ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ
FAKULTA STAVEBNÍ
Katedra technologie staveb

DIPLOMOVÁ PRÁCE

Elektronický stavební deník

Bc. Vít Dajbych

2016

Vedoucí diplomové práce: doc. Ing. Pavel Svoboda, CSc.

Prohlášení autora

Prohlašuji, že jsem předkládanou diplomovou práci vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

V Praze …… **…….**

Jméno a příjmení diplomanta

Poděkování

Chtěl bych touto cestou poděkovat doc. Ing. Pavlu Svobodovi, CSc. za jeho vstřícnost a cenné rady při vedení této práce.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Thákurova 7, 166 29 Praha 6

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Dajbych Jméno: Vít Osobní číslo: 395646

Zadávající katedra: Katedra technologie staveb

Studijní program: Stavební inženýrství

Studijní obor: Příprava, realizace a provoz staveb

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce: Elektronický stavební deník

Název diplomové práce anglicky: Electronic site diary

Pokyny pro vypracování:
Provedení analýzy současné legislativy o možnostech vedení stavebního deníku v elektronické podobě. Na základě provedené analýzy stavu příslušné legislativy provedení naprogramování a dokumentace webové aplikace plnící funkci elektronického stavebního deníku a to ve vztahu k praktickým zkušenostem a potřebám jeho vedení v běžné "papírové" praxi.

Seznam doporučené literatury:
[1] Zákon č. 183/2006 Sb., o územním plánování a stavebním řádu včetně novel
[2] Vyhláška 499/2006 Sb., o dokumentaci staveb
Dokumenty ČKAIT

Jméno vedoucího diplomové práce: doc. Ing. Pavel Svoboda, CSc.

Datum zadání diplomové práce: 3. 10. 2016 Termín odevzdání diplomové práce: 8. 1. 2017
Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku

Podpis vedoucího práce Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v diplomové práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce" a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací".

13.10.2016
Datum převzetí zadání Podpis studenta(ky)

Abstrakt

Tato diplomová práce pojednává o návrhu webové aplikace, která plní funkci elektronického stavebního deníku. Zabývá se současnou legislativou týkající se vedení stavebních deníků a jejího vlivu pro tento návrh. Dále práce pojednává o elektronickém podepisování a způsobu jeho používání v této aplikaci. Nakonec práce shrnuje výhody a nevýhody elektronického stavebního deníku oproti, v současné době používanému, klasickému stavebního deníku.

Klíčová slova

stavební deník, elektronický stavební deník, elektronický podpis, kvalifikovaný certifikát, certifikační autorita, kryptografie, web, databáze, ASP.NET, SQL, C#

Abstract

This thesis is about the designing of a web application that performs the function of a site diary. It deals with current legislation regarding the management of the site diaries and its impact on this design. Furthermore, the work deals with on the electronic signing and the way of its use in this application. Finally, the work summarizes the advantages and disadvantages of electronic site diary, compared to currently used, classical site diary.

Key words

site diary, electronic site diary, electronic signature, qualified certificate, certificate authority, cryptography, web, database, ASP.NET, SQL, C#

Obsah

Úvod

Přestože stavební zákon uvádí, že je možné za určitých podmínek vést stavební deník elektronicky, není tato možnost v dnešní době prakticky využívána. Vedení stavebních deníků klasickou formou má oproti elektronické verzi některé nedostatky. Mezi ně patří například nečitelnosti ručního písma či podpisů nebo nedostatečný prostor pro uvedení všech dodavatelů či poddodavatelů stavby. Nelze také zaručit, zda byly záznamy do stavebního deníku zapisovány postupně během průběhu výstavby nebo jestli nejsou ve stavebním deníku falešné podpisy. Elektronická verze stavebního deníku s pomocí digitálního podepisování může těmto problémům předejít.

Tato diplomová práce nejdříve zkoumá současnou legislativu ohledně vedení stavebních deníků a její dopad na návrh elektronického stavebního deníku. Dále je v této práci teoreticky vysvětleno, co je to elektronický podpis a jak elektronické podepisování funguje. Elektronický stavební deník je zdokumentován a demonstrován ukázkami zdrojových kódů. V praktické části je pak konkrétní používání elektronického podepisování popsáno. Důraz je také kladen také na bezpečnost, proto práce řeší zabezpečení hesel uživatelů této aplikace. Vzhledem k tomu, že se stavební deník musí archivovat po dobu deseti let, je řešena možnost archivování elektronického stavebního deníku.

V závěru práce jsou shrnuty výhody a nevýhody elektronického stavebního deníku oproti klasickému a je také zdůvodněno, proč je program řešen formou webové aplikace.

1 Současný stav problematiky

Stavební deník musí splňovat zákonné požadavky. Proto je v následujících kapitolách pojednáváno o legislativě, která se týká náležitostí stavebního deníku a dopadu této legislativy na návrh elektronického stavebního deníku. O stavebních denících pojednává zákon číslo 183/2006 Sb., o územím plánování a stavebním řádu, vyhláška číslo 499/2006 Sb., o dokumentaci staveb a vyhláška číslo 503/2006 Sb., o podrobnější úpravě územního řízení, veřejnoprávní smlouvy a územního opatření.

1.1 Zákon č. 183/2006 Sb., o územním plánování a stavebním řádu

Stavební zákon uvádí povinnost vlastníka stavby *uchovávat stavební deník po dobu 10 let od vydání kolaudačního souhlasu, popřípadě od dokončení stavby, pokud se kolaudační souhlas nevyžaduje* [1]. Z tohoto důvodu bude ve webové aplikaci možnost stavební deník exportovat do PDF souboru, který může vlastník stavby uchovávat po zmíněnou dobu.

Paragraf číslo 157 stavebního zákona dále pojednává o některých náležitostech stavebního deníku:

> *Stavební deník nebo jednoduchý záznam o stavbě je povinen vést zhotovitel stavby, u stavby prováděné svépomocí stavebník. Záznamy do nich jsou oprávněni provádět stavebník, stavbyvedoucí, osoba vykonávající stavební dozor, osoba provádějící kontrolní prohlídku stavby a osoba odpovídající za provádění vybraných zeměměřických prací. Záznamy jsou dále oprávněny provádět osoby vykonávající technický dozor stavebníka a autorský dozor, jsou-li takové dozory zřízeny, koordinátor bezpečnosti a ochrany zdraví při práci, působí-li na staveništi, autorizovaný inspektor u stavby, pro jejíž provedení vydal certifikát podle § 117, a další osoby oprávněné plnit úkoly správního dozoru podle zvláštních právních předpisů.* [1]

Návrh aplikace proto musí respektovat oprávnění osob zapisovat do elektronického stavebního deníku. Otevřít elektronický stavební deník pro úpravy

bude moci pouze jeho zakladatel. Tím se zabrání možnosti zápisu do deníku osobám, které k tomu nemají oprávnění. Vlastník elektronického stavebního deníku nechá prostřednictvím aplikace zapisovat ostatní oprávněné osoby záznamy do tohoto deníku. Jméno a podpis zapisovatele se bude získávat z certifikátu, který musí zapisovatel vlastnit. S pomocí digitálního certifikátu se záznam v deníku digitálně podepíše. Digitálním podpisem záznamu v deníku se zároveň zajistí, že nepůjde záznam dodatečně upravovat. Pokud by k takové úpravě došlo, digitální podpis by nebyl platný.

Dále je v zákoně uvedeno, že *po dokončení stavby předá její zhotovitel originál stavebního deníku nebo jednoduchého záznamu o stavbě stavebníkovi* [1]. Pro tento účel bude sloužit export deníku do PDF souboru.

1.2 Vyhláška č. 499/2006 Sb., o dokumentaci staveb

V příloze číslo 9 vyhlášky č. 499/2006 Sb. je uvedeno, jaké obsahové náležitosti musí mít stavební deník:

> *A. Identifikační údaje*
> *a) název stavby (nebo její části) podle jejího ohlášení, stavebního povolení, veřejnoprávní smlouvy nebo oznámení stavebního záměru s certifikátem autorizovaného inspektora, datum jejich vydání, popřípadě číslo jednací,* [2]

Jedná se o vyplnění názvu stavby. Při vytváření nového stavebního deníku proto bude formulář obsahovat tuto možnost. Dále text přílohy pokračuje:

> *b) místo stavby,*
> *c) obchodní firma, místo podnikání nebo sídlo účastníků výstavby (není-li účastník výstavby zapsán v obchodním rejstříku jeho jméno a příjmení):*
> *- zhotovitele (resp. zhotovitelů částí stavby)*
> *- stavebníka (investora)*
> *- projektanta*
> *- poddodavatelů,* [2]

Klasické stavební deníky nemají mnohdy dostatek prostoru pro vyplnění všech poddodavatelů, jak zákon požaduje. U elektronické verze bude možné přidávat zhotovitele a poddodavatele podle potřeb. Příloha dále uvádí:

> *d) jména a příjmení osob zabezpečujících odborné vedení provádění stavby podle § 153 stavebního zákona s rozsahem jejich oprávnění a odpovědnosti,* [2]

Tímto je odkazováno na stavbyvedoucího a stavební dozor. Pro tyto osoby bude rovněž nutné vyplnit rozsah jejich oprávnění a odpovědnosti.

> *e) jména a příjmení osob, vykonávajících technický dozor stavebníka a autorský dozor (jsou-li tyto dozory zřízeny),* [2]

Pro konkrétní návrh aplikace se jedná o stejné požadavky jako v minulém bodě, ale pouze bez dodatečné informace o rozsahu oprávnění a odpovědnosti. Formulář při tvorbě nového stavebního deníku nabídne možnost vyplnit informace o těchto osobách, nebudou však vyžadovány. Pokud se nechá formulářové pole prázdné, nebudou v deníku uvedeny. Stavebník deník dále musí obsahovat:

> *f) jména, příjmení a funkce dalších osob, oprávněných k provádění záznamů do stavebního deníku podle § 157 odst. 2 stavebního zákona,* [2]

Tento paragraf pojednává o těch osobách: stavebník, stavbyvedoucí, osoba vykonávající stavební dozor, osoba provádějící kontrolní prohlídku stavby, osoba odpovídající za provádění vybraných zeměměřických prací, technický dozor stavebníka a autorský dozor, koordinátor bezpečnosti a ochrany zdraví při práci, autorizovaný inspektor a osoby oprávněné plnit úkoly správního dozoru. Nadále:

> *g) údaje o projektové a ostatní technické dokumentaci stavby, včetně jejich případných změn,*
> *h) seznam nebo odkazy na dokumenty a doklady ke stavbě (smlouvy, povolení, souhlasy, správní rozhodnutí, protokoly o kontrolách, zkouškách, přejímkách apod.),* [2]

Pro účely mého řešení bude možnost zadávat údaje o projektové a ostatní technické dokumentaci a později bude možné zapisovat i změny. Při zakládání nového

deníku půjde vyplnit seznam dokumentů a dokladů a po založení také zapisovat jejich doplňky. Dále je v příloze uváděno:

> *i) změny zhotovitelů stavby nebo odpovědných osob během výstavby.* [2]

Zde je opět výhoda elektronického deníku oproti klasickému. V elektronické verzi není problém vnášet velké množství změn a nejsme limitování velikostí papíru. Nakonec je uváděno:

> *Osoby, vykonávající vybrané činnosti ve výstavbě podle § 158 stavebního zákona, prokazují oprávnění k výkonu těchto činností otiskem svého razítka a podpisem ve stavebním deníku. Totéž platí při změně těchto osob v průběhu výstavby.* [2]

Do elektronického deníku, na rozdíl od klasického, není možné otisknout razítko. Abychom vyhověli tomuto požadavku, bude ve webové aplikaci možné nahrát otisky autorizačních razítek.

1.3 Vyhláška č. 503/2006 Sb., o podrobnější úpravě územního řízení, veřejnoprávní smlouvy a územního opatření

Ve vyhlášce č. 503/2006 Sb. uvádí §18q:

> *e) zda je veden stavební deník, popřípadě jednoduchý záznam o stavbě [u ohlašovaných staveb uvedených v § 104 odst. 1 písm. e) až k) stavebního zákona],* [3]

Kontrolovat lze vedení jak elektronického, tak i klasického stavebního deníku. Z tohoto pohledu pro návrh této aplikace neplynou žádné důsledky. Dále text vyhlášky obsahuje:

> *a) soulad vytyčení prostorové polohy stavby s ověřenou dokumentací nebo projektovou dokumentací stavby, hloubku základové spáry, provedení ochrany před škodlivými vlivy vnějšího prostředí (radon, spodní voda, seismicita, poddolování, ochranná a*

> *bezpečnostní pásma); kontrolu stavební úřad provádí na základě předložených dokladů, zápisu ve stavebním deníku nebo v jednoduchém záznamu o stavbě nebo při vlastním zjišťování stavu stavby nebo pozemku,* [3]

Stavební úřad může tedy vyžadovat nahlédnutí do stavebního deníku. Jako u klasického stavebního deníku osoba vedoucí stavební deník může ukázat záznamy v elektronickém stavebním deníku.

1.4 Elektronický podpis

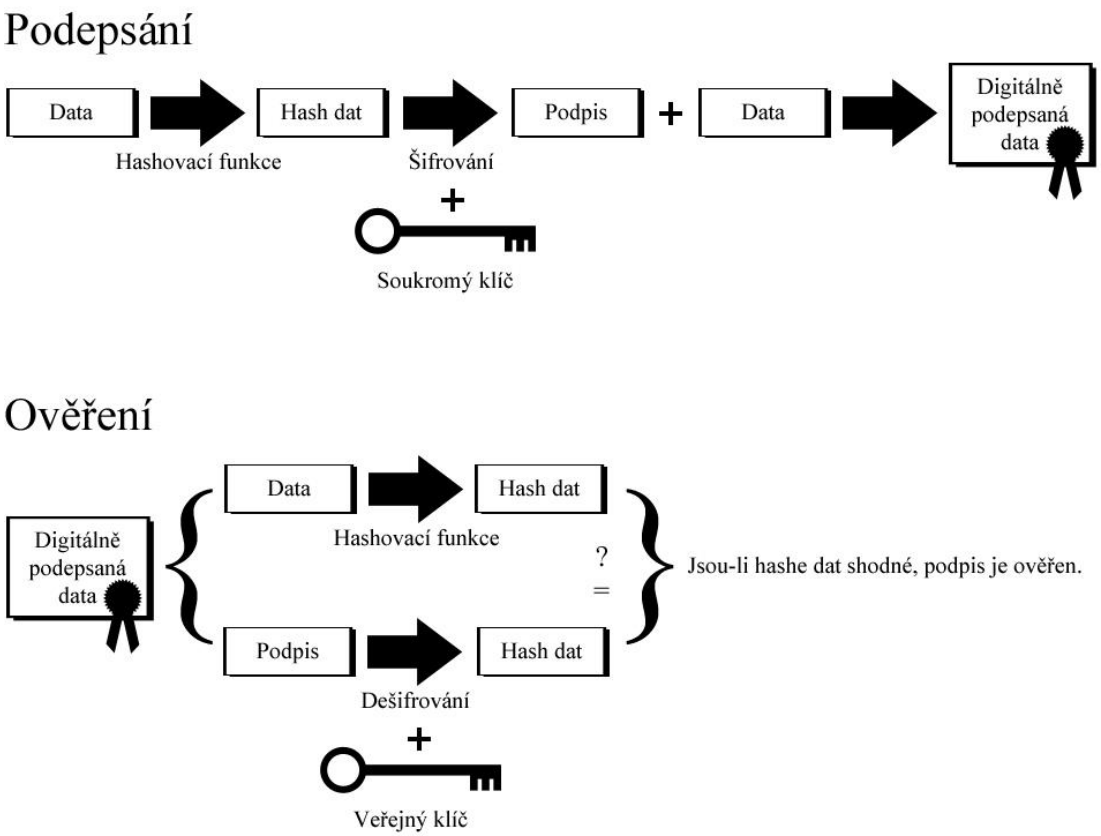
Elektronický podpis je elektronická náhrada vlastnoručního podpisu. Podpisy jsou vytvořeny pomocí asymetrické kryptografie. Ta používá dva odlišné klíče. Soukromý klíč a klíč veřejný. Podepisující svým soukromým klíčem data zašifruje. Pro ověření podpisu se data dešifrují veřejným klíčem podepisujícího.

Ve výpočetní technice existují k elektronickému podepisování kvalifikované certifikáty, které vydávají certifikační autority. Certifikáty obsahují informace jako například jméno, soukromý a veřejný klíč, název vydavatele certifikátu, datum platnosti a další.

Z dat[1] se pomocí hashovací funkce spočítá hash (otisk) dat. Ten se zašifruje pomocí soukromého klíče a vznikne podpis. Podpis spolu s daty se bude ukládat do databáze. Při ověřování se z podpisu pomocí veřejného klíče dešifruje hash, který se porovná s hashem původních dat. Pokud jsou shodné, pak je podpis ověřený. Postup podepisování a ověření je znázorněn na následujícím obrázku:

[1] V našem případě se jedná o záznamy do stavebního deníku, které se budou podepisovat.

Obrázek č. 1 Podepisování a ověření elektronického podpisu



1.5 Certifikační autorita

Certifikační autorita je vydavatelem digitálních certifikátů. Její odpovědností je, aby se osoba svým certifikátem nemohla vydávat za někoho cizího. Pokud budeme důvěřovat dotyčné certifikační autoritě, důvěřujeme tím i údajům uvedeným v digitálním certifikátu tohoto vydavatele.

V České republice jsou ministerstvem vnitra definování tři kvalifikovaní poskytovatelé, kteří jsou oprávněni vydávat kvalifikované certifikáty. Ty je možné použít například pro elektronickou komunikaci s úřady státní správy. Mezi kvalifikované poskytovatele certifikačních služeb patří [4]:

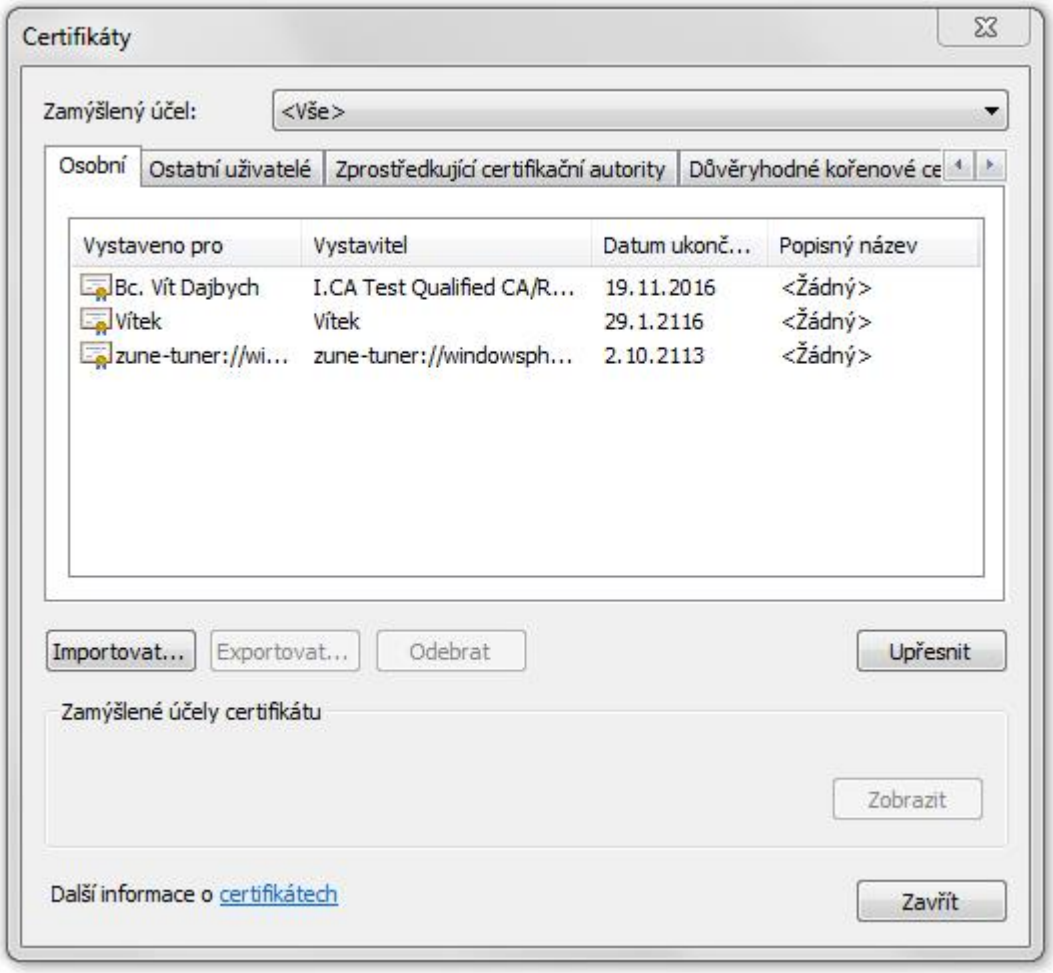
- První certifikační autorita, a. s.,
- Česká pošta, s. p.,
- eIdentity, a. s.

