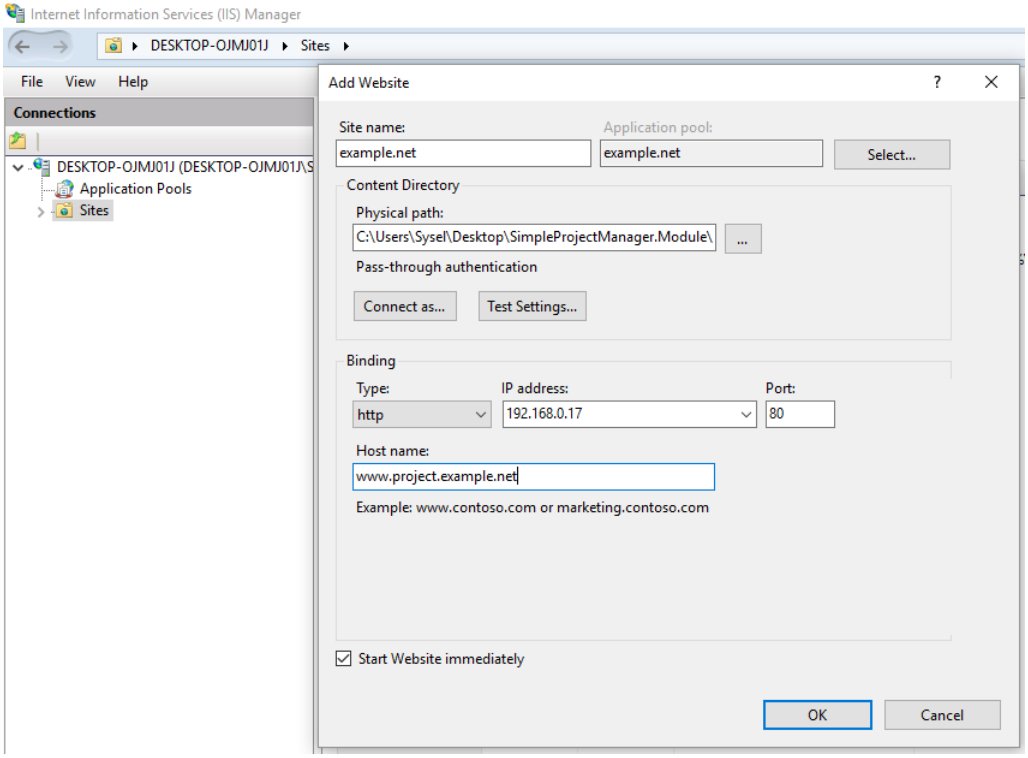
Zde se nachází výchozí a současně doporučený postup k instalaci webové aplikace na server. Prerekvizitou je vykonávat instalaci na OS Windows, kde je potřebné mít administrátorská práva. Co se softwaru týče, uživatel potřebuje mít nainstalovaný **Internet Information Services** (IIS)[1], který je možné najít v prostředí Windows Server přímo v Server Manageru jako jednu z rolí (Web Server). K nahrání databázových dat je vhodné mít nainstalovaný **Microsoft SQL Server Management Studio** (SSMS)[2].

1. Začněte tím, že spustíte program IIS.
2. Pak je třeba kliknout na položku *Sites* pravým tlačítkem myši a poté zvolit *Add Website*.

[1] https://www.microsoft.com/en-us/download/details.aspx?id=48264
[2] https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017

3. Otevře se nové pop-up okno viz obr. A.1 s následujícími položkami ke konfiguraci:

   - *Site name:* Název webové stránky, který vyplní i název aplikačního poolu.
   - *Application pool:* Ideální je vytvořit aplikaci vlastní pool, proto zanechte předvyplněné pole se stejným názvem jako obsahuje předchozí pole *Site name*.
   - *Physical path:* Výběr cesty k adresáři v lokálním PC, kde se nachází zdrojové soubory aplikace připravené k deploymentu.
   - *Type:* Zde jsou dvě možnosti:
     - https – se zapezpečením certifikátu SSL[3]
     - http – bez SSL zabezpečení, což je i současný stav testovací verze
   - *IP address:* Volba IP adresy serveru, na které bude aplikace dostupná.
   - *Port:* Určuje, na kterém portu bude aplikace přístupná (standardní webový je port 80, v případě SSL se jedná o port 443).
   - *Host name:* Které doménové jméno chcete použít. Slouží k rozlišení aplikací běžících na stejné IP adrese a stejném portu zároveň.
   - *Start Website immediately:* Pro bezprostřední spuštění webové sítě.