1.6 Standard X.509

Standart X.509 byl přijat jako formát digitálních certifikátů. Původní verze byla vyvinuta v roce 1988 pro jednoduché přiřazení veřejných klíčů jednotlivým subjektům [5]. Dnes je zkratka X.509 nejvíce používána jako označení třetí verze tohoto standartu, který definuje norma RFC 5280. Nejpoužívanějším formátem souboru digitálního certifikátu je PKCS č. 12 (formát Personal Information Exchange). Tento standart povoluje vytváření digitálních certifikátů, které obsahují také heslem zašifrovaný soukromý klíč. Tato aplikace ho bude používat.

Pro účely vyvíjení a testování aplikace jsem si nechal vygenerovat vlastní testovací kvalifikovaný certifikát u společnosti První certifikační autorita, a. s. Tento certifikát[2] jsem si uložil do svého počítače.

Obrázek č. 2 Testovací certifikát v úložišti operačního systému



[2] Vystaven pro Bc. Vít Dajbych.

Tato aplikace bude provozována na webovém serveru, který nemá do úložiště certifikátů v operačním systému počítače přístup. Proto je zapotřebí certifikát exportovat spolu se soukromým klíčem. Tento exportovaný PFX soubor budou uživatelé posílat na webový server za účelem digitálního podpisu dat. Soubor má soukromý klíč chráněn heslem, které se musí při podpisu také zadat.

2 Cíle práce

Cílem této práce je vytvořit a zdokumentovat software, který bude plnit funkci elektronického stavebního deníku. Nezbytnou součástí řešení je implementace digitálního (elektronického) podepisování. Ve stavebním zákoně, který dovoluje vést stavební deník elektronicky, je uvedeno, že všechny zúčastněné osoby musí být majiteli elektronického podpisu.

Webová část se musí naprogramovat tak, aby nebylo možné provádět neoprávněné zásahy do cizích stavebních deníků. Stránka musí mít přístup pouze pro registrované uživatele. Je proto nutné mít v programu implementovány autorizace a autentifikace uživatelů.

Další součástí elektronického stavebního deníku bude nahrávání souborů, aby bylo možné ke stavebnímu deníku přidávat autorizační razítka. Je tedy zapotřebí vyřešit způsob ukládání těchto souborů a znemožnění jejich neoprávněnému zobrazení.

Stavební deník je nutné archivovat po dobu deseti let. Není vhodné, aby byl stavební deník k dispozici pouze na webové stránce. Bude tedy zapotřebí najít vhodný způsob archivace elektronického stavebního deníku.

3 Metody použité k dosažení výsledků

Elektronický stavební deník jsem se rozhodl vyřešit formou webové aplikace. Takto může být aplikace používána na široké škále zařízení a operačních systémech, které mají internetový prohlížeč podporující moderní web.

Pro ukládání dat bude tato aplikace využívat SQL databázi. Vzhledem k tomu, že není žádoucí, aby docházelo k manipulaci dat ve stavebním deníku, je volba úložiště mimo počítač uživatele aplikace bezpečnější. Je to také jeden z důvodů, proč jsem se rozhodl řešit elektronický deník formou webové aplikace. Při vývoji této webové aplikace jsem využíval databázi Microsoft SQL Server 2012.

Webová část programu je naprogramovaná pro ASP.NET verze 4.5 v jazyce C#. Autentifikace uživatelů je řešená pomocí *Froms Authentication*. Kaskádové styly stránky jsou napsány pomocí jazyka Sass, který je ve vývojovém prostředí automaticky přeložen do CSS.

Pro exportování elektronického stavebního deníku do PDF souboru je využita knihovna iTextSharp. Tu je možné používat ve zvoleném prostředí .NET a nabízí možnost digitálního podepisování vygenerovaných souborů kvalifikovanými certifikáty.

4 Praktická část

4.1 Struktura databáze

Pro úplnou funkci elektronického stavebního deníku jsou v SQL databázi vytvořeny tyto tabulky:

Tabulka č. 1 SQL tabulka Diaries

Name	Data type	Length	Allow nulls	Identity
id	int		False	True
owner	int		False	False
created	datetime		False	False
closed	bit		False	False
name	nvarchar	max	False	False
region	nvarchar	max	False	False
village	nvarchar	max	False	False
cadastral	nvarchar	max	False	False

Tato tabulka slouží pro ukládání dat o stavebním deníku. Každý deník má svůj unikátní identifikátor *id.* Ve sloupci *owner* je uloženo *id* uživatele, který stavební deník založil. Datum založení deníku je uloženo ve sloupci *created.* Název stavby je uložen ve sloupci *name*. Následující tři sloupce slouží pro uchování údajů o umístění stavby. Kraj ve sloupci *region,* obec ve sloupci *village* a katastrální území ve sloupci *cadastral.* Další údaje spojené se stavebním deníkem jsou uchovávány v následující tabulce:

Tabulka č. 2 SQL tabulka Diary_details

Name	Data type	Length	Allow nulls	Identity
id	int		False	True
diary_id	int		False	False
initial	bit		False	False
changed	bit		False	False
datetime	datetime		False	False
key	nvarchar	max	False	False
value	nvarchar	max	False	False
phone	nvarchar	max	True	False
value_extended	nvarchar	max	True	False
old_value	nvarchar	max	True	False
old_phone	nvarchar	max	True	False
old_value_extended	nvarchar	max	True	False

V této tabulce jsou ukládány jména a příjmení účastníků výstavby, poddodavatelů, seznam dokumentů a dokladů, údaje o technické a ostatní dokumentaci. Také jsou zde uchovávány změny a doplňky těchto údajů. Jako každá SQL tabulka obsahuje i tato svůj unikátní identifikátor *id*, který bude používán při změnách údajů. Sloupec *diary_id* je pro spojení údaje s konkrétním stavebním deníkem, jelikož tato tabulka bude obsahovat údaje všech vytvořených stavebních deníků. Ve sloupci *initial* je informace o tom, zda byl údaj vložen při zakládání deníku, nebo dodatečně přidaný. Bude-li některý údaj pozměněn, zanese se tato informace do sloupce *changed*. V uživatelském rozhraní se pak změny a doplňky budou ukazovat na jiném místě, než původní hodnoty při zakládání deníku.

Protože se jedná o vkládání dat typu klíč-hodnota, avšak s dalšími volitelnými informacemi navíc, zvolil jsem pro uchovávání těchto dat názvy sloupců *key, value.* Pro volitelné ukládání telefonních čísel slouží sloupec *phone.* Ostatní sloupce tabulky pak nemusí obsahovat žádnou hodnotu. Například položka seznamu dokumentů nemusí obsahovat telefonní číslo.

Stavbyvedoucí a stavební dozor musí mít vyplněn rozsah oprávnění a odpovědnosti. K tomu slouží sloupec *value_extended*.

Změna některého údaje v deníku bude uchovávat i informace o původních hodnotách, pro které jsou vyhrazeny zbývající sloupce *old_value, old_phone, old_value_extended.*

Pro záznamy stavby a jejich digitální podpisy slouží následující tabulka:

Tabulka č. 3 SQL tabulka Diary_records

Name	Data type	Length	Allow nulls	Identity
id	int		False	True
diary_id	int		False	False
datetime	datetime		False	False
posted	datetime		False	False
text	nvarchar	max	False	False
text_signed	varbinary	max	False	False
value_extended	nvarchar	max	False	False
signature_name	nvarchar	max	False	False
signature_key	nvarchar	max	False	False

Sloupce *id* a *diary_id* mají funkci jako u předchozí tabulky. Záznamy o postupu prací se mohou zapisovat i později než v den, ve kterém byly prováděny. Pro datum záznamu je sloupec *datetime*, který uživatel bude vybírat v kalendáři uživatelského rozhraní webu. Sloupec *posted* obsahuje datum a čas, kdy se záznam odeslal na webový server. Slouží ke kontrole, řazení a uživatel ho nemůže nijak ovlivnit.

Samotný obsah záznamu je ve sloupci *text*, podpis záznamu ve sloupci *text_signed*. Jméno podepisujícího ve sloupci *signature_name* a jeho veřejný klíč pro ověřování podpisu ve sloupci *signature_key*. Jméno podepisujícího a jeho veřejný klíč jsou získány z kvalifikovaného certifikátu.

Pro ukládání obrázků do databáze slouží následující tabulka:

Tabulka č. 4 SQL tabulka Diary_stamps

Name	Data type	Length	Allow nulls	Identity
id	int		False	True
diary_id	int		False	False
filename	nvarchar	max	False	False
image	varbinary	max	False	False
type	nvarchar	max	False	False

Tabulka slouží pro ukládání obrázků, které uživatel přidělí ke stavebnímu deníku. Primární účel tabulky je vkládání otisků autorizačních razítek. Krom sloupců *id* a *diary_id* se ukládá název souboru ve sloupci *filename*, binární data souboru ve sloupci *image* a formát obrázku ve sloupci *type*.

Zakladatel stavebního deníku má možnost tento deník sdílet s ostatními uživateli této webové aplikace. Tato funkcionalita je nazvaná „Náhledy do deníku“. Pro tyto účely je zřízena následující tabulka:

Tabulka č. 5 SQL tabulka Diary_views

Name	Data type	Length	Allow nulls	Identity
id	int		False	True
user_id	int		False	False
diary_id	int		False	False

Kromě unikátního identifikátoru *id* se v tabulce uloží identifikační číslo uživatele *user_id*, který má možnost náhledu do deníku s identifikačním číslem *diary_id*.

Poslední tabulka v databázi uchovává informace o uživatelích:

Tabulka č. 6 SQL tabulka Users

Name	Data type	Length	Allow nulls	Identity
id	int		False	True
first_name	nvarchar	50	False	False
last_name	nvarchar	50	False	False
password	nvarchar	256	False	False
email	nvarchar	50	False	False
salt	nvarchar	256	True	False
email_token	nvarchar	256	False	False
degree_prefix	nvarchar	50	True	False
degree_append	nvarchar	50	True	False

V této tabulce je uloženo jméno uživatele ve sloupci *first_name*, příjmení ve sloupci *last_name* a hash hesla ve sloupci *password*. Pro emailovou adresu je sloupec *email* a pro kryptografickou sůl[3] sloupec *salt*.

Před dokončením registrace uživatele se musí ověřit jeho emailová adresa. Ta se ověří tak, že si pro tuto emailovou adresu vygenerujeme a uložíme náhodný řetězec. V tomto případě je uložen ve sloupci *email_token*. Na emailovou adresu pošleme zprávu, která bude obsahovat odkaz na ověřovací stránku našeho webu. Tento odkaz může mít například tuto podobu:

[3] Náhodně vygenerovaný řetězec pro zvýšení bezpečnosti hesla.

```
http://stavebni-denik.aspone.cz/confirm-
email.aspx?token=Eg65MdWdpWUMMQ8btoo8FPXPPkAkhNNn9XPoC7sDT%2bQ%3d&em
ail=vitek12%40msn.com
```

Na ověřovací stránce se zkontroluje, zda ověřovací řetězec a emailová adresa získané z odkazu jsou shodné s těmi, které máme uložené v databázi. Pokud ano, je emailová adresa ověřená.

Dále má uživatel možnost uvést své tituly před a za jménem. Pokud tak učiní, budou se zobrazovat spolu s jeho jménem po přihlášení. K tomu je v tabulce sloupec *degree_prefix* pro tituly před jménem a *degree_append* pro tituly za jménem.

4.2 Web

Stránky, které jsou přístupné pouze po přihlášení uživatele, jsou ve složce *dashboard*. V konfiguračním souboru webu *web.config* je pak v části *system.web* definován *Form Authentication*:

```
<authentication mode="Forms">
  <forms loginUrl="logon.aspx" name=".ASPXFORMSAUTH"
    timeout="1440"></forms>
</authentication>
```

Autorizace pro zobrazování stránek ve složce *dashboard*, které jsou pouze pro přihlášené, je definována:

```
<location path="dashboard">
  <system.web>
    <authorization>
      <deny users="?"/>
    </authorization>
  </system.web>
</location>
```

Soubory v této složce, je-li na ně požadavek, jsou zobrazeny pouze přihlášeným uživatelům. V opačném případě je uživatel přesměrován na stránku *logon.aspx* s přihlašovacím formulářem. Stránky ve složce *dashboard* jsou:

- add-document.aspx – Přidání dokumentu do stavebního deníku
- add-participants.aspx – Přidání osoby do stavebního deníku
- close-diary.aspx – Uzavření stavebního deníku
- diaries.aspx – Seznam stavebních deníků
- diary-record.aspx – Vložení záznamu do stavebního deníku

- diary-stamps.aspx – Autorizační razítka
- diary-views-add.aspx – Přidání možnosti náhledu do stavebního deníku
- diary-views.aspx – Náhledy do deníku
- diary.aspx – Stavební deník
- edit-diary.aspx – Úprava stavebního deníku
- export-diary.aspx – Export stavebního deníku
- new-diary.aspx – Založení nového stavebního deníku
- user.aspx – Osobní údaje uživatele
- views.aspx – Náhledy do stavebních deníků

Web má také definován vlastní ovládací prvek pro odesílání souborů *FileUpload.ascx*, který je definován kvůli sjednocení vzhledu. Výchozí HTML prvek pro odesílání souborů nelze vzhledově dostatečně upravit. Proto je tento prvek posunut mimo viditelnou oblast webové stránky, ale zůstává stále plně funkční. Grafickou podobu tohoto prvku převezme vlastní HTML kód, který je na jeho původní pozici:

```
<label class="file-upload">
  <asp:FileUpload ID="InputFile" runat="server" />
  <span class="button">Vybrat soubor&hellip;</span> <span
    id="filename">Soubor nevybrán.</span>
</label>
<script type="text/javascript">
  var fn = document.getElementById("filename");
  document.getElementById('<%=InputFile.ClientID %>')
    .onchange = function () {
    fn.innerHTML = this.value;
  };
</script>
```

Aby bylo možné využívat tento ovládací prvek ve zdrojovém kódu, jsou definovány jeho veřejné vlastnosti:

```
public partial class FileUpload : System.Web.UI.UserControl {
  public HttpPostedFile PostedFile {
    get { return InputFile.PostedFile; }
  }
  public bool HasFile {
    get { return InputFile.HasFile; }
  }
```

```
  public string Accept {
    get { return InputFile.Attributes["accept"]; }
    set { InputFile.Attributes["accept"] = value; }
  }
}
```

Veřejná vlastnost *HttpPostedFile* vrací odeslaný soubor, *HasFile* značí, zda byl soubor vybrán a vlastnost *Accept* nastavuje hodnotu atributu *accept* ovládacího prvku *asp:FileUpload* a vrací jeho hodnotu.

Obrázek č. 3 Ovládací prvek FileUpload

Ovládací prvek *Message.ascx* je určen pro zobrazování oznámení uživatelům. Jedná se o oznámení například špatně zadaného hesla nebo úspěšného zapsání záznamu do stavebního deníku. HTML kód je:

```
<div>
  <asp:Label runat="server" ID="Message_icon"></asp:Label><asp:Label
    runat="server" ID="Message_body"></asp:Label>
</div>
```

Dvojice serverových prvků *asp:label* je pro zobrazení ikony a textu oznámení. Podle zvolené hodnoty veřejné vlastnosti *Type* se nastaví vzhled. Tento prvek je dále definován:

```
public partial class Message : System.Web.UI.UserControl {
  private string _text;
  protected void Page_Load(object sender, EventArgs e) {
    Message_body.PreRender += Message_body_PreRender;
  }
  private void Message_body_PreRender(object sender, EventArgs e) {
    Message_body.Text = _text;
    if (!string.IsNullOrEmpty(_text)) {
      Message_body.Attributes.Add("class", "message");
```

Podle zvoleného typu zprávy je nastaven styl tohoto prvku. Může se jednat o oznámení kladné, které bude zobrazeno zeleně, záporné červeně nebo neutrální žlutě.

```
      if (Type == TypeOption.OK) {
        Message_icon.Text = "<i class=\"fa fa-check fa-lg\"></i>";
        Message_icon.Attributes.Add("class", "message-icon message-
        ok");
        Message_body.Attributes.Add("class", "message message-ok");
      }
      if (Type == TypeOption.Alert) {
        Message_icon.Text = "<i class=\"fa fa-exclamation fa-
        lg\"></i>";
```

```
            Message_icon.Attributes.Add("class", "message-icon message-
            alert");
            Message_body.Attributes.Add("class", "message message-
            alert");
          }
          if (Type == TypeOption.Hint) {
            Message_icon.Text = "<i class=\"fa fa-info fa-lg\"></i>";
            Message_icon.Attributes.Add("class", "message-icon message-
            hint");
            Message_body.Attributes.Add("class", "message message-
            hint");
          }
        }
      }
```

Veřejná vlastnost *Text* vrací a nastavuje obsah zprávy:

```
      public string Text {
        get {
          return _text;
        }
        set {
          _text = value;
        }
      }
```

Následuje veřejná vlastnost *Type* a definice jejího typu:

```
      public TypeOption Type { get; set; }
      public enum TypeOption {
        None,
        OK,
        Alert,
        Hint
      }
    }
```

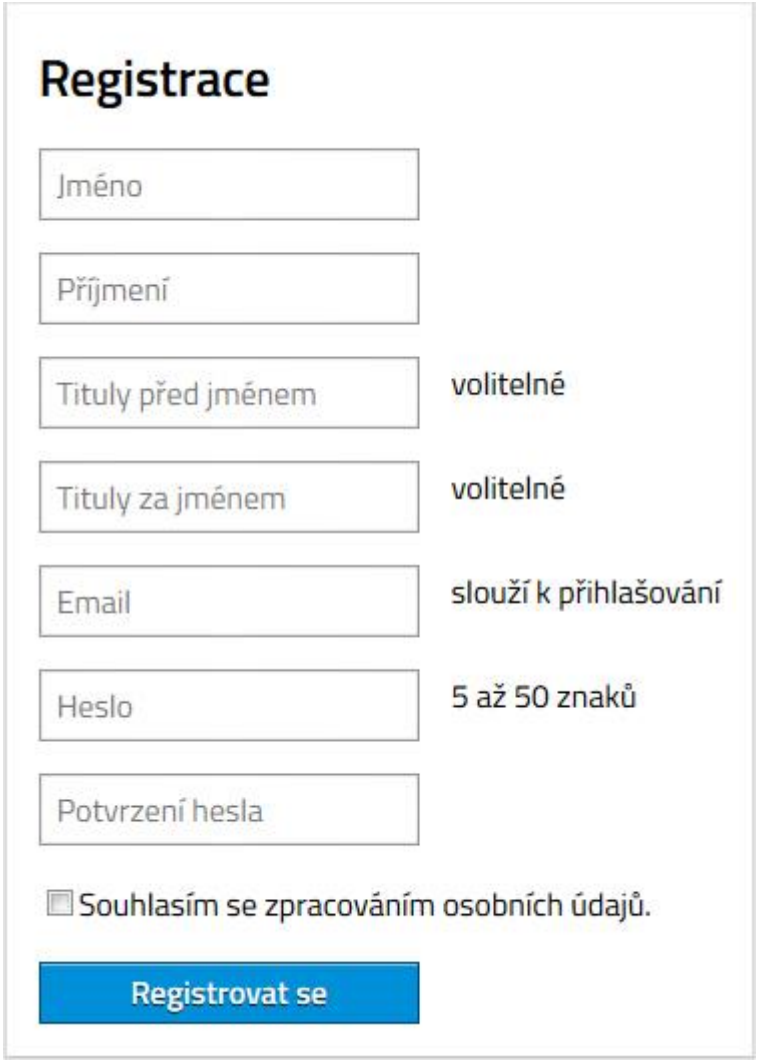
Výsledné podoby toho prvku demonstruje následující obrázek:

Obrázek č. 4 Podoby ovládacího prvku Message

4.2.1 Registrace uživatele

Před používáním webové aplikace se uživatel musí zaregistrovat. K tomu slouží stránka *registration.aspx* (viz příloha č. 1). Vyplňuje se jméno, příjmení, volitelně i své tituly před jménem a za jménem, emailová adresa, heslo a kontrola hesla. Dále musí uživatel souhlasit se zpracováním svých osobních údajů.

Obrázek č. 5 Registrační formulář



Po odeslání těchto informací z formuláře je na straně serveru provedena kontrola. Kontroluje se, zda uživatel souhlasil se zpracováním osobních údajů, jestli jsou vyplněna povinná pole a vyplněný text není moc dlouhý. Také je ověřeno, zda byla zadána platná emailová adresa a jestli se shodují zadaná hesla. Pokud je vše

v pořádku, je odeslána zpráva na zadanou emailovou adresu, která obsahuje odkaz na dokončení registrace a tím se i potvrdí platnost emailové adresy.

Emailová adresa slouží k přihlašování uživatelů, proto musí být v databázi unikátní. Následující kód kontroluje, zda není emailová adresu již použita:

```
using (var connetion = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
    SELECT email FROM users WHERE email = @email", connetion)) {
    cmd.Parameters.AddWithValue("@email", Email.Text);
    connetion.Open();
    using (var reader = cmd.ExecuteReader()) {
      if (reader.HasRows) {
        Message.Text = "Email je již zaregistrován.";
        Message.Type =
          Stavebni_Denik.Controls.Message.TypeOption.Alert;
        return;
      } else {
        unique_mail = true;
      }
    }
  }
}
```

Pokud již v databázi emailová adresa je, na stránce se zobrazí prvek *Message*, kterému se nastaví v atributu *Text* pro uživatele zpráva, že emailovou adresu nelze pro registraci použít. Také se prvku *Message* nastaví atribut *Type* na hodnotu *TypeOption.Alert*, který zapříčiní, že se prvek zobrazí jako varovná hláška.

Pokud emailová adresa nebyla v databázi nalezena, je možné ji pro registraci použít a formulářová data jsou uložena do databázové tabulky *Users* (viz tabulka č. 6). Po uložení dat je odeslána emailová zpráva pro dokončení registrace.

4.2.2 Zabezpečení hesel uživatelů

Z bezpečnostních důvodů se do databáze neukládá heslo uživatele v nezměněné podobě. Pokud by došlo k odcizení dat z databáze, mohl by si útočník hesla jednoduše přečíst. Proto se do databáze ukládá hash hesla.

Pro zvýšení celkové entropie hesla a tudíž i zvýšení bezpečnosti, je pro každého uživatele vygenerováno pole náhodných bajtů (v kryptografii známé jako sůl). Pro tuto aplikaci jsem si zvolil pole o velikosti 128 bajtů. Tyto bajty jsou vygenerovány kryptografickým generátorem pseudonáhodných čísel za použití systémové knihovny *System.Security.Cryptography:*

```
var RNG = new RNGCryptoServiceProvider();
byte[] salt = new byte[128];
RNG.GetNonZeroBytes(salt);
```

Převedeme pole bajtů na řetězec, se kterým se bude v následujících krocích snadněji pracovat:

```
string salt_string = Convert.ToBase64String(salt);
```

Hash spočítáme kryptografickou hashovací funkcí SHA-256 z uživatelského hesla, které je zadáno v textovém poli s identifikátorem *Password1* a z vygenerované soli:

```
byte[] pass = new UTF8Encoding().GetBytes(Password1.Text +
salt_string);
byte[] hash = new SHA256Managed().ComputeHash(pass);
```

Výsledek je převeden na řetězec v hexadecimálním formátu, který bude uložen v databázi:

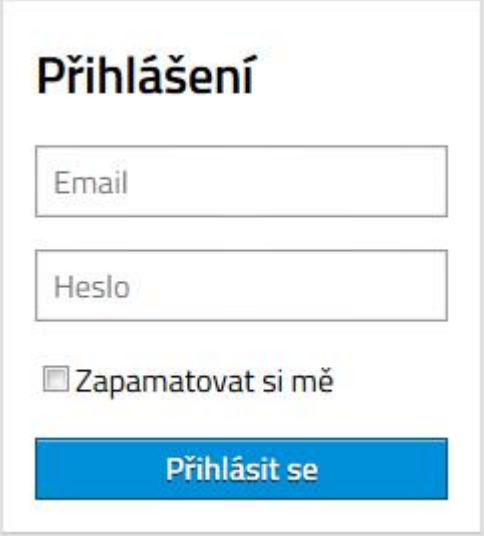
```
StringBuilder passBuilder = new StringBuilder();
for (int i = 0; i < hash.Length; i++) {
  passBuilder.Append(hash[i].ToString("x2"));
}
string pass_hashed = passBuilder.ToString();
```

Pokud by někdo heslo v tomto stavu odcizil, bylo by pro něj obtížné heslo dešifrovat, protože jsme k němu přidali pseudonáhodná čísla.

4.2.3 Přihlašování do aplikace

Na stránce *logon.aspx* (viz příloha č. 2) se může uživatel přihlásit do aplikace a mít přístup ke svým stavebním deníkům. Pro přihlášení do aplikace slouží emailová adresa uživatele a heslo zadané při registraci. Pomocí formuláře jsou zadané údaje odeslány na server, kde je provedena kontrola.

Obrázek č. 6 Přihlašovací formulář



Pomocí SQL dotazu jsou z databáze přečteny údaje, které se pojí se zadanou emailovou adresou:

```
SELECT email_confirmed, id, first_name, last_name, degree_prefix,
degree_append, password, salt FROM users WHERE email = @email
```

Pokud je emailová adresa nalezena, je třeba ověřit heslo. K tomu je nezbytné načíst kryptografickou sůl, připojit ji k heslu a spočítat hash:

```
byte[] pass = new UTF8Encoding().GetBytes(Password.Text +
  (string)reader["salt"]);
byte[] hash = new SHA256Managed().ComputeHash(pass);
StringBuilder passBuilder = new StringBuilder();

for (int i = 0; i < hash.Length; i++) {
  passBuilder.Append(hash[i].ToString("x2"));
}

string pass_hashed = passBuilder.ToString();
```

Hodnota proměnné *pass_hadhed* je porovnána s uloženým hash v databázi, který se vygeneroval při registraci uživatele:

```
if (pass_hashed == (string)reader["password"])
```

Pokud je tato podmínka splněna, tak uživatel zadal správně svou emailovou adresu, heslo a je autentifikován pomocí následující metody:

```
FormsAuthentication.RedirectFromLoginPage(userName.ToString(),
Persist.Checked);
```

V prvním parametru této metody je zadáno uživatelské jméno, které se bude po přihlášení zobrazovat a také identifikační číslo uživatele, které bude na webu skryto, ale je používáno ve zdrojovém kódu. Druhý parametr této metody nastaví, zda se má autentifikace zapamatovat. Pokud bylo v přihlašovacím formuláři zaškrtnuto políčko *Persist* označené textem „Zapamatovat si mě“, nastaví se tento druhý parametr na hodnotu *true*, v opačném případě na hodnotu *false*.

4.2.4 Změna osobních údajů

Stránka *user.aspx* (viz příloha č. 3) zobrazuje osobní údaje uživatele a také nabízí možnost si je upravit. V události stránky *Page_LoadComplete* jsou z databázové tabulky *Users* (viz tabulka č. 6) nahrány do uživatelského rozhraní informace o přihlášeném uživateli pomocí SQL dotazu, kde parametr *id* je identifikační číslo uživatele:

```
SELECT [first_name], [last_name], [degree_prefix], [degree_append]
FROM [Users]
WHERE [id] = @id
```

Událost *Page_LoadComplete* nastane až po události ovládacích prvků stránky (kliknutí na tlačítko „Změnit osobní údaje“). Díky tomu jsou zobrazeny při načtení stránky již případně pozměněné údaje. Uživatel si na této stránce může změnit jméno, příjmení, tituly před jménem, tituly za jménem a také své heslo.

Pokud uživatel klikl na tlačítko „Změnit heslo“, je provedena kontrola formuláře, zda bylo vyplněno původní heslo, nové heslo a potvrzení nového hesla. Pokud je vše vyplněno správně, z databázové tabulky *Users* (viz tabulka č. 6) se zjistí uživatelova kryptografická sůl jednoduchým SQL dotazem, kde parametr *id* je identifikační číslo uživatele:

```
SELECT [salt]
FROM [Users]
WHERE [id] = @id
```

Pomocí nově zvoleného hesla a kryptografické soli se spočítá hash nového hesla. K tomu účelu sloučí moje pomocná funkce *HashPassword*:

```
public static string HashPassword(this string password, string salt)
{
  byte[] pass = new UTF8Encoding().GetBytes(password + salt);
  byte[] hash = new SHA256Managed().ComputeHash(pass);
```

```
  StringBuilder passBuilder = new StringBuilder();
  for (int i = 0; i < hash.Length; i++) {
    passBuilder.Append(hash[i].ToString("x2"));
  }
  return passBuilder.ToString();
}
```

Nově spočtený hash je v databázové tabulce *Users* (viz tabulka č. 6) aktualizován:

```
UPDATE [Users]
SET [password] = @password
WHERE [id] = @id
```

Parametr *password* v tomto SQL dotazu je nový hash hesla a *id* identifikační číslo uživatele. Zobrazované jméno přihlášeného uživatele v pravém horním rohu webové stránky se změní až při opětovném přihlášení.

4.2.5 Vytvoření stavebního deníku

Stránka *new-diary.aspx* (viz příloha č. 4) slouží k vytváření nových stavebních deníků. Na tuto stránku se mohou dostat pouze přihlášení uživatelé. Formulář pro vytvoření nového stavebního deníku obsahuje několik kategorií: identifikační údaje stavby, zúčastněné strany, údaje o projektové a ostatní technické dokumentaci a nakonec seznam dokumentů a dokladů ke stavbě.

V tomto formuláři je možné přidávat některé položky podle potřeby. Uživatel si tak může vyplnit potřebný počet zhotovitelů, poddodavatelů a seznam dokumentů a dokladů ke stavbě.

Počet přidaných položek formuláře je uložen v QueryString. Sestavování formuláře se odehrává na straně serveru. Aby se uchoval stav i dynamicky přidaných položek formuláře, jsou vytvářeny v události stránky *Page_Init*.

Nejdříve se v *Page_Init* vygenerují formulářová tlačítka:

```
addContractor = new Button();
addContractor.Text = "Přidat dalšího zhotovitele";
addContractor.EnableViewState = false;

addSubcontractor = new Button();
addSubcontractor.Text = "Přidat dalšího poddodavatele";
addSubcontractor.EnableViewState = false;
```

```
addDocument = new Button();
addDocument.Text = "Přidat dalšího dokument nebo doklad";
addDocument.EnableViewState = false;
```

U těchto ovládacích prvků je nezbytné explicitně zakázat ViewState, jinak by nastala chyba při generování této stránky. Jejich stav, na rozdíl od ostatních formulářových prvků, si uchovávat nepotřebujeme.

Dále se z QueryString přečte, kolik je uživatelem přidaných položek:

```
int.TryParse(Request.QueryString["addedContractors"], out
addedContractors);
int.TryParse(Request.QueryString["addedSubcontractors"], out
addedSubcontractors);
int.TryParse(Request.QueryString["addedDocuments"], out
addedDocuments);
```

Následuje zpracování události *Page_Load*, ve které se nastaví přidávacím tlačítkům *PostBackUrl*. Uvedu příklad pro tlačítko, které slouží k přidání zhotovitele:

```
addContractor.PostBackUrl =
$"?addedContractors={addedContractors+1}&addedSubcontractors={addedS
ubcontractors}&addedDocuments={addedDocuments}";
```

Když uživatel klikne na toto tlačítko, odešle požadavek na server s QueryString, ve kterém se nastaví příslušný počet přidaných formulářových polí. Webová stránka je znovu načtena ze serveru, formulář sestaven se všemi přidanými položkami a vyplněná textová pole z předchozího požadavku jsou pomocí ViewState zachována.

Při odesílání vyplněného formuláře je nejprve provedena kontrola, zda jsou vyplněny jeho všechny povinné části. To jsou: název stavby, kraj, obec, katastrální území, v případě vyplnění stavbyvedoucího nebo stavebního dozoru jejich rozsah oprávnění a odpovědnosti, údaje o projektové a ostatní technické dokumentaci stavby, alespoň jeden dokument nebo doklad ke stavbě. Pokud je vše v pořádku, je nový stavební deník uložen do databáze.

4.2.6 Seznam stavebních deníků

Seznam uživatelových stavebních deníků na stránce *diaries.aspx* (viz příloha č. 5) se dělí na dvě části: aktivní stavební deníky a uzavřené stavební deníky. Aktivní stavební deníky jsou ty deníky, které nebyly uzavřeny. Lze je upravovat a vkládat do nich

informace. Uzavřené stavební deníky nelze upravit a je možné do nich pouze nahlížet a exportovat je.

Aktivní stavební deníky uživatele jsou z databáze získány pomocí následujícího SQL dotazu:

```
SELECT id, created, name, region, village, cadastral
FROM Diaries
WHERE closed = 0 AND owner = @owner
ORDER BY created DESC
```

Pro odlišení aktivních a uzavřených stavebních deníků je v tabulce *Diaries* (viz tabulka č. 1) sloupec *closed*. Výsledek je seřazen sestupně podle data vytvoření deníku. Aby se zobrazily pouze ty deníky, které si uživatel vytvořil, je v SQL dotazu parametr *owner*, který se vloží do dotazu *cmd* s následující hodnotou:

```
cmd.Parameters.AddWithValue("@owner", Extensions.UserID());
```

V třídě *Extensions*, kterou jsem si vytvořil pro pomocné funkce a metody, je funkce *UserID*, která vrací identifikační číslo přihlášeného uživatele:

```
public static int UserID() {
  return int.Parse(HttpContext.Current.User.Identity.Name
    .Split('|').Last());
}
```

4.2.7 Stavební deník

Na stránce *diary.aspx* (viz příloha č. 6) je zobrazen konkrétní stavební deník. Identifikační číslo deníku je přečteno z QueryString. Pro informace o deníku je v aplikaci vytvořena třída *Diary*:

```
public class Diary {
  public int ID { get; set; }
  public int OwnerID { get; set; }
  public DateTime Created { get; set; }
  public bool Closed { get; set; }
  public string Name { get; set; }
  public string Region { get; set; }
  public string Village { get; set; }
  public string Cadastral { get; set; }
  public string Contacts { get; set; }
}
```

Tato třída je pro data ze SQL tabulky *Diaries* (viz tabulka č. 1). V události stránky *Page_Load* je vytvořena instance této třídy:

```
pageDiary = new Diary();
```

Následně je ověřeno, zda má uživatel právo stavební deník zobrazit. Stavební deník smí zobrazit ten, kdo stavební deník vytvořil a osoby, kterým byl přidělen náhled do deníku. K ověření slouží pomocná funkce *PageDiaryAuthorization*, která je na této stránce volána:

```
pageDiary.ID = Extensions.DiaryPageAuthorization("id", out
isViewOnly, "/dashboard/diaries.aspx");
```

Funkce má tři parametry. V prvním se zadává název klíče v QueryString, ve kterém se nachází identifikační číslo deníku. Druhý parametr poukazuje na proměnnou, do které se zapíše hodnota *true*, pokud má uživatel právo do deníku pouze nahlížet a neprovádět řádné úpravy. V opačném případě se zapíše hodnota *false*. Třetí parametr obsahuje URL stránky, kam bude uživatel přesměrován, pokud nemá právo do deníku nahlížet. Kód této funkce je následující:

```
public static int DiaryPageAuthorization(string queryStringName, out
bool isViewOnly, string redirectUrl) {
  int diaryID;
  if (!int.TryParse(HttpContext.Current.Request.
  QueryString[queryStringName], out diaryID))
    HttpContext.Current.Response.Redirect(redirectUrl);
    using (var connection = new SqlConnection(connectionString)) {
      using (var cmd = new SqlCommand(@"
        SELECT [Diaries].[id], [Diaries].[owner],[Diaries].[closed],
        [Diary_views].[user_id]
        FROM [Diaries]
        LEFT JOIN [Diary_views]
        ON [Diaries].[id] = [Diary_views].[diary_id]
        WHERE [Diaries].[id] = @diary_id", connection)) {
        cmd.Parameters.AddWithValue("@diary_id", diaryID);
        connection.Open();
          using (var reader = cmd.ExecuteReader()) {
            if (reader.HasRows) {
              while (reader.Read()) {
                if ((int)reader["owner"] == UserID()) {
                  if ((bool)reader["closed"]) {
                    isViewOnly = true;
                    return diaryID;
                  } else {
                    isViewOnly = false;
                    return diaryID;
                  }
                } else {
                  if (reader["user_id"] != DBNull.Value) {
```

```
                    if ((int)reader["user_id"] == UserID()) {
                      isViewOnly = true;
                      return diaryID;
                    } else {
                      isViewOnly = true;
                      HttpContext.Current.Response
                      .Redirect(redirectUrl);
                    }
                  }
                }
              }
            } else {
              HttpContext.Current.Response.Redirect(redirectUrl);
            }
          }
        }
      }
  isViewOnly = true;
  HttpContext.Current.Response.Redirect(redirectUrl);
  return diaryID;
}
```

Na stránce je zobrazen kalendář se zvýrazněnými dny, které mají ve stavebním deníku alespoň jeden záznam.