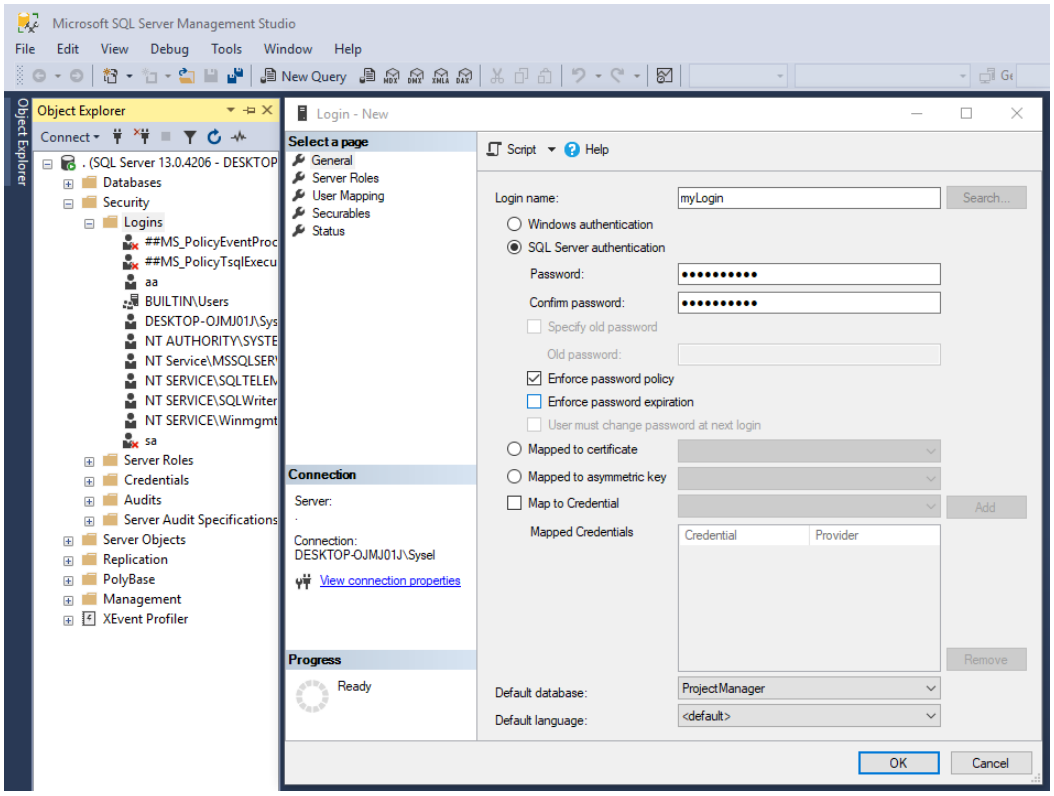
Obrázek A.1: Pop-up okno – Add Website (příklad) v IIS

[3] https://support.microsoft.com/cs-cz/help/324069/how-to-set-up-an-https-service-in-iis

4. Spusťte program MS SQL Server Management Studio (SSMS).

5. Vytvořte si v **Security** nový login po vzoru obrázku A.2, jelikož nechceme, aby aplikace běžela pod výchozím administrátorským účtem.