Obrázek č. 7 Kalendář se zvýrazněnými dny

Při kliknutí na konkrétní den v kalendáři se zobrazí jemu příslušné záznamy. Pokud není vybrán žádný den, zobrazí se posledních deset. Seznam dní pro zvýraznění v kalendáři je vytvořen:

```
calendarDates = new List<DateTime>();
using (var connetion = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
    SELECT DISTINCT [datetime]
    FROM [Diary_records]
    WHERE [diary_id] = @diary_id", connetion)) {
    cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
    connetion.Open();
    using (var reader = cmd.ExecuteReader()) {
```

```
      if (reader.HasRows) {
        while (reader.Read()) {
          calendarDates.Add((DateTime)reader["datetime"]);
        }
      }
    }
  }
}
```

Ovládací prvek kalendáře má událost *OnDayRender*, ve které se nastaví zvýraznění dne:

```
protected void DiaryCalendar_DayRender(object sender,
DayRenderEventArgs e) {
  if (e.Day.Date == DiaryCalendar.SelectedDate.Date) return;
  foreach (var date in calendarDates) {
    if (date.Date == e.Day.Date) {
        e.Cell.BackColor = Color.FromArgb(194, 234, 255);
    }
  }
}
```

Nedílnou součástí této stránky jsou záznamy stavebního deníku. Pro načtení posledních deseti záznamů z databáze slouží SQL dotaz:

```
SELECT TOP 10 [Diary_records].[id] as [id], [datetime], [posted],
[text], [text_signed], [signature_name], [signature_key]
FROM [Diary_records]
INNER JOIN [Diaries]
ON [Diary_records].[diary_id] = [Diaries].[id]
WHERE [Diaries].[id] = @id
ORDER BY [Diary_records].[posted] DESC
```

Při výběru konkrétního dne v kalendáři se použije podobný SQL dotaz, který zohledňuje i vybrané datum:

```
SELECT [Diary_records].[id] as [id], [datetime], [posted], [text],
[text_signed], [signature_name], [signature_key]
FROM [Diary_records]
INNER JOIN [Diaries]
ON [Diary_records].[diary_id] = [Diaries].[id]
WHERE [Diaries].[id] = @id AND
CONVERT(DATE,[Diary_records].[datetime]) = @date
ORDER BY [Diary_records].[posted] DESC
```

Každý záznam ve stavebním deníku je digitálně podepsán a podpis se kontroluje. Pro kontrolu jsem si vytvořil funkci *CheckDigitalSignature*, která je v kódu použita:

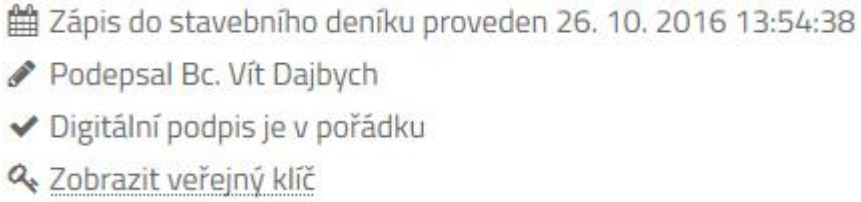
```
CheckDigitalSignature(reader["text"].ToString(),
reader["signature_key"].ToString(), reader["text_signed"] as byte[])
```

Funkce má tři vstupní parametry. První parametr je obsah zprávy, který bude kontrolován, druhý veřejný klíč podpisu a třetí hash dat (podpis). V uvedeném příkladu jsou vstupy do této funkce přímo z SQL tabulky *Diary_records* (viz tabulka č. 3). Teoretický postup kontroly elektronického podpisu byl vysvětlen v kapitole 1.4. Funkce má kód:

```
public string CheckDigitalSignature(string text, string
publicKeyXML, byte[] signedHash) {
  RSACryptoServiceProvider publicKeyProvider = new
  RSACryptoServiceProvider();
  publicKeyProvider.FromXmlString(publicKeyXML);

  string hashAlgorithm = "SHA1";
  var sha = new SHA1CryptoServiceProvider();
  byte[] data = Encoding.UTF8.GetBytes(text);
  var hash = sha.ComputeHash(data);

  RSAPKCS1SignatureDeformatter RSADeformatter = new
  RSAPKCS1SignatureDeformatter(publicKeyProvider);
  RSADeformatter.SetHashAlgorithm(hashAlgorithm);

  if (RSADeformatter.VerifySignature(hash, signedHash)) {
    return "<i class=\"fa fa-fw fa-check\"></i><span>
      Digitální podpis je v pořádku</span>";
  } else {
    return "<i class=\"fa fa-fw fa-times\"></i><span
      style=\"color:#bf2200;\">Digitální podpis není v
      pořádku</span>";
  }
}
```

Pokud by došlo k záměrné úpravě dat v databázi, zneplatnil by se tím digitální podpis a výstup funkce by byl formátovaný text „Digitální podpis není v pořádku“. Za každým záznamem ve stavebním deníku jsou uváděny informace o zapsání a elektronickém podpisu. To je demonstrováno na následujícím obrázku:

Obrázek č. 8 Příklad zápatí záznamu ve stavebním deníku



Na stránce jsou také uvedeny ostatní informace o stavebním deníku, včetně jejich změn a doplňků. Pro tyto informace jsem vytvořil třídu *DiaryDetail*:

```
public class DiaryDetail {
  public int ID { get; set; }
  public int DiaryID { get; set; }
  public bool Initial { get; set; }
  public bool Changed { get; set; }
  public DateTime DateTime { get; set; }
  public string Key { get; set; }
  public string Value { get; set; }
  public string Phone { get; set; }
  public string ValueExtended { get; set; }
  public string OldValue { get; set; }
  public string OldPhone { get; set; }
  public string OldValueExtended { get; set; }
  public DiaryDetail() { }
  public DiaryDetail(string key, string value, string
  value_extended, string phone) {
    Key = key;
    Value = value;
    ValueExtended = value_extended;
    Phone = phone;
  }
}
```

Tato třída slouží pro data z SQL tabulky *Diary_details* (viz tabulka č. 2). Všechny informace o stavebním deníku jsou načteny do seznamu pomocnou funkcí *LoadDiaryDetails*:

```
List<DiaryDetail> details = new List<DiaryDetail>();
details.LoadDiaryDetails(pageDiary.ID);
```

Jako vstupní parametr této funkce je identifikační číslo deníku, ze kterého chceme data získat. Kód funkce je:

```
public static List<DiaryDetail> LoadDiaryDetails(this
List<DiaryDetail> list, int diaryID) {
  using (var connetion = new SqlConnection(connectionString)) {
    using (var cmd = new SqlCommand(@"
    SELECT [id], [diary_id], [initial], [changed], [datetime],
    [key], [value], [phone], [value_extended], [old_value],
    [old_phone], [old_value_extended]
    FROM [Diary_details]
    WHERE [diary_id] = @diary_id", connetion)) {
      cmd.Parameters.AddWithValue("@diary_id", diaryID);
      connetion.Open();
      using (var reader = cmd.ExecuteReader()) {
        if (reader.HasRows) {
          while (reader.Read()) {
            DiaryDetail detail = new DiaryDetail();
            detail.ID = (int)reader["id"];
```

```
                detail.DiaryID = (int)reader["diary_id"];
                detail.Initial = (bool)reader["initial"];
                detail.Changed = (bool)reader["changed"];
                detail.DateTime = (DateTime)reader["datetime"];
                detail.Key = (string)reader["key"];
                detail.Value = (string)reader["Value"];
                detail.Phone = reader["phone"] == DBNull.Value ?
                  string.Empty : (string)reader["phone"];
                detail.ValueExtended = reader["value_extended"] ==
                  DBNull.Value ? string.Empty :
                  (string)reader["value_extended"];
                detail.OldValue = reader["old_value"] == DBNull.Value ?
                  string.Empty : (string)reader["old_value"];
                detail.OldPhone = reader["old_phone"] == DBNull.Value ?
                  string.Empty : (string)reader["old_phone"];
                detail.OldValueExtended = reader["old_value_extended"]
                  == DBNull.Value ? string.Empty :
                  (string)reader["old_value_extended"];
                list.Add(detail);
              }
            }
          }
        }
      }
      return list;
    }
```

Následuje zpracování načtených hodnot. Nejdříve jsou do HTML tabulky vypsáni účastníci výstavby, kteří jsou vybráni ze seznamu *details* pomocí cyklů:

```
foreach (var person in Extensions.Participants)
  foreach (var item in (from item in details where (item.Key ==
  person && item.Initial == true) select item))
```

Tyto dva cykly slouží k vytvoření tabulky s účastníky výstavby, kteří byli zapsáni při zakládání stavebního deníku. V seznamu *Extension.Participants* jsou všichni účastníci výstavby, kteří mohou ve stavebním deníku být uvedeni. Změny a doplňky jsou zobrazeny v jiné tabulce. Seznam změn a doplňků *participantsChanges* je definován LINQ dotazem:

```
var participantsChanges = (from item in details where
((Extensions.Participants.Contains(item.Key) ||
Extensions.Firms.Contains(item.Key)) && item.Initial == false)
orderby item.DateTime select item);
```

Další tabulka na této stránce obsahuje údaje o projektové a ostatní technické dokumentaci stavby. Tyto údaje *techDocInital* a jejich změny *techDocChanges* jsou definovány LINQ dotazy:

```
var techDocInitial = ((from item in details where (item.Key ==
"Dokumentace"
&& item.Initial == true) select item));
var techDocChanges = ((from item in details where (item.Key ==
"Dokumentace" && item.Initial == false) orderby item.DateTime
select item));
```

Nakonec je umístěna tabulka se seznamem dokumentů a dokladů ke stavbě a jejich doplňků. Seznamy dokumentů *documents* a doplňků *documentAddons* jsou definovány LINQ dotazy:

```
var documents = (from item in details where (item.Key == "Dokument"
&& item.Initial == true) select item);
var documentAddons = (from item in details where (item.Key ==
"Dokument" && item.Initial == false) orderby item.DateTime select
item);
```

Do stavebního deníku je možné zapisovat změny. Původní informace zůstávají zachovány a změna se uvádí v příslušné tabulce (viz obrázek č. 9). Každou položku, která připouští změny, lze změnit pouze jednou. Lze ovšem upravovat změny a doplňky.

Obrázek č. 9 Tabulka změn a doplňků

Datum	Strana	Půdovní	Nový	
15. 12. 2016 11:23:34	Technický dozor stavebníka (investora)	**Jméno a příjmení** REALSTAV MB, spol. s r. o. **Telefon** *Nevyplněno.*	**Jméno a příjmení** Ing. Milan Novák **Telefon** 723 468 552	Změnit
15. 12. 2016 11:28:25	Poddodavatel		**Obchodní firma, místo podnikání nebo sídlo** REMIRENT ČR s. r. o. Před Nádražím 68 140 00 Praha 4	Změnit

4.2.8 Úprava dat stavebního deníku

Stránka *edit-diary.aspx* (viz příloha č. 7) umožňuje vkládání změn a doplňků do stavebního deníku. Protože lze upravovat různé druhy položek obsahující jedno až tři textové pole, je na této stránce definován výčet hodnot *EditType*:

```
private enum EditType {
    Firm,
    Participant,
    ParticipantExtended,
    TechnicalDocumentation
}
```

V události stránky *Page_Load* je ověřeno, jestli úpravu chce provádět vlastník stavebního deníku a také jestli se nejedná o uzavřený stavební deník. Pokud kontrola vyhoví, je z databáze načtena položka, kterou uživatel vybral k úpravě:

```
currentDetail = new DiaryDetail();
using (var connetion = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
  SELECT [key], [value], [phone], [value_extended]
  FROM [Diary_details]
  WHERE [diary_id] = @diary_id AND [id] = @item_id AND [changed] =
  0", connetion)) {
    cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
    cmd.Parameters.AddWithValue("@item_id", editItemID);
    connetion.Open();
    using (var reader = cmd.ExecuteReader()) {
      if (reader.HasRows) {
        while (reader.Read()) {
          currentDetail.Key = (string)reader["key"];
          currentDetail.Value = (string)reader["Value"];
          currentDetail.Phone = reader["phone"] == DBNull.Value ?
            string.Empty : (string)reader["phone"];
          currentDetail.ValueExtended = reader["value_extended"] ==
            DBNull.Value ? string.Empty :
            (string)reader["value_extended"];
        }
      } else {
        Response.Redirect("/dashboard/diaries.aspx");
      }
    }
  }
}
```

Následně se zjistí, o jaký druh upravované položky se jedná a vygenerují se příslušné ovládací prvky pro úpravu. Pro jednotlivé druhy to jsou konktrétně:

- *Firm*
  - Obchodní firma, místo podnikání nebo sídlo
- *ParticipansExtended*
  - Jméno a příjmení
  - Rozsah oprávnění a odpovědnosti
  - Telefon

- *Participant*
  - Jméno a příjmení
  - Telefon
- *TechnicalDocumentation*
  - Údaje o projektové a ostatní technické dokumentaci stavby

Pro tvorbu těch prvků jsem vytvořil pomocnou funkci *AddTableInput*, která do tabulky na stránce přidá textové pole s příslušným *placeholder*:

```
private void AddTableInput(string value, string placeholder,
TextBoxMode mode, string ID) {
  TableRow row = new TableRow();
  TableCell cell = new TableCell();
  TextBox textBox = new TextBox();
  textBox.Text = value;
  textBox.ID = ID;
  textBox.AutoCompleteType = AutoCompleteType.None;
  textBox.TextMode = mode;
  textBox.Attributes.Add("placeholder", placeholder);
  if (mode == TextBoxMode.MultiLine) {
      textBox.Rows = 6;
  } else {
      textBox.Attributes.Add("onkeydown", "return
        (event.keyCode!=13);");
  }
  cell.Controls.Add(textBox);
  row.Cells.Add(cell);
  EditTable.Rows.Add(row);
}
```

Webové formuláře v prohlížečích lze v některých případech odeslat stisknutím klávesy Enter. Tento formulář pro úpravu může obsahovat jednořádkové i víceřádkové textové pole, ve kterém se klávesa Enter používá. Neúmyslnému odeslání formuláře je zabráněno pomocí JavaScript. Jednořádková textová pole mají definovanou událost *onkeydown*, která v případě stisknutí klávesy Enter (kód klávesy 13) vrátí hodnotu *true* a formulář se neodešle.

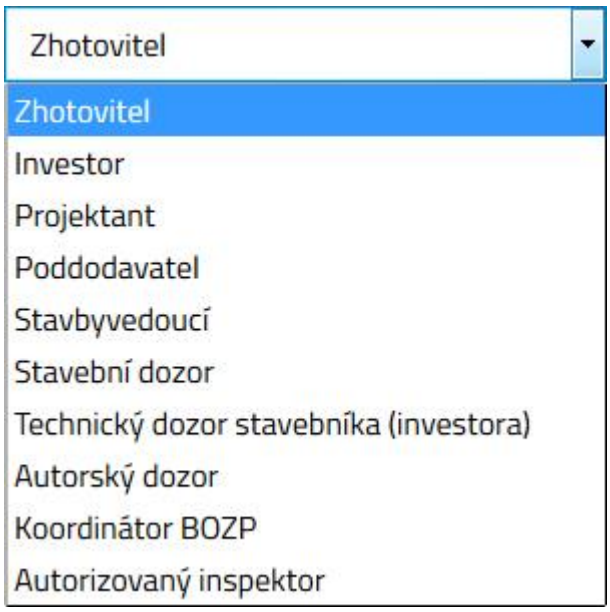
Po odeslání formuláře se v SQL tabulce *Diary_details* (viz tabulka č. 2) vytvoří nový řádek, který obsahuje text před úpravou položky a po úpravě. Takový řádek má nastavenou hodnotu sloupce *initial* rovnu *0*, jelikož se nejedná o řádek vložení při vytváření stavebního deníku. Také se vkládá s hodnotou sloupce *changed* roven *0*. Původní záznam v tabulce se nemění, pouze se nastaví hodnota sloupce *changed* na hodnotu *1*.

4.2.9 Přidání osob do stavebního deníku

Stránka *add-participant.aspx* (viz příloha č. 8) umožnuje dodatečně přidávat další osoby do stavebního deníku. Jelikož nejsou definovány při zakládání deníku, jsou uvedeny v části „Změny a doplňky“. Stejně jako na stránce pro úpravu dat stavebního deníku (viz kapitola 4.2.8) je i zde využit výčet hodnot *EditType*. Při načítání stránky se ověří vlastnictví stavebního deníku a také se ověří, zda se nejedná o uzavřený stavební deník. Následuje načtení informací o stavebním deníku z databáze.

Na stránce je vytvořen výběr osob (HTML tag *select*), které je možné do stavebního deníku přidat. Výsledek je vidět na následujícím obrázku:

Obrázek č. 10 Seznam osob pro přidání do stavebního deníku



Podle konkrétního výběru je pak sestaven formulář v pomocné funkci *CreateTable*. Ta je zavolána jednak v události stránky *Page_Load* a také v události výběrového prvku:

```
protected void List_SelectedIndexChanged(object sender, EventArgs e)
{
  CreateTable(((DropDownList)sender).SelectedItem.Text);
}
```

Funkce *CreateTable*, sestavující výsledný formulář, má kód:

```
private void CreateTable(string selectedParticipant) {
  AddTable.Rows.Clear();
  if (Extensions.Firms.Contains(selectedParticipant)) {
    mode = EditType.Firm;
```

```
    AddTableInput(string.Empty, "Obchodní firma, místo podnikání
nebo sídlo", TextBoxMode.MultiLine, "Item_0");
  } else if (Extensions.Participants.Contains(selectedParticipant))
{
    if (Extensions.ParticipantsExtended
        .Contains(selectedParticipant)) {
      mode = EditType.ParticipantExtended;
      AddTableInput(string.Empty, "Jméno a příjmení",
        TextBoxMode.SingleLine, "Item_0");
      AddTableInput(string.Empty, "Rozsah oprávnění a odpovědnosti",
        TextBoxMode.MultiLine, "Item_1");
      AddTableInput(string.Empty, "Telefon", TextBoxMode.SingleLine,
        "Item_2");
    } else {
      mode = EditType.Participant;
      AddTableInput(string.Empty, "Jméno a příjmení",
        TextBoxMode.SingleLine, "Item_0");
      AddTableInput(string.Empty, "Telefon", TextBoxMode.SingleLine,
        "Item_1");
    }
  }
}
```

Vstupním parametrem *selectedParticipant* je role osoby, která bude přidána. Funkce sestaví tabulku podle typu *EditType* vybrané osoby *selectedParticipant*. Pro některé je možnost vyplnit jméno a příjmení, telefonní číslo a rozsah oprávnění a odpovědnosti, pro jiné pouze jedno textové pole pro vyplnění obchodní firmy, místa podnikání nebo sídla. Kód pomocné funkce *AddTableInput* v uvedeném příkladu je popsán v kapitole 4.2.9.

Po odeslání formuláře se provede kontrola, zda byly vyplněny všechny požadované položky:

```
Control c = AddForm.FindControl("Item_0");
TextBox t = (TextBox)c;
if (string.IsNullOrWhiteSpace(t.Text)) {
  Message.Text = "Nejsou vyplněny požadované položky.";
  Message.Type = Stavebni_Denik.Controls.Message.TypeOption.Alert;
  return;
}
if (!Extensions.Participants.Contains(List.SelectedItem.Text.Trim())
  && !Extensions.Firms.Contains(List.SelectedItem.Text.Trim())) {
  Message.Text = "Neznámá osoba!";
  Message.Type = Stavebni_Denik.Controls.Message.TypeOption.Alert;
  return;
}
string newValue = t.Text.Trim();
string newPhone = null;
string newValueExtended = null;
switch (mode) {
  case EditType.Participant:
    c = AddForm.FindControl("Item_1");
```

```
      t = (TextBox)c;
      newPhone = t.Text.Trim();
      break;
    case EditType.ParticipantExtended:
      c = AddForm.FindControl("Item_1");
      t = (TextBox)c;
      newValueExtended = t.Text.Trim();
      if (string.IsNullOrWhiteSpace(t.Text)) {
        Message.Text = "Nejsou vyplněny požadované položky.";
        Message.Type = Stavebni_Denik.Controls.Message
          .TypeOption.Alert;
        return;
      }
      c = AddForm.FindControl("Item_2");
      t = (TextBox)c;
      newPhone = t.Text.Trim();
      break;
}
```

Pokud je vše v pořádku, je záznam uložen do databázové tabulky *Diary_details* (viz tabulka č. 2). Nový řádek tabulky je vkládán s hodnotami sloupců *initial* rovno *0* a *changed* rovno *0*, jelikož se nejedná o údaje definované při zakládání stavebního deníku a údaje zatím nejsou změněné.

4.2.10 Vložení záznamu do stavebního deníku

Stránka *diary-record.aspx* (viz příloha č. 9) slouží k zapisování záznamů do stavebního deníku. Uživatel zde vyplňuje záznam o stavbě, datum a posílá svůj kvalifikovaný certifikát včetně hesla k privátnímu klíči.

Při odeslání na webový server se záznam zkontroluje. Záznam se nesmí vložit před již existující záznam a zároveň nelze vložit záznam s budoucím datem. Zároveň je porovnáno datum platnosti kvalifikovaného certifikátu s vybraným datem. Díky tomu, že kontrola je prováděna na vzdáleném serveru a nikoli přímo ve vlastním počítači, nemůže uživatel změnou svého času kontrolu dat ovlivnit. Pokud je vše v pořádku a uživatel zadal správné heslo ke svému privátnímu klíči, vytvoří se elektronický podpis a informace jsou uloženy do databáze.

Při načítání stránky *diary-records.aspx* se nejdříve v události stránky *Page_Load* ověří, zda má uživatel právo do stavebního deníku zapisovat. Identifikační číslo deníku je získáváno z QueryString. Nejdříve je vytvořena nová instance veřejné proměnné *Diary* a následuje funkce, která ověřuje oprávnění zápisu:

```
pageDiary = new Diary();
pageDiary.ID = Extensions.DiaryOwnerAuthorization("id",
"/dashboard/diaries.aspx");
```

Také se musí ověřit, zda se nejedná o uzavřený stavební deník, do kterého již nelze žádné záznamy vkládat:

```
pageDiary.ID.DiaryNotClosedAuthorization("/dashboard/diaries.aspx");
```

Pokud by uživatel neměl příslušnou autorizaci, bude okamžitě přesměrován na stránku se svými stavebními deníky. Má-li však autorizaci k zápisu, nahrají se z databáze informace o stavebním deníku pomocnou funkcí *LoadDiaryInfo*:

```
pageDiary.LoadDiaryInfo();
```

Na stránce máme také definovanou proměnnou pro datum posledního záznamu v databázi:

```
public DateTime? lastDiaryRecord = null;
```

Pokud se jedná o nový stavební deník, v databázi se nenachází zatím žádný záznam a tato proměnná proto nemusí obsahovat hodnotu. Samotné nalezení data zajišťuje kód:

```
using (var connetion = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
  SELECT TOP 1 [Diary_records].[datetime]
  FROM [Diary_records]
  INNER JOIN [Diaries]
  ON [Diary_records].[diary_id] = [Diaries].[id]
  WHERE [Diaries].[id] = @id
  ORDER BY [Diary_records].[datetime] DESC", connetion)) {
    cmd.Parameters.AddWithValue("@id", pageDiary.ID);
    connetion.Open();
    using (var reader = cmd.ExecuteReader()) {
      if (reader.HasRows) {
        while (reader.Read()) {
          lastDiaryRecord = (DateTime)reader["datetime"];
        }
      }
    }
  }
}
```

Když některý záznam v databázi existuje, nesmí se před něj vložit jiný. Uživatel je na stránce informován o datu posledního záznamu ve stavebním deníku:

```
<%=lastDiaryRecord.HasValue ? "<p>Poslední záznam v deníku ze dne
<strong>" + lastDiaryRecord.Value.ToLongDateString() +
"</strong></p>" : string.Empty %>
```

Při kliknutí na tlačítko „Zapsat do stavebního deníku“ jsou vyplněné informace odeslány na webový server. Nejdříve se kontroluje, zda byl vyplněn záznam o stavbě, vybráno datum v kalendáři, kvalifikovaný certifikát a zadáno heslo k privátnímu klíči. Pokud ano, kontroluje se, jestli nebylo vybráno budoucí datum a následuje porovnání vybraného data s datem posledního záznamu v databázi. Je-li vše v pořádku, následuje práce s kvalifikovaným certifikátem:

```
byte[] fileData = null;
using (var br = new BinaryReader(DiaryCertificate
    .PostedFile.InputStream)) {
  fileData = br.ReadBytes(DiaryCertificate
    .PostedFile.ContentLength);
}

X509Certificate2 cert = new X509Certificate2();
try {
  cert.Import(fileData, DiaryCertificatePass.Text,
    X509KeyStorageFlags.PersistKeySet);
} catch(CryptographicException ex) {
  Message.Text = $"Nepodařilo se načíst váš certifikát.
    <em>{ex.Message}</em>";
  Message.Type = Stavebni_Denik.Controls.Message.TypeOption.Alert;
  return;
}
```

Do proměnné *fileData* je nahrán obsah nahraného souboru. Vytvoří se nová proměnná *cert* typu *X509Certificate2* pro práci s kvalifikovaným certifikátem. Uživatel ale mohl odeslat špatný soubor. Proto se nejdříve pokusíme zachytit výjimku při importování dat kvalifikovaného certifikátu. Nezdaří-li se, vypíše se chybová hláška „Nepodařilo se načíst certifikát“, v opačném případě kód pokračuje.

Je zapotřebí zkontrolovat, zda kvalifikovaný certifikát obsahuje soukromý (privátní) klíč, který se nezbytný k elektronickému podpisu:

```
if (!cert.HasPrivateKey) {
  Message.Text = "Certifikát neobsahuje soukromý klíč.";
  Message.Type = Stavebni_Denik.Controls.Message.TypeOption.Alert;
  return;
}
```

Dále se porovná datum platnosti kvalifikovaného certifikátu se současným datem při odesílání záznamu:

```
if (DateTime.Now < cert.NotBefore) {
  Message.Text = $"Cerfitikát je platný od
    {cert.NotBefore.ToString()}.";
  Message.Type = Stavebni_Denik.Controls.Message.TypeOption.Alert;
  return;
}
if (DateTime.Now > cert.NotAfter) {
  Message.Text = $"Certifikát je platný do
    {cert.NotAfter.ToString()}.";
  Message.Type = Stavebni_Denik.Controls.Message.TypeOption.Alert;
  return;
}
```

Toto byla poslední kontrola. Nyní se vytvoří elektronický podpis záznamu soukromým klíčem a také si uchováme veřejný klíč pro pozdější ověřování podpisu:

```
RSACryptoServiceProvider privateKeyProvider = cert.PrivateKey as
  RSACryptoServiceProvider;
RSACryptoServiceProvider publicKeyProvider = cert.PublicKey.Key as
  RSACryptoServiceProvider;
string hashAlgorithm = "SHA1";
byte[] data = Encoding.UTF8.GetBytes(DiaryRecord.Text);
var sha = new SHA1CryptoServiceProvider();
var hash = sha.ComputeHash(data);
RSAPKCS1SignatureFormatter RSAFormatter = new
  RSAPKCS1SignatureFormatter(privateKeyProvider);
RSAFormatter.SetHashAlgorithm(hashAlgorithm);
byte[] signedHash = RSAFormatter.CreateSignature(hash);
string publicKeyXML = publicKeyProvider.ToXmlString(false);
string signer = cert.GetNameInfo(X509NameType.SimpleName, false);
```

V tomto kódu jsme spočítali hash záznamu o stavbě algoritmem SHA-1, který je v proměnné *hash*. Poté je tento hash podepsán soukromým klíčem a v proměnné *signedHash* se vytvoří podpis. Dále do proměnné *publicKeyXML* načteme z kvalifikovaného certifikátu veřejný klíč ve formátu XML a do proměnné *signer* jméno podepisujícího. Tyto získané údaje uložíme do databázové tabulky *Diary_records* (viz tabulka č. 3):

```
using (var connetion = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
    INSERT INTO [Diary_records]
        ([diary_id]
        ,[datetime]
        ,[posted]
        ,[text]
        ,[text_signed]
        ,[signature_name]
        ,[signature_key])
    VALUES
        (@diary_id
        ,@datetime
```

```
        ,@posted
        ,@text
        ,@text_signed
        ,@signature_name
        ,@signature_key)
    ", connetion)) {
    cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
    cmd.Parameters.AddWithValue("@datetime",
      DiaryDate.SelectedDate);
    cmd.Parameters.AddWithValue("@posted", DateTime.Now);
    cmd.Parameters.AddWithValue("@text", DiaryRecord.Text);
    cmd.Parameters.AddWithValue("@text_signed", signedHash);
    cmd.Parameters.AddWithValue("@signature_name", signer);
    cmd.Parameters.AddWithValue("@signature_key", publicKeyXML);
    connetion.Open();
    cmd.ExecuteNonQuery();
  }
}
```

4.2.11 Náhledy do stavebního deníku

Stránka *diary-views.aspx* (viz příloha č. 10) zobrazuje uživatele, kteří mají náhled do stavebního deníku. V události stránky *Page_Load* je ověřeno, jestli tuto stránku zobrazuje vlastník stavebního deníku a jestli se nejedná o uzavřený stavební deník. Po této kontrole se provádí případné odebrání uživatele ze seznamu náhledů do deníku. Pokud je stránka načítána s příslušným QueryString, provede se smazání uživatele z SQL tabulky a stránka se opětovně načte:

```
if (Request.QueryString["user"] != null) {
  int userID;
  if (!int.TryParse(HttpContext.Current.Request
    .QueryString["user"], out userID)) Response
    .Redirect("/dashboard/diaries.aspx");
  using (var connetion = new SqlConnection(connectionString)) {
    using (var cmd = new SqlCommand(@"
      DELETE FROM [Diary_views]
      WHERE [user_id] = @user_id
      AND [diary_id] = @diary_id", connetion)) {
      cmd.Parameters.AddWithValue("@user_id", userID);
      cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
      connetion.Open();
      cmd.ExecuteNonQuery();
      Response.Redirect($"/dashboard/diary-
        views.aspx?id={pageDiary.ID}");
    }
  }
}
```

Informace o náhledech do stavebních deníků jsou uchovávány v SQL tabulce *Diary_views* (viz tabulka č. 5). Seznam uživatelů s náhledem do konkrétního deníku je definován jednoduchým SQL dotazem:

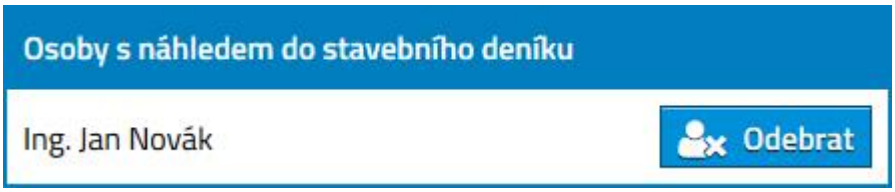
```
SELECT [user_id]
FROM [Diary_views]
WHERE [diary_id] = @diary_id
```

Vygeneruje se HTML tabulka, ve které je uvedeno jméno a tlačítko s možností uživateli odebrat náhled do stavebního deníku. Jméno jsme z identifikačního čísla uživatele *user_id* získali pomocí pomocné funkce *UserName*, která je použita v kódu stránky následovně:

```
((int)reader["user_id"]).UserName()
```

Podoba výsledné HTML tabulky se seznamem uživatelů s náhledem do stavebního deníku je znázorněna na obrázku:

Obrázek č. 11 Seznam uživatelů s náhledem do stavebního deníku



Při kliknutí na tlačítko „Odebrat“ je stránka znovu načtena a identifikační číslo uživatele, který má být odebrán, je předáno v QueryString. Relativní URL stránky s odebíráním může mít podobu:

```
/dashboard/diary-views.aspx?id=4&user=7
```

V události stránky *Page_Load* popsané na začátku této kapitoly se pak provede smazání řádku v SQL tabulce. Pro přidávání náhledů do stavebního deníku slouží stránka *diary-views-add.aspx*, na kterou je odkazováno tlačítkem „Přidat možnost náhledu“.

4.2.12 Přidání možnosti náhledu do stavebního deníku

Stránka *diary-views-add.aspx* (viz příloha č. 11) umožňuje vlastníkovi stavebního deníku přidělit náhled do svého deníku jinému uživateli. Na stránce je textové pole pro vyhledávání osoby, které chceme náhled přidělit. Do vyhledávání jsou zahrnuti všichni

zaregistrování uživatelé. Z výsledků hledání je odebrán majitel stavebního deníku a uživatelé, kteří již náhled mají. V kódu stránky je definován seznam *diaryViews*, který bude obsahovat identifikační čísla odebraných uživatelů:

```
private List<int> diaryViews;
```

V události stránky *Page_Load* je ověřeno, zda stránku zobrazuje majitel stavebního deníku a jestli se nejedná o neuzavřený stavební deník. Následně jsou do seznamu *diaryViews* přidáni všichni uživatelé, kteří mají právo náhledu do stavebního deníku:

```
using (var connetion = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
    SELECT [user_id]
    FROM [Diary_views]
    WHERE [diary_id] = @diary_id", connetion)) {
    cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
    connetion.Open();
    using (var reader = cmd.ExecuteReader()) {
      if (reader.HasRows) {
        while (reader.Read()) {
          diaryViews.Add((int)reader["user_id"]);
        }
      }
    }
  }
}
```

Je-li stránka načítána po výběru uživatele, kterému chce náhled přidat, pak QueryString obsahuje identifikační číslo tohoto uživatele. Relativní URL může mít podobu:

```
/dashboard/diary-views-add.aspx?id=4&user=7
```

Tato URL obsahuje nenulovou hodnotu proměnné *user*, tudíž se provede samotné přidání uživatele do databázové tabulky *Diary_views* (viz tabulka č. 5). Nejdříve se ovšem zkontroluje: zda je *user* číslo, zda poukazuje na konkrétního uživatele v databázi a zda se nejedná o vlastníka stavebního deníku či uživatele, který již náhled do deníku má:

```
if (Request.QueryString["user"] != null) {
  int userID;
  if (!int.TryParse(HttpContext.Current.Request
    .QueryString["user"], out userID)) Response
    .Redirect("/dashboard/diaries.aspx");
  if (!userID.UserExists()) Response
    .Redirect("/dashboard/diaries.aspx");
```

```
  if (userID == Extensions.UserID()) Response
    .Redirect("/dashboard/diaries.aspx");
  if (diaryViews.Contains(userID)) Response
    .Redirect("/dashboard/diaries.aspx");
  using (var connetion = new SqlConnection(connectionString)) {
    using (var cmd = new SqlCommand(@"
    INSERT INTO [Diary_views]
        ([user_id]
        ,[diary_id])
    VALUES
        (@user_id
        ,@diary_id)", connetion)) {
        cmd.Parameters.AddWithValue("@user_id", userID);
        cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
        connetion.Open();
        cmd.ExecuteNonQuery();
        Response.Redirect($"/dashboard/diary-
          views.aspx?id={pageDiary.ID}");
    }
  }
}
```

Nevyhoví-li některá z podmínek, je uživatel přesměrován z této stránky na seznam svých stavebních deníků. Při úspěšném přidání uživatele do databáze je stránka znovu načtena bez proměnné *user* v QueryString.

Stránka obsahuje textové pole *SearchQuery* na zadávání vyhledávacího výrazu a tlačítko *SearchSubmit* pro jeho odeslání. Událost tlačítka *SearchSubmit_Click* obsahuje kód pro vyhledávání osob v databázi a zobrazení výsledků. Nejdříve je ověřeno, zda vyhledávací výraz není prázdný nebo obsahuje pouze bílé znaky[4]:

```
if (string.IsNullOrWhiteSpace(SearchQuery.Text)) return;
```

SQL dotaz pro vyhledání osob v databázi je následující:

```
SELECT [id], [first_name], [last_name], [degree_prefix],
[degree_append]
FROM [users]
WHERE [id] != @user_id
AND ([last_name] LIKE ('%'+@query+'%')
    OR [first_name] LIKE ('%'+@query+'%')
    OR ([first_name] + ' ' + [last_name]) LIKE ('%'+@query+'%')
    OR ([last_name] + ' ' + [first_name]) LIKE ('%'+@query+'%')
)
```

Dovolím si myšlenku toho SQL dotazu popsat. Chtěl jsem, aby byla osoba ve výsledcích vyhledávání, pokud se zadá pouze příjmení nebo pouze jméno. Zároveň je možné osobu vyhledat podle jména včetně příjmení, přičemž nenáleží na pořadí

[4] Například mezera, tabulátor nebo nový řádek.

(například „Jan Novák“ či „Novák Jan“). Výsledky hledání, jsou-li nějaké, se vypíší do HTML tabulky:

```
TableRow headerRow = new TableRow();
headerRow.Cells.Add(new TableHeaderCell() { Text = "Výsledek
  hledání" });
headerRow.Cells.Add(new TableHeaderCell() { Text = "" });
SearchResults.Rows.Add(headerRow);
while (reader.Read()) {
  SearchLabel.Visible = false;
  TableRow row = new TableRow();
  TableCell cellName = new TableCell();
  StringBuilder userName = new StringBuilder();
  if (reader["degree_prefix"] != DBNull.Value) userName
    .Append($"{reader["degree_prefix"].ToString()} ");
  userName.Append($"{reader["first_name"]
    .ToString()} {reader["last_name"].ToString()}");
  if (reader["degree_append"] != DBNull.Value) userName
    .Append($", {reader["degree_append"].ToString()}");
  cellName.Text = userName.ToString();
  TableCell cellButton = new TableCell();
  if (!diaryViews.Contains((int)reader["id"])) {
    cellButton.Text = $"<a class=\"button\" href=\"/dashboard/diary-
    views-add.aspx?id={pageDiary.ID}&user={reader["id"]
    .ToString()}\"><i class=\"fa fa-user-plus fa-
    lg\"></i>Přidat</a>";
  } else {
    cellButton.Text = "<em>má náhled do deníku</em>";
  }
row.Cells.Add(cellName);
row.Cells.Add(cellButton);
SearchResults.Rows.Add(row);
```

Výslednou podobu této tabulky je možné vidět na obrázku:

Obrázek č. 12 Výsledek hledání

Kliknutím na tlačítko „Přidat“ se odešle požadavek na server s proměnnou *user* v URL a osoba je přidána do databáze. Tento postup jsem popsal na začátku této kapitoly. Po úspěšném přidání je uživatel přesměrován na stránku *diary-views.aspx* se sezamem osob s náhledem do stavebního deníku.

4.2.13 Autorizační razítka

Stránka *diary-stamps.aspx* (viz příloha č. 12) umožňuje ke stavebnímu deníku nahrát obrázky. Některé osoby musí své oprávnění prokázat otisknutím svého autorizačního razítka. Proto jsem pro elektronický stavební deník naprogramoval možnost nahrávat do SQL tabulky *Diary_stamps* (viz tabulka č. 4) obrázky. Je tedy možné k deníku přiložit otisk autorizační razítka. Aplikace akceptuje formáty obrázků JPEG, PNG a BMP. Maximální velikost nahrávaného obrázku nesmí být větší než 1 MB.

Událost stránky *Page_Load* ověří totožnost uživatele. Vlastník stavebního deníku smí na této stránce obrázky nahrávat a mazat. Osoby s náhledem do stavebního deníku nejsou z této stránky přesměrováni. Mají možnost si nahrané obrázky zobrazovat. Ostatním uživatelům je přístup na stránku zamítnut.

Autorizace na této stránce funguje stejně jako na stránce se stavebním deníkem. Pomocí pomocné funkce *DiaryPageAuthorization* (odtud také název funkce) je nastavena hodnota veřejné proměnné *IsViewOnly*:

```
pageDiary.ID = Extensions.DiaryPageAuthorization("id", out
isViewOnly, "/dashboard/diaries.aspx");
```

Funkce vrací identifikační číslo deníku, pokud byla autorizace provedena v pořádku. Jinak je uživatel ihned přesměrován z této stránky. Má-li uživatel právo pouze nahlížet, je hodnota proměnné *isViewOnly* nastavena na *true*, v opačném případě na hodnotu *false*.

Pokud je stránka nahrávána po kliknutí na tlačítko „Odstranit obrázek“, obsahuje QueryString proměnnou *delete* s identifikačním číslem obrázku, který chce uživatel odstranit. V *Page_Load* se tedy provede před načtením stránky smazání obrázku z SQL tabulky *Diary_stamps* (viz tabulka č. 4). Kód pro odstranění obrázku z databáze je následující:

```
if (Request.QueryString["delete"] != null) {
  if (isViewOnly) Response.Redirect("/dashboard/diaries.aspx");
  int deleteID;
  if (!int.TryParse(HttpContext.Current.Request
    .QueryString["delete"], out deleteID)) Response
    .Redirect("/dashboard/diaries.aspx");
  Extensions.DiaryOwnerAuthorization("id",
    "/dashboard/diaries.aspx");
  using (var connetion = new SqlConnection(connectionString)) {
    using (var cmd = new SqlCommand(@"
      DELETE FROM [Diary_stamps]
```

```
      WHERE [id] = @id", connetion)) {
        cmd.Parameters.AddWithValue("@id", deleteID);
        connetion.Open();
        cmd.ExecuteNonQuery();
        Response.Redirect($"/dashboard/diary-
          stamps.aspx?id={pageDiary.ID}");
    }
  }
}
```

Aby nebyly z formuláře pro nahrávání obrázku zbytečně odesílány soubory na webový server, je kontrola souboru provedena nejdříve na straně klienta. Pokud by tomu tak nebylo a uživatel by vybral pro nahrání soubor o velikosti například 500 MB, data by se musela odeslat na webový server, kde by se teprve provedla kontrola. Odesílání by zabralo zbytečný čas, protože by server data ihned odmítl. Proto je na stránce JavaScript kód, který zkontroluje příponu a velikost obrázku před odesláním formuláře. Pokud je soubor nevyhovující, pak odeslání dat zabrání.