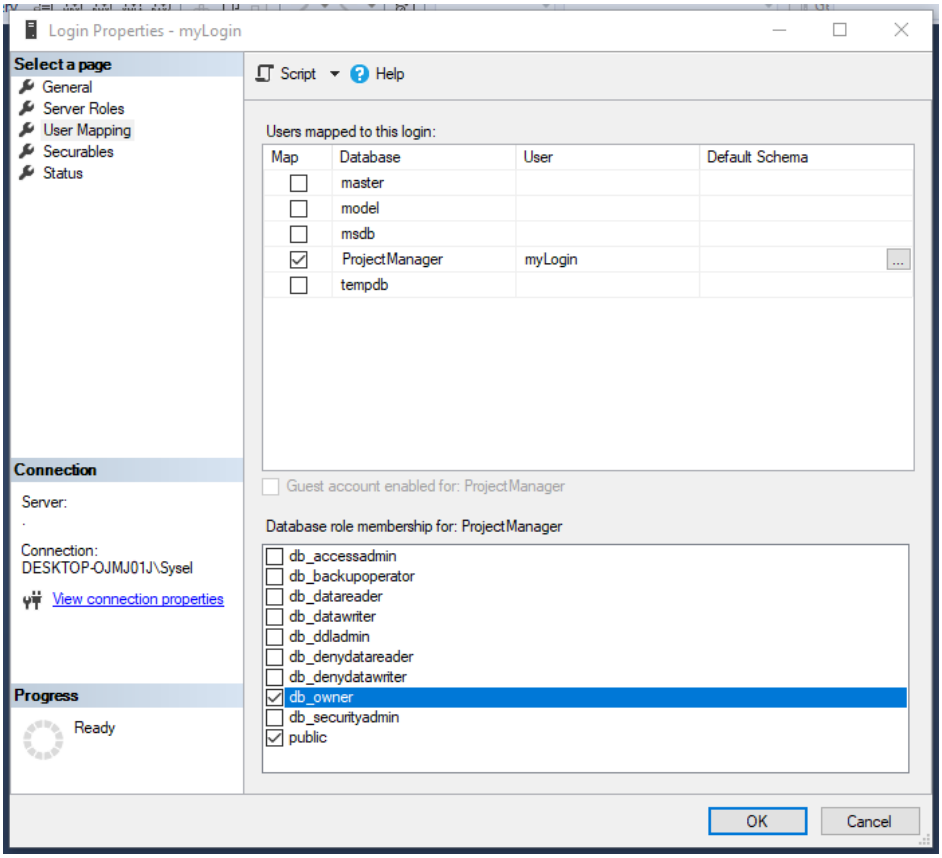
   - *Login name:* Jednoduchá volba loginu (použitého v `ConnectionStringu`).
   - *SQL Server authentication:* Důležité je označit toto pole, aby se následně zpřístupnila volba hesla (použitého v `ConnectionStringu`).
   - *Enforce password policy:* Pokud nechcete, aby Vás systém donutil dodržovat zásady bezpečnosti při volbě hesla, odznačte toto pole.
   - *Enforce password expiration:* Pokud nechcete každý měsíc měnit heslo, odznačte pole o expiraci.
   - *Default Database:* Zvolte databázi, která patří k webové aplikaci.



Obrázek A.2: Vytvoření nového loginu v SSMS

6. Otevřete položku **User Mapping** dle obrázku A.3, nacházející se na další straně. Zde je nutné přidělit k uživateli explicitní práva k databázi.

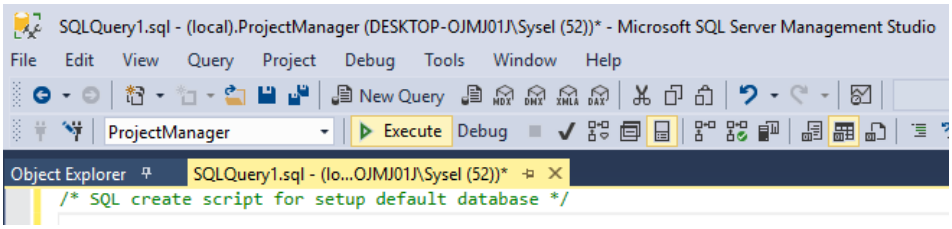
   - Nejprve označte příslušnou databázi k deploymentu.
   - Označte roli `db_owner`, aby měl XAF právo k zásahu do struktury databázového schématu.
   - Potvrďte provedené změny.



Obrázek A.3: Vytvoření nového loginu v SSMS

7. **Spuštění SQL scriptu:** Na závěr už zbývá jen nahrát SQL create script do databáze za účelem vytvoření výchozího schématu databázových tabulek.

   - V SSMS je třeba otevřít SQL script, který se nachází na přiloženém CD. Použijte k tomu lištu horního menu, kde je třeba postupně otevřít: *File*, *Open* a *File*. Alternativou je klávesová zkratka **Ctrl + O**.
   - Otevře se pop-up okno, v němž najděte a vyberte z adresáře soubor obsahující příslušný SQL script.
   - Poté je nutné vybrat správnou databázi ze seznamu lokálních databází.
   - Nakonec použijte tlačítko **Execute** ke spuštění scriptu viz obrázek A.4.



Obrázek A.4: Výběr a spuštění SQL scriptu

Příloha B

Obsah přiloženého CD

Struktura obsahu CD

- Zdrojové soubory:
  - Aplikační zdrojové soubory
  - SQL script k vygenerování výchozí databáze
- Dokumenty:
  - BP.pdf
  - Zdrojové kódy LaTeX
  - README.md