Pro nahrání souborů obsahuje webová aplikace vlastní ovládací prvek *denik:file*. Formulář pro odesílání obrázků má následující kód:

```
<form id="UploadForm" runat="server">
  <div class="bpadding">
  <denik:file ID="UploadFile" Accept=".jpg,.png,.bmp"
    runat="server"></denik:file>
  </div>
  <div class="bpadding">
    <asp:Button Text="Nahrát soubor" ID="SubmitFile" runat="server"
      OnClientClick="return Validate();"
      OnClick="SubmitFile_Click" />
    </div>
</form>
```

Výběr souboru *denik:file* má nastavený atribut *Accept*. Díky tomu se průzkumníku souborů nastaví požadované podporované typy souborů. JavaScript funkce *Validate* vyvolaná kliknutím na tlačítko *SubmitFile* nejdříve definuje prvky stránky pro zobrazování chybových hlášek:

```
function Validate() {
  var uploadedFile = document.getElementById("<%=UploadFile
    .FindControl("InputFile").ClientID %>");
  var msgIcon = document.getElementById("<%=Message
    .FindControl("Message_icon").ClientID %>");
  var msgText = document.getElementById("<%=Message
    .FindControl("Message_body").ClientID %>");
```

Dále je zkontrolována velikost souboru, která je získána v bajtech. Jeden megabajt je roven jednomu milionu bajtů:

```
  if (uploadedFile.files[0].size > 1000000) {
    msgText.innerHTML = "Soubor je příliš velký.";
    msgText.setAttribute("class", "message message-alert");
    msgIcon.innerHTML = "<i class=\"fa fa-exclamation fa-lg\"></i>";
    msgIcon.setAttribute("class", "message-icon message-alert");
    return false;
  }
```

Pokud je velikost souboru menší nebo rovna 1 MB, je přečtena jeho přípona. Ta se musí po převedení na malá písmena shodovat s požadavkem:

```
  var ext = uploadedFile.value.substr(
    uploadedFile.value.indexOf(".") + 1, uploadedFile.value.length);
  if (ext.toLowerCase() != "jpg" && ext.toLowerCase() != "png"
    && ext.toLowerCase() != "bmp") {
    msgText.innerHTML = "Je vyžadován obrázek JPEG, PNG nebo BMP.";
    msgText.setAttribute("class", "message message-alert");
    msgIcon.innerHTML = "<i class=\"fa fa-exclamation fa-lg\"></i>";
    msgIcon.setAttribute("class", "message-icon message-alert");
    return false;
  }
  return true;
}
```

Splňuje-li vybraný soubor všechny podmínky, vrátí funkce hodnotu *true*, v opačném případě hodnotu *false*. Pokud událost *OnClientClick* odesílacího tlačítka *SubmitFile* vrátí hodnotu *false*, zabrání se tím odeslání formuláře. Jelikož klientský kód nepovažuji za spolehlivý a je možné ho „obejít“, je na webovém serveru po odeslání požadavku provedena opět kontrola odeslaného souboru:

```
protected void SubmitFile_Click(object sender, EventArgs e) {
  if (isViewOnly) Response.Redirect("/dashboard/diaries.aspx");
  HttpPostedFile file = UploadFile.PostedFile;
  string fileName = file.FileName;
  string contentType = file.ContentType;
  if (file.ContentLength < 1000000) {
    if (contentType.ToLower() == "image/jpeg" ||
      contentType.ToLower() == "image/png" ||
      contentType.ToLower() == "image/bmp") {
```

Po ověření velikosti souboru a jeho datového typu následuje uložení obrázku do SQL tabulky *Diary_stamps* (viz tabulka č. 4):

```
using (Stream fs = file.InputStream) {
  using (BinaryReader br = new BinaryReader(fs)) {
    byte[] imageBytes = br.ReadBytes((int)fs.Length);
    using (var connetion = new SqlConnection(connectionString)) {
      using (var cmd = new SqlCommand(@"
        INSERT INTO [Diary_stamps]
          ([diary_id]
          ,[filename]
```

```
            ,[image]
            ,[type])
          VALUES
            (@diary_id
            ,@filename
            ,@image
            ,@type)
          ", connetion)) {
          cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
          cmd.Parameters.AddWithValue("@filename", fileName);
          cmd.Parameters.AddWithValue("@image", imageBytes);
          cmd.Parameters.AddWithValue("@type", contentType.ToLower());
          connetion.Open();
          cmd.ExecuteNonQuery();
        }
      }
      Message.Text = "Soubor byl nahrán.";
      Message.Type = Stavebni_Denik.Controls.Message.TypeOption.OK;
    }
}
```

V databázi je obrázek uchován ve formě *byte[]*. Pro zobrazení na webové stránce je *byte[]* převeden na *Base64* řetězec. Zobrazování HTML obrázků pomocí *Base64* demonstruji na následující část zdrojového kódu:

```
sb.AppendLine($"
<img src=\"data:{reader["type"]};base64,{Convert.ToBase64String(
(byte[])reader["image"])}\"alt=\"{reader["filename"]}\" /></div>
");
```

Atribut obrázku *src* obsahuje nejdříve datový typ obrázku. Ten je vložen ze sloupce *type* tabulky *Diary_stamps*. Poté následují data obrázku.

4.2.14 Uzavření stavebního deníku

Na stránce *close-diary.aspx* (viz příloha č. 13) je možné uzavřít stavební deník. Uzavření deníku je nevratné, a proto je od uživatele vyžadováno jeho heslo. Tím se předejde neúmyslnému uzavření deníku a zároveň se ověří totožnost majitele stavebního deníku. Když je stavební deník uzavřen, nelze do něj vkládat záznamy o stavbě, autorizační razítka, úpravy či doplňky. Uzavřením se také odeberou veškeré náhledy do stavebního deníku a nelze je už přidávat. Uživatel má ale možnost uzavřený stavební deník exportovat do PDF souboru. Samotný proces exportování je popsán v kapitole 4.2.15.

Před zobrazením stránky *close-diary.aspx* je v události *Page_Load* ověřena totožnost uživatele. Pokud se jedná o majitele stavebního deníku a není otevírán

uzavřený stavební deník, dojde k načtení stránky. Uživatel je seznámen s následky uzavření stavebního deníku. Pokud tuto operaci chce provést, musí zadat své heslo do textového pole a potvrdit kliknutím na tlačítko „Uzavřít stavební deník“, čímž se požadavek odešle na webový server. Na serveru je nejprve zkontrolováno, zda bylo heslo vyplněno. Pokud ano, ověří se, zda je zadané heslo správné.

Po ověření hesla probíhá smazání náhledů do stavebního deníku v SQL *Diary_Views* (viz tabulka č. 2):

```
using (var connection = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
  DELETE FROM [Diary_views]
  WHERE [diary_id] = @diary_id", connection)) {
    cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
    connection.Open();
    cmd.ExecuteNonQuery();
  }
}
```

Následuje úprava SQL tabulky *Diaries* (viz tabulka č. 1), ve které se nastaví hodnota sloupce *closed*:

```
using (var connection = new SqlConnection(connectionString)) {
  using (var cmd = new SqlCommand(@"
  UPDATE [Diaries]
  SET [closed] = 1
  WHERE [id] = @diary_id", connection)) {
    cmd.Parameters.AddWithValue("@diary_id", pageDiary.ID);
    connection.Open();
    cmd.ExecuteNonQuery();
  }
}
```

Nastavením sloupce *closed* tabulky *Diaries* na hodnotu *1* je stavební deník považován za uzavřený. Program takový deník nepovolí jakkoliv dále upravovat. Při úspěšném dokončení této operace se uživateli ukáže hláška „Stavební deník byl uzavřen“.

4.2.15 Export stavebního deníku

Pro vytváření PDF souborů je v tomto programu použita knihovna iTextSharp verze 4.1.6 pod licencí LGPL[5]. Knihovna umožňuje pohodlnou tvorbu a digitální

[5] Library General Public License, tedy licence svobodného softwaru.

podepisování PDF souborů v prostředí .NET. Stránka *export-diary.aspx* (viz příloha č. 14) po ověření totožnosti uživatele a kontrole, je-li otevírán uzavřený stavební deník, nabízí možnost exportu do PDF. Je nutné ve formuláři připojit svůj kvalifikovaný certifikát a heslo k privátnímu klíči. Formulář odešle požadavek na *Web Handler pdf.ashx*. Ten vygeneruje PDF soubor a odešle ho uživateli jako přílohu ke stažení.

Pomocí HTTP POST obdrží *pdf.ashx* soubor odeslaný uživatelem, heslo k privátnímu klíči a identifikační číslo stavebního deníku. Provede se kontrola, zda je požadavek od uživatele, který je vlastníkem stavebního deníku. Dále je třeba ověřit, zda odeslaný soubor je certifikát:

```
HttpPostedFile certificate = context.Request.Files["certificate"];
string certificatePassword = context.Request.Form["password"];

byte[] fileData = null;
using (var br = new BinaryReader(certificate.InputStream)) {
  fileData = br.ReadBytes((int)certificate.InputStream.Length);
}
X509Certificate2 cert = new X509Certificate2();
try {
  cert.Import(fileData, certificatePassword,
X509KeyStorageFlags.PersistKeySet);
} catch {
  context.Response.Redirect($"/dashboard/export-
    diary.aspx?id={diary.ID}&error=1");
  return;
}
```

Dále se ověří, zda certifikát obsahuje privátní klíč:

```
if (!cert.HasPrivateKey) {
  context.Response.Redirect($"/dashboard/export-
    diary.aspx?id={diary.ID}&error=2");
  return;
}
```

Poslední kontroly ověří, jestli je certifikát platný:

```
if (DateTime.Now < cert.NotBefore) {
  context.Response.Redirect($"/dashboard/export-
    diary.aspx?id={diary.ID}&error=3");
  return;
}
if (DateTime.Now > cert.NotAfter) {
  context.Response.Redirect($"/dashboard/export-
    diary.aspx?id={diary.ID}&error=4");
  return;
}
```

Pokud některá kontrola nevyhoví, je požadavek přesměrován zpět na stránku *export-diary.aspx* s příslušnou hodnotou proměnné *error* v QueryString. Uživateli se zobrazí chybová hláška s vysvětlením, proč nedošlo ke zpracování požadavku.

Kód pokračuje načtením údajů stavebního deníku z databáze a ověřením, jedná-li se o uzavřený stavební deník. Pokud ano, začne samotný proces generování PDF souboru, jehož součástí je:

- titulní stránka,
- identifikační údaje stavby,
- zúčastněné strany včetně změn a doplňků,
- údaje o projektové a ostatní technické dokumentaci stavby včetně změn,
- seznam dokumentů a dokladů ke stavbě včetně doplňků,
- autorizační razítka,
- denní záznamy stavby.

Postup generování výše uvedených bodů se podobá postupu na stránce *diary.aspx* (viz kapitola 4.2.7), ale místo generování HTML kódu je zde používána knihovna iTextSharp pro generování PDF. Použité SQL a LINQ dotazy jsou však identické. Pro práci s touto knihovnou jsem čerpal poznatky z oficiální stránky pro vývojáře [6]. Pro konkrétní demonstraci využívání této knihovny uvedu příklad výpisu seznamu dokumentů a dokladů ke stavbě. Funkce *GeneratePDF* generující PDF soubor začíná:

```
private MemoryStream GeneratePDF() {
  MemoryStream stream = new MemoryStream();
  Document document = new Document(PageSize.A4, PageMargins.Left,
    PageMargins.Right, 200, PageMargins.Bottom);
  PdfWriter writer = PdfWriter.GetInstance(document, stream);
  document.Open();
```

Samotný výpis seznamu dokumentů a dokladů ke stavbě pak řeší následující část kódu funkce *GeneratePDF*:

```
document.NewPage();
var documents = (from item in details where (item.Key == "Dokument"
  && item.Initial == true) select item);
var documentAddons = (from item in details where (item.Key ==
  "Dokument" && item.Initial == false) orderby item.DateTime select
  item);
Paragraph DocsHeading = new Paragraph("Seznam dokumentů a dokladů ke
  stavbě", fontLarge);
DocsHeading.SpacingAfter = 24;
document.Add(DocsHeading);
```

```
PdfPTable tableDocs = new PdfPTable(1);
tableDocs.WidthPercentage = 100;
tableDocs.SpacingAfter = 12;
tableDocs.AddCell(MyCell("Seznam dokumentů a dokladů ke stavbě",
fontNormalBold));
foreach (var item in documents) {
  tableDocs.AddCell(MyCell(item.Value, fontNormal));
}
document.Add(tableDocs);
```

Tato ukázka zdrojového kódu využívá moji pomocnou funkci *MyCell*, která vytvoří *Paragraph* a vloží ho do *PdfPCell*. Vzhledem k tomu, že téměř veškerý obsah PDF souboru tvoří tabulky, je tato funkce velké zjednodušení:

```
private static PdfPCell MyCell(string text, Font font, int colspan =
1, int verticalAlign = Element.ALIGN_MIDDLE, int horizontalAlign =
Element.ALIGN_LEFT, int border = Rectangle.BOX) {
  Paragraph p = new Paragraph(text, font);
  p.Alignment = horizontalAlign;
  PdfPCell cell = new PdfPCell();
  cell.Colspan = colspan;
  cell.UseAscender = true;
  cell.UseDescender = true;
  cell.PaddingTop = 12;
  cell.PaddingRight = 8;
  cell.PaddingBottom = 8;
  cell.PaddingLeft = 8;
  cell.VerticalAlignment = verticalAlign;
  cell.Border = border;
  cell.AddElement(p);
  return cell;
}
```

Konečným krokem před odesláním hotového PDF souboru je jeho digitální podepsání kvalifikovaným certifikátem, který byl k požadavku přiložen. K tomu slouží funkce *SignedPDF*:

```
private MemoryStream SignedPDF(MemoryStream pdf, byte[]
certificateFileData, string password) {
  MemoryStream file = new MemoryStream();
  PdfReader reader = new PdfReader(pdf.ToArray());
  PdfStamper stamper = PdfStamper.CreateSignature(reader, file,
    '\0', null, true);
  PdfSignatureAppearance apperance = stamper.SignatureAppearance;
  Pkcs12Store pk12;
  string alias = null;
  pk12 = new Pkcs12Store(new MemoryStream(certificateFileData),
    password.ToCharArray());
  IEnumerator i = pk12.Aliases.GetEnumerator();
  while (i.MoveNext()) {
    alias = ((string)i.Current);
    if (pk12.IsKeyEntry(alias))
      break;
```

```
  }
  X509CertificateEntry[] ce = pk12.GetCertificateChain(alias);
  var chain = new Org.BouncyCastle.X509.X509Certificate[ce.Length];
  for (int k = 0; k < ce.Length; ++k) {
    chain[k] = ce[k].Certificate;
  }
  apperance.SetCrypto(pk12.GetKey(alias).Key, chain, null,
    PdfSignatureAppearance.SELF_SIGNED);
  float width = 180;
  float height = 50;
  float left = 0.5f * reader.GetPageSize(1).Width - 0.5f * width;
  float bottom = 70;
  apperance.SetVisibleSignature(new Rectangle(left, bottom, left +
    width, bottom + height), 1, null);
  stamper.Close();
  return file;
}
```

Funkce digitálně podepíše soubor a podpis je také znázorněn na jeho titulní stránce. Vizuální podoba je dána PDF standardem. Pokud je certifikát od důvěryhodné certifikační autority, tak je podoba následující:

Obrázek č. 13 Podpis PDF souboru důvěryhodným certifikátem

V případě podepsání PDF souboru certifikátem od vydavatele, který není v operačním systému mezi důvěryhodnými, je podoba následující:

Obrázek č. 14 Podpis PDF souboru nedůvěryhodným certifikátem

Výsledek tedy není závislý na této aplikaci, ale na individuálním nastavení certifikačních autorit konkrétního uživatele. Vygenerovaný PDF soubor je z *Web Handler pdf.ashx* odeslán s HTTP hlavičkou *Content-Disposition*, která zapříčiní ukládání souboru namísto jeho otevření přímo v prohlížeči:

```
context.Response.ContentType = "application/pdf";
context.Response.AddHeader("content-disposition", $"attachment;
  filename=Stavební-deník-{diary.ID}.pdf");
context.Response.OutputStream.Write(pdf.GetBuffer(), 0,
pdf.GetBuffer().Length);
context.Response.OutputStream.Flush();
context.Response.OutputStream.Close();
pdf.Close();
```

4.2.16 Náhledy do stavebních deníků

Uživatel, který má od někoho přidělen náhled do stavebního deníku, může seznam těchto deníků vidět na stránce *views.aspx* (viz příloha č. 15). Tento seznam je sestaven pomocí SQL dotazu:

```
SELECT [Diaries].[id], [Diaries].[created], [Diaries].[name],
[Diaries].[region], [Diaries].[village], [Diaries].[cadastral]
FROM [Diaries]
INNER JOIN [Diary_views]
ON [Diaries].[id] = [Diary_views].[diary_id]
WHERE [Diaries].[closed] = 0
AND [Diary_views].[user_id] = @user_id
```

Výsledek je poté sestaven do tabulky, která vypadá jako seznam stavebních deníků (viz kapitola 4.2.6), ale není zde dělení na uzavřené a neuzavřené stavební deníky. Pokud je stavební deník uzavřen, jsou odebrány všechny náhledy. Zobrazovat seznam s uzavřenými stavebními deníky by nemělo význam.

5 Přínosy

Tato webová aplikace nabízí vedení stavebního deníku v elektronické podobě. Práce také demonstruje praktické využívání kvalifikovaných certifikátů pro elektronické podepisování.

Možnost náhledů do stavebního deníku se dá využívat pro kontrolní činnost. Tím, že se jedná o webovou aplikaci, je možné nahlížet do elektronického stavebního deníku prakticky odkudkoliv a kdykoliv.

Při zapisování záznamů se evidují přesná data. Pokud by byl stavební deník vytvářen „na poslední chvíli“, bylo by to možné poznat. Díky tomu, že je aplikace na webovém serveru, nemůže uživatel nastavením svého systému nikterak výsledné datum ovlivnit.

Závěr

Cíl práce, kterým bylo vytvoření softwaru plnící funkci elektronického stavebního deníku, se mi podařilo splnit. Naprogramoval jsem webovou aplikaci, která vychází z legislativních požadavků na stavební deníky. Podle získaných poznatků o elektronickém podepisování na platformě .NET jsem elektronické podepisování implementoval do této aplikace. Také jsem za pomocí knihovny iTextSharp naprogramoval exportování vytvořených stavebních deníků do PDF souboru, aby je bylo možno archivovat po požadovanou dobu deseti let. Protože není možné do elektronického stavebního deníku, oproti klasickému, otisknout autorizační razítko, naprogramoval jsem nahrávání souborů. Tím lze otisky razítek ke stavebnímu deníku připojit. Majitel elektronického stavebního deníku může ostatním uživatelům této aplikace přidělit náhled do deníku. Tím mohou sledovat průběh stavby prakticky odkudkoliv. Také jsem kladl důraz na zabezpečení hesla uživatelů této aplikace. Díky využití kryptografické soli a šifrovacího algoritmu SHA-256 je zajištěna dostatečná ochrana hesel. Před uložením dat do databáze nebo před jejich načtením je hlídáno, zda je uživatel k tomuto kroku oprávněn.

Výhodou elektronického stavebního deníku je čitelnost. Odpadá ruční psaní, které může být (možná i záměrně) nečitelné. Elektronické podepisování zaručuje autorství záznamů ve stavebním deníku včetně nemožnosti jejich úprav, čímž by se podpis zneplatnil. Mezi další výhody patří možnost evidovat velké množství dodavatelů nebo poddodavatelů. Klasický stavební deník má omezení „místem na papíru“. Nevýhodou tohoto řešení oproti klasickému stavebnímu deníku je nutnost připojení k internetu. Také je nezbytné, aby všechny zúčastněné osoby vlastnily elektronický podpis.

Volbou řešení elektronického stavebního deníku formou webové aplikace je možné tento program používat na všech zařízeních, které mají internetový prohlížeč podporující moderní web. Sice může být používání takových zařízení na některých stavbách problematické, ale vzhledem k tomu, že práce na PC se stává prakticky standardem, nevidím důvod, proč by stavební deník nemohl být v dnešní době veden v elektronické podobě.

Doufám, že tato aplikace povede k možnosti praktického využívání stavebních deníků v elektronické podobě.

Seznam literatury

[1] ČESKO. Zákon č. 183 ze dne 14. března 2006 o územním plánování a stavebním řádu (stavební zákon). In: *Sbírka zákonů České republiky*. 2006, částka 63, s. 2226–2289. Dostupný také z: *http://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=4909*. ISSN 1211-1244

[2] ČESKO. Vyhláška č. 499 ze dne 10. listopadu 2006 o dokumentaci staveb. In: *Sbírka zákonů České republiky*. 2006, částka 163, s. 6872–6910. Dostupná také z: *http://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=5009*. ISSN 1211-1244

[3] ČESKO. Vyhláška č. 503 ze dne 10. listopadu 2006 o podrobnější úpravě územního řízení, veřejnoprávní smlouvy a územního opatření. In: *Sbírka zákonů České republiky*. 2006, částka 163, s. 6963–7011. Dostupná také z: *http://aplikace.mvcr.cz/sbirka-zakonu/ViewFile.aspx?type=c&id=5009*. ISSN 1211-1244

[4] Přehled kvalifikovaných poskytovatelů certifikačních služeb a jejich kvalifikovaných služeb. *Ministerstvo vnitra České republiky.* [online]. 30.5.2016 [cit. 2016-11-29]. Dostupné z: *http://www.mvcr.cz/clanek/prehled-kvalifikovanych-poskytovatelu-certifikacnich-sluzeb-a-jejich-kvalifikovanych-sluzeb.aspx*

[5] Matteo Slaviero. Beginning with Digital Signatures in .NET Framework. *Simple Talk.* [online]. 07.10.2009 [cit. 2016-10-30]. Dostupné z: *https://www.simple-talk.com/dotnet/net-framework/beginning-with-digital-signatures-in-net-framework/*

[6] iText Developers. *iText.* [online]. 2016 [cit. 2016-12-13]. Dostupné z: *http://developers.itextpdf.com/*

Seznam tabulek

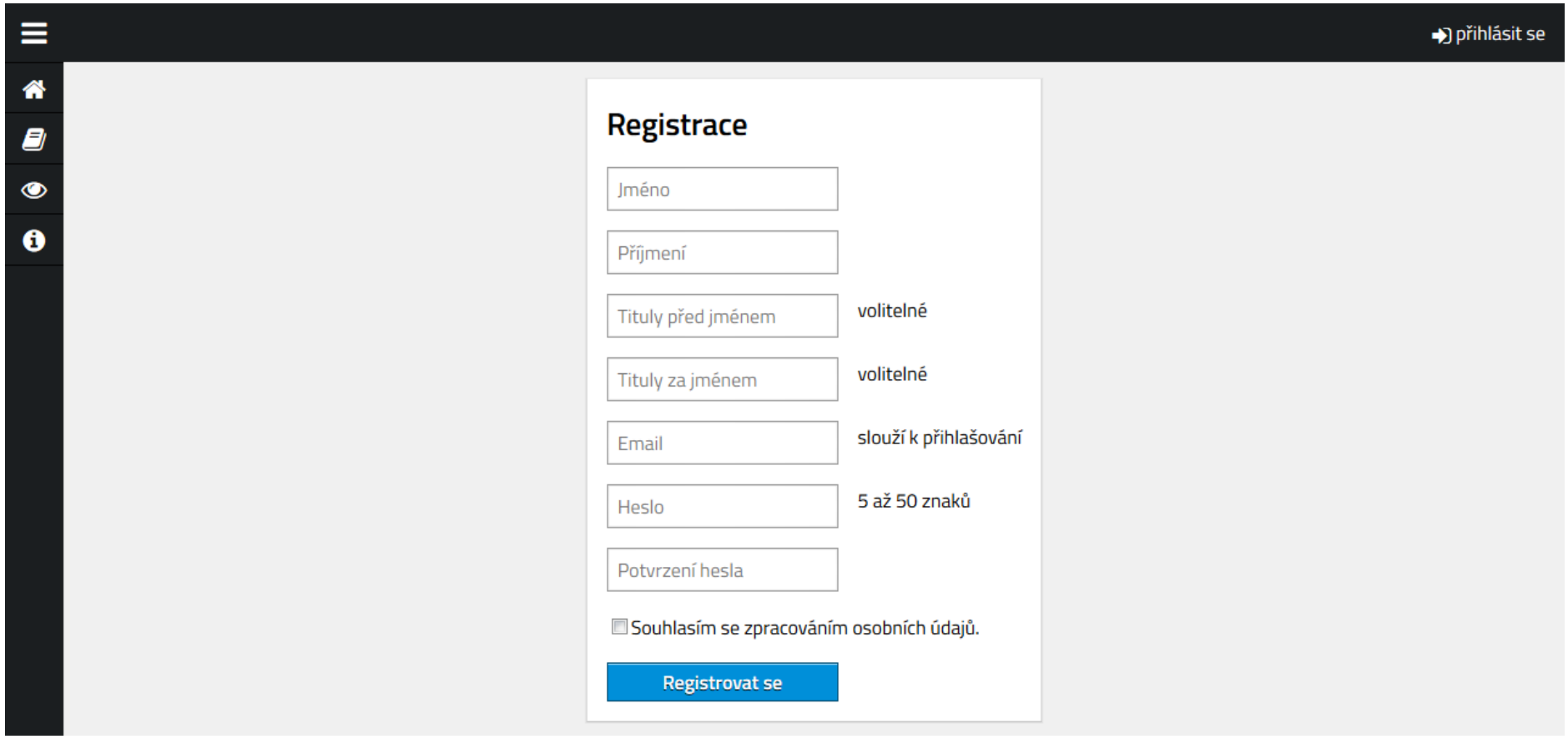
Seznam obrázků

Seznam zkratek

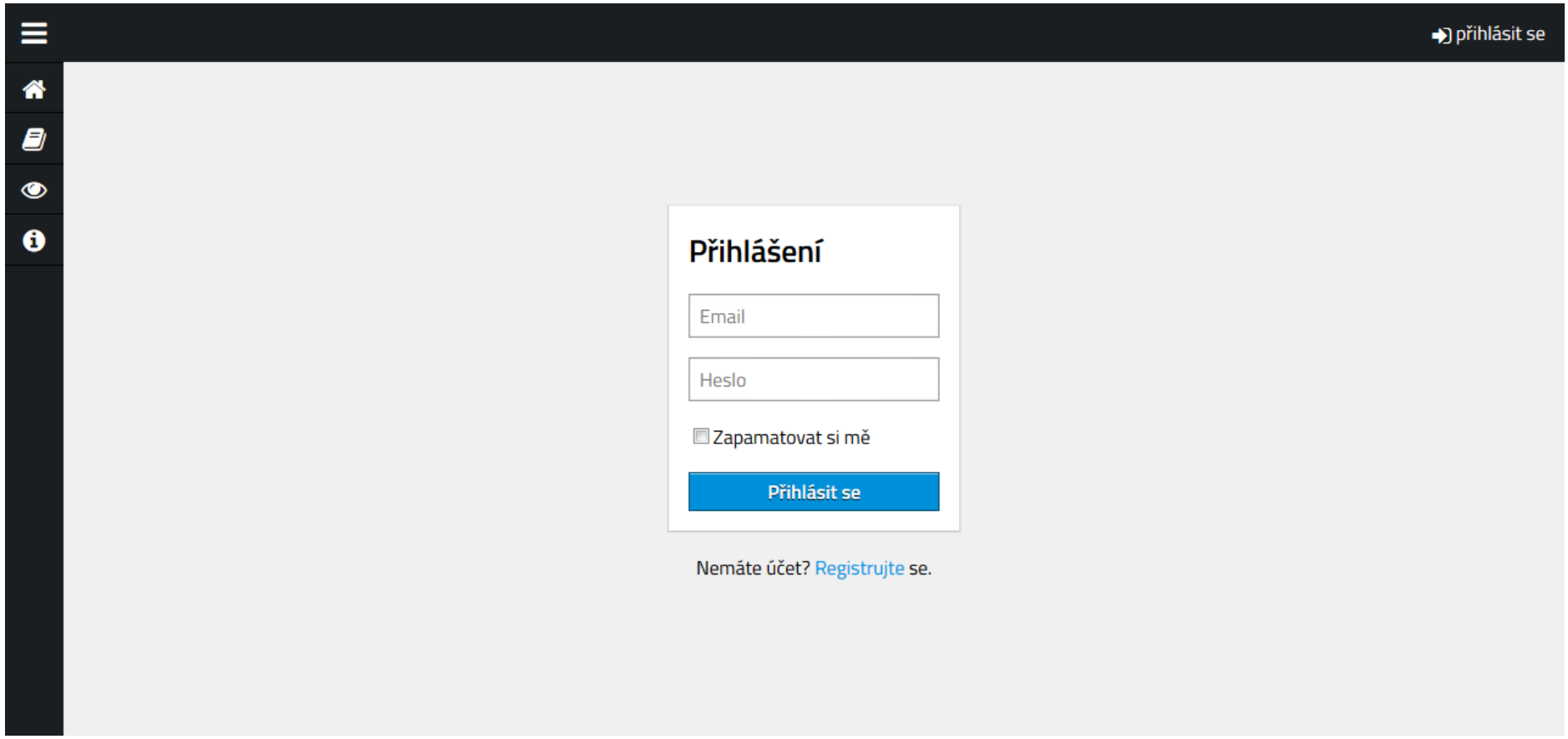
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
LINQ	Language Integrated Query
PDF	Portable Document Format
PFX	Personal Information Exchange
SQL	Structured Query Language
URL	Uniform Resource Locator

Seznam příloh

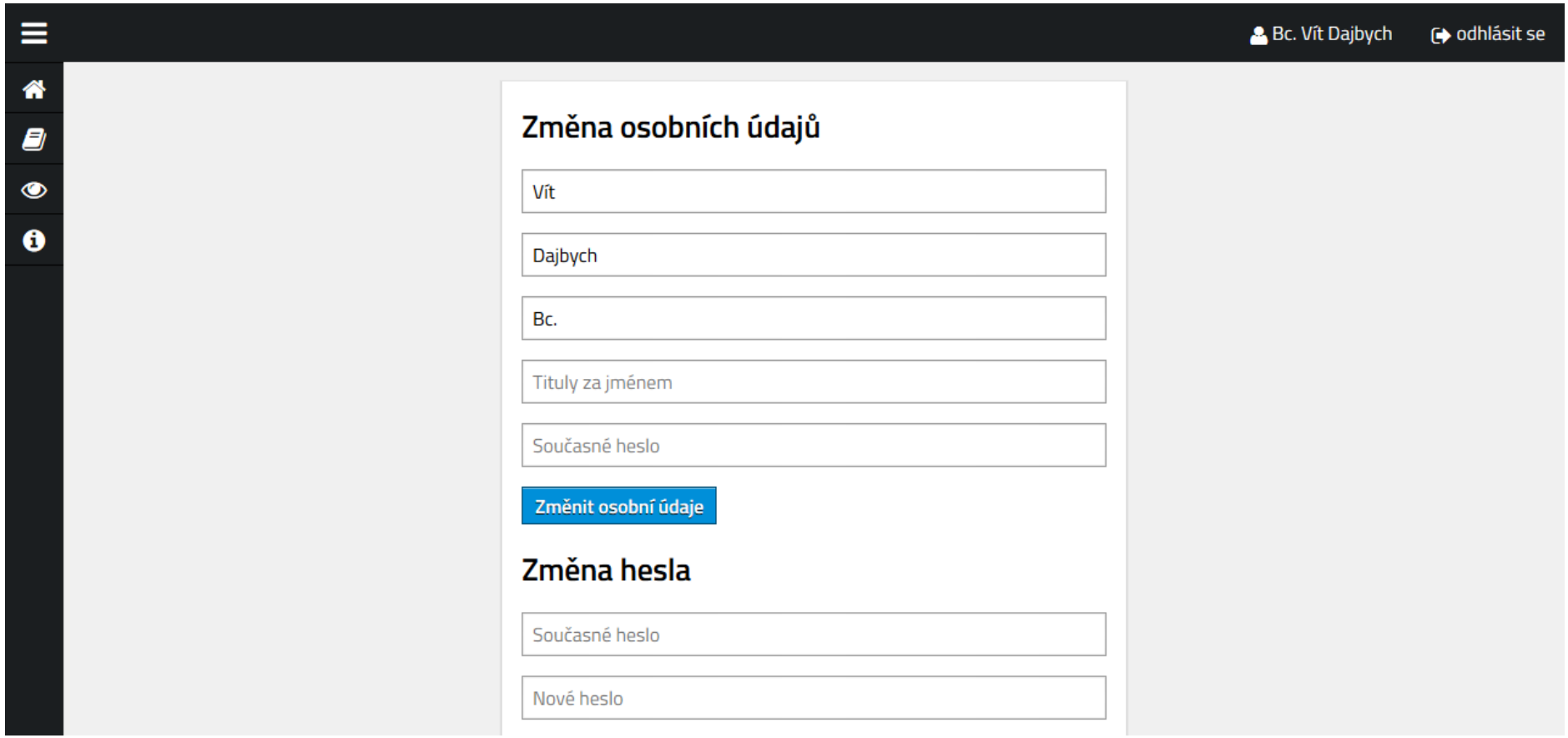
Příloha č. 1 Registrace uživatele



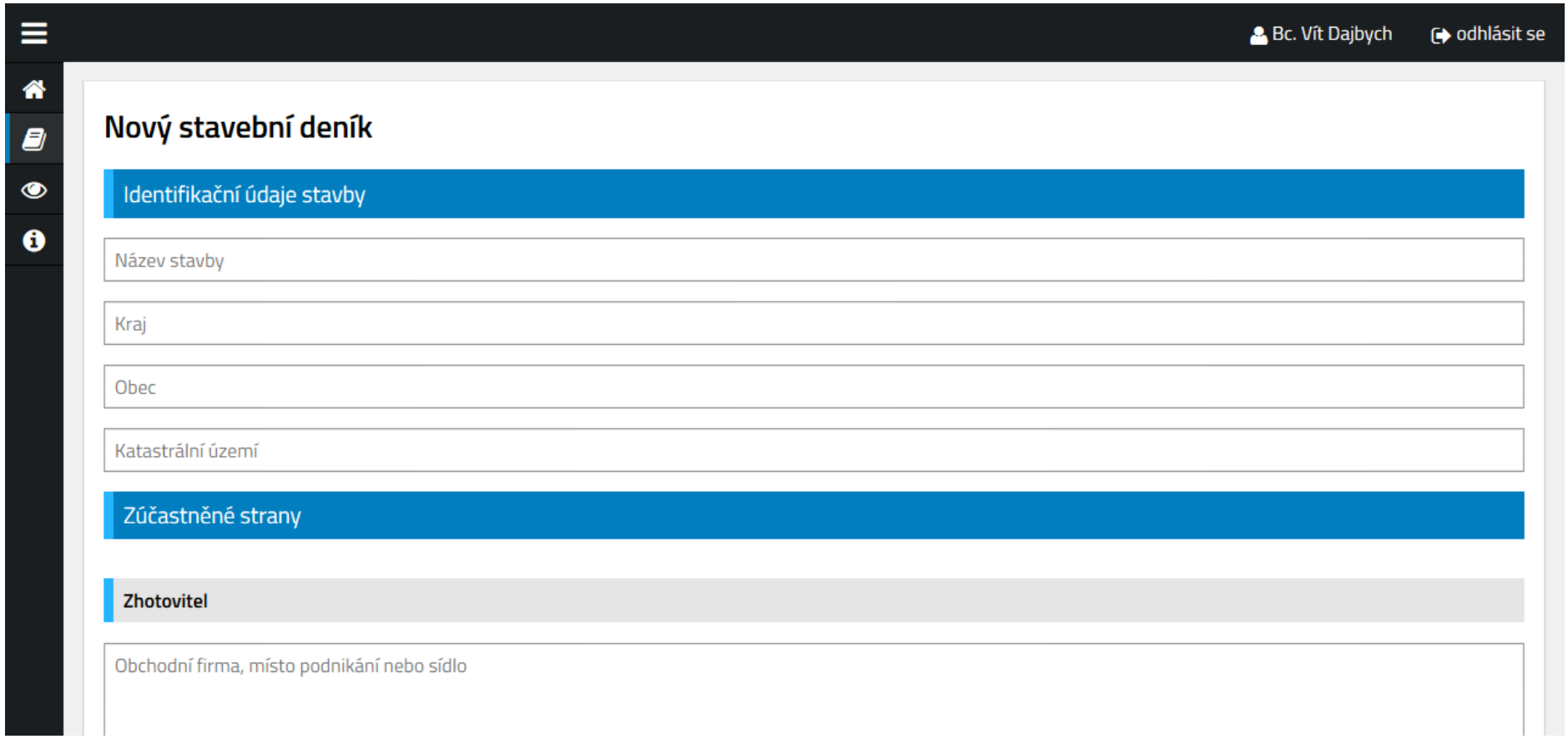
Příloha č. 2 Přihlášení uživatele



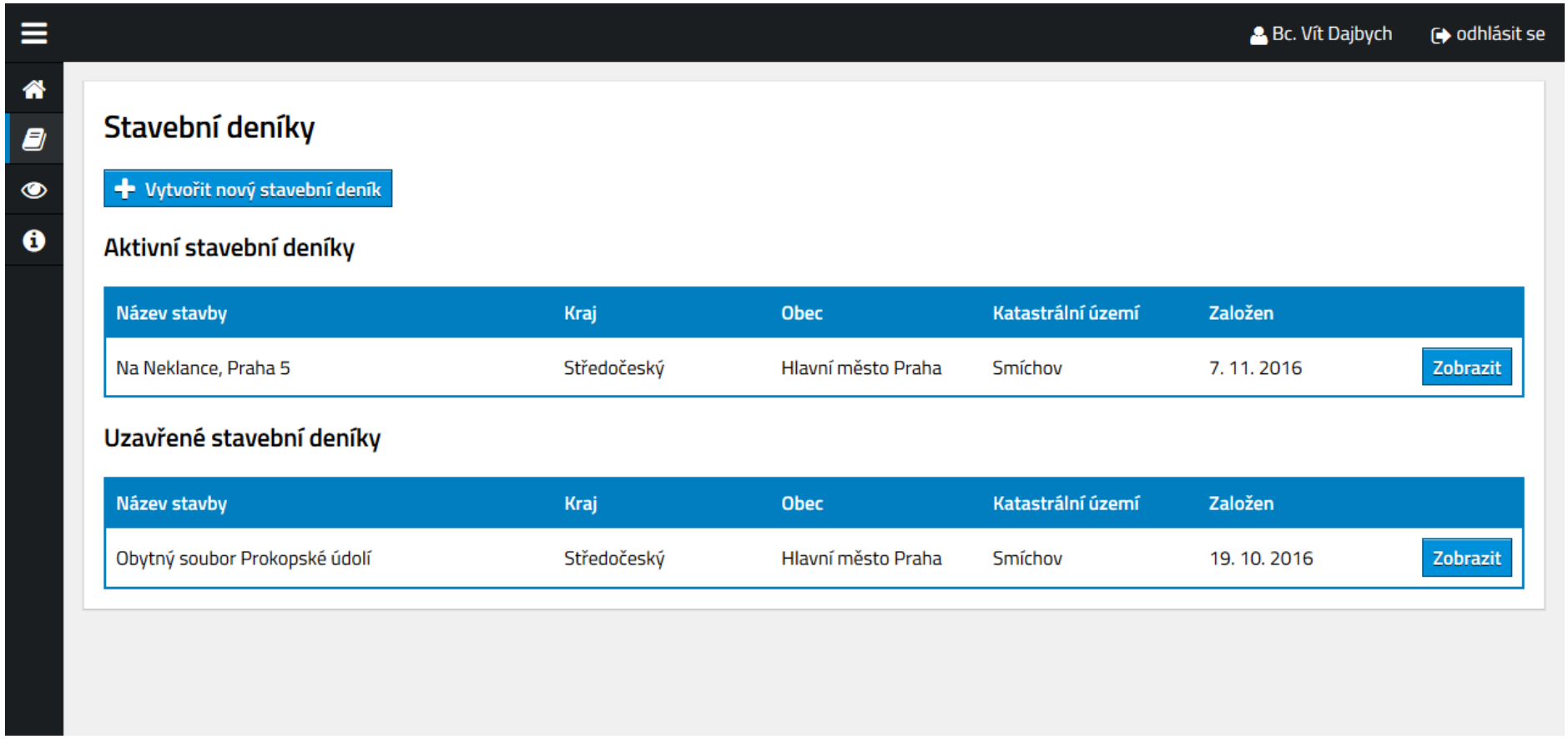
Příloha č. 3 Změna osobních údajů



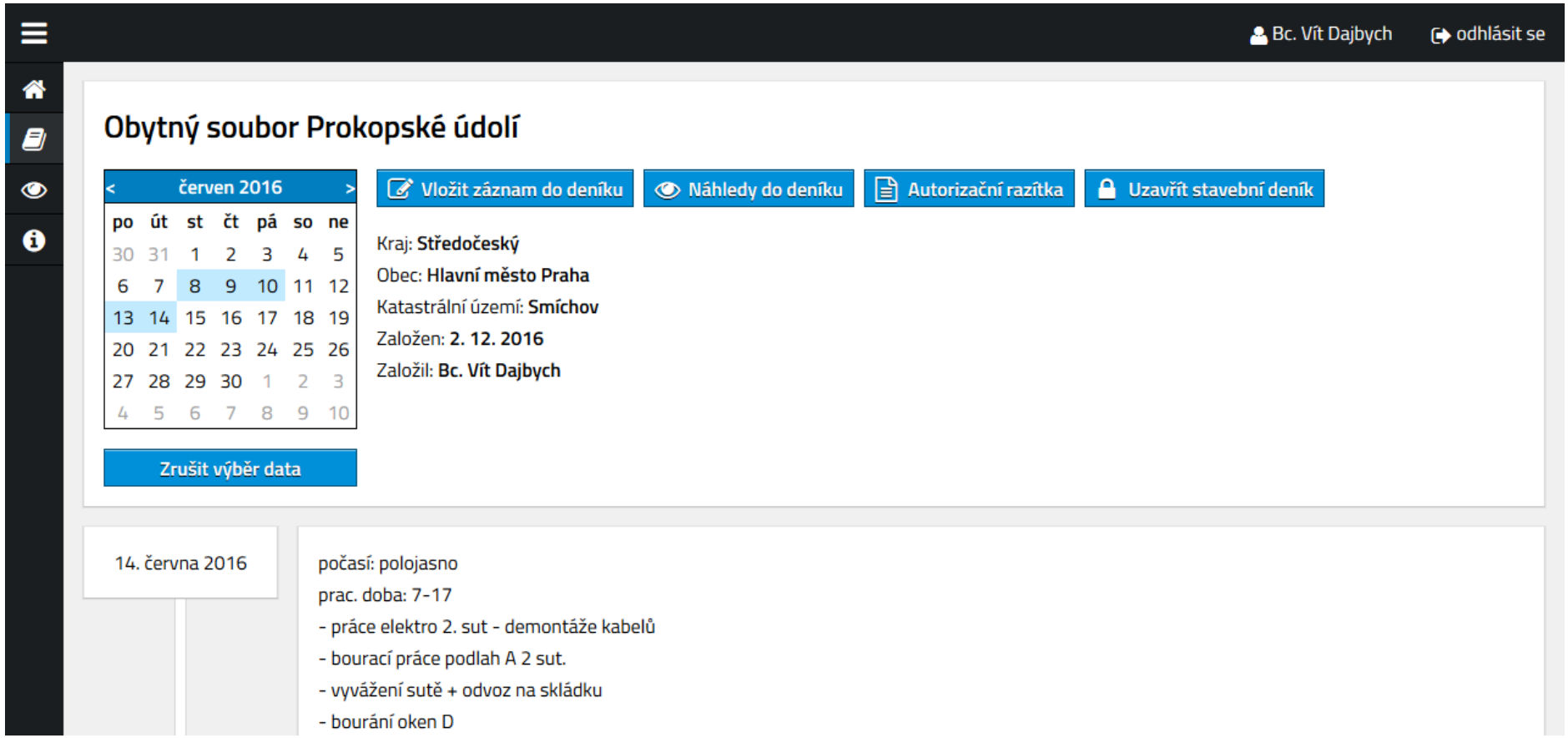
Příloha č. 4 Nový stavební deník



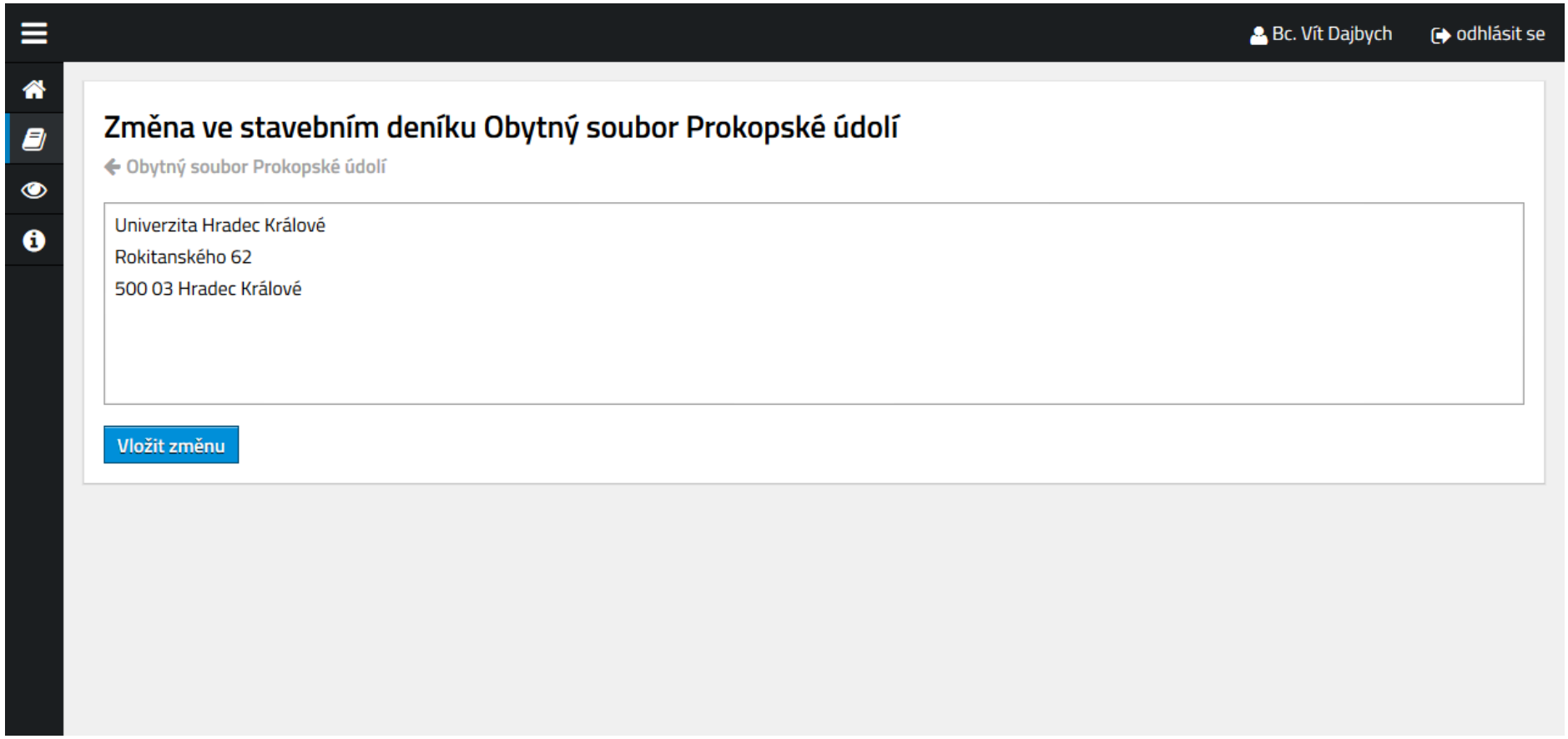
Příloha č. 5 Stavební deníky



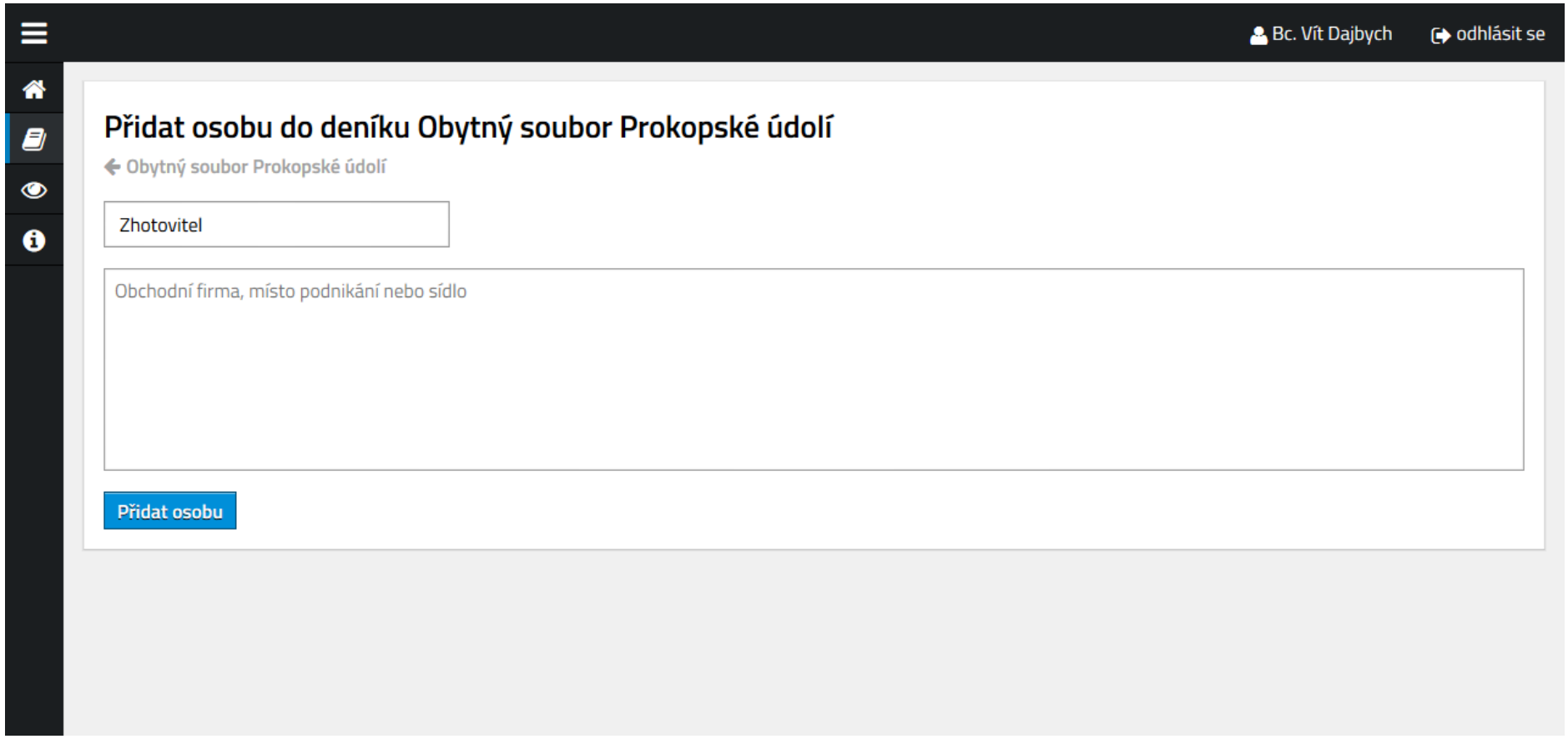
Příloha č. 6 Stavební deník



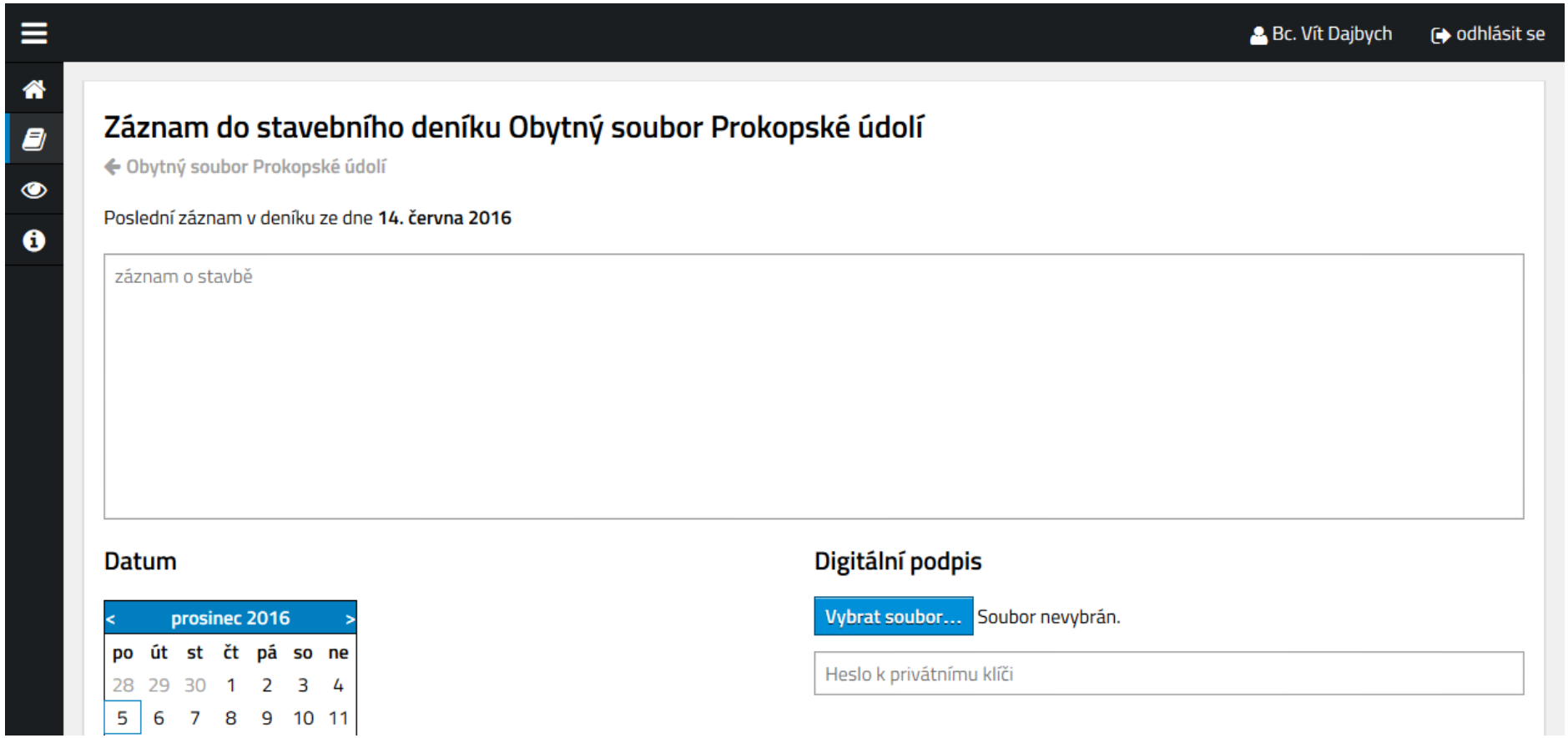
Příloha č. 7 Změna ve stavebním deníku



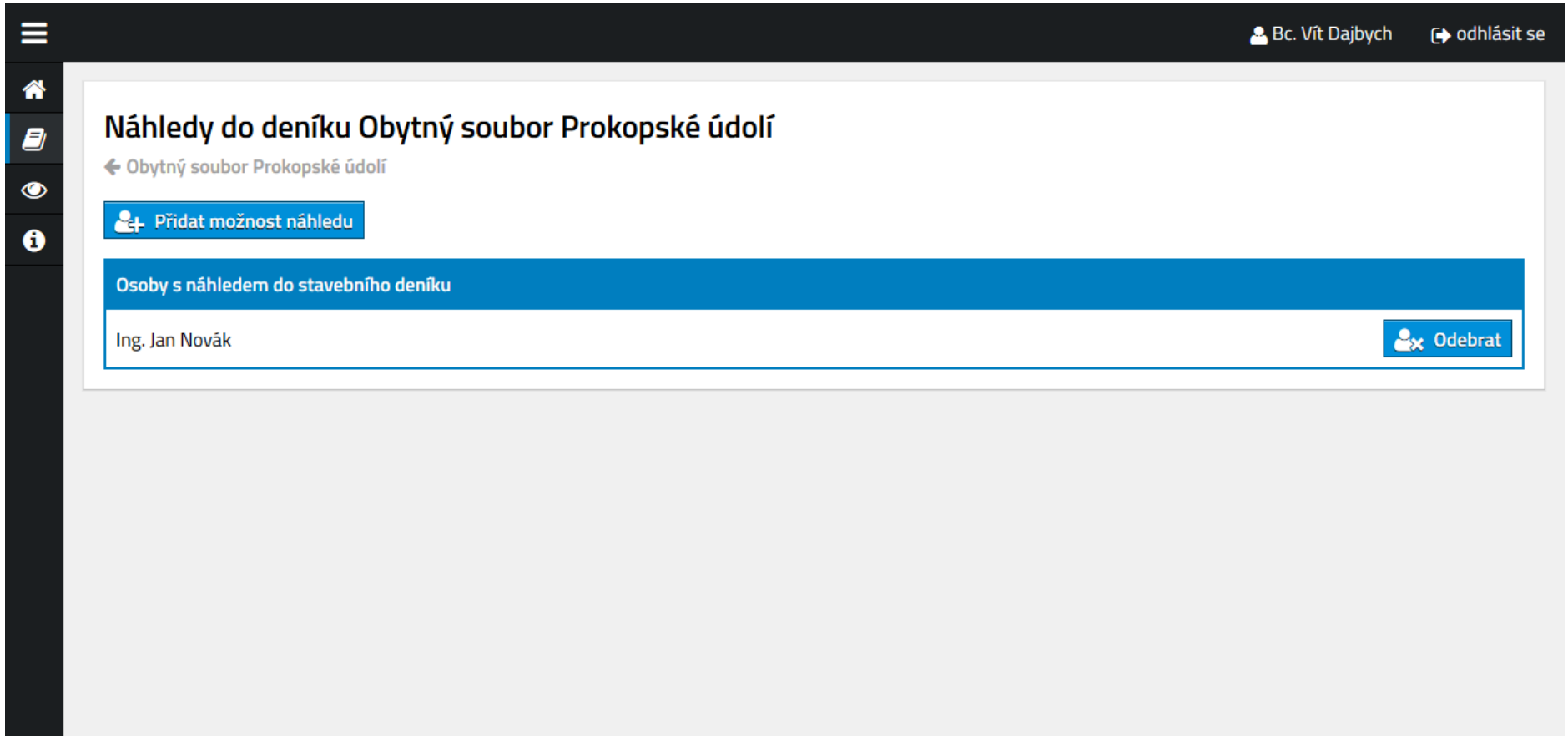
Příloha č. 8 Přidání osoby do stavebního deníku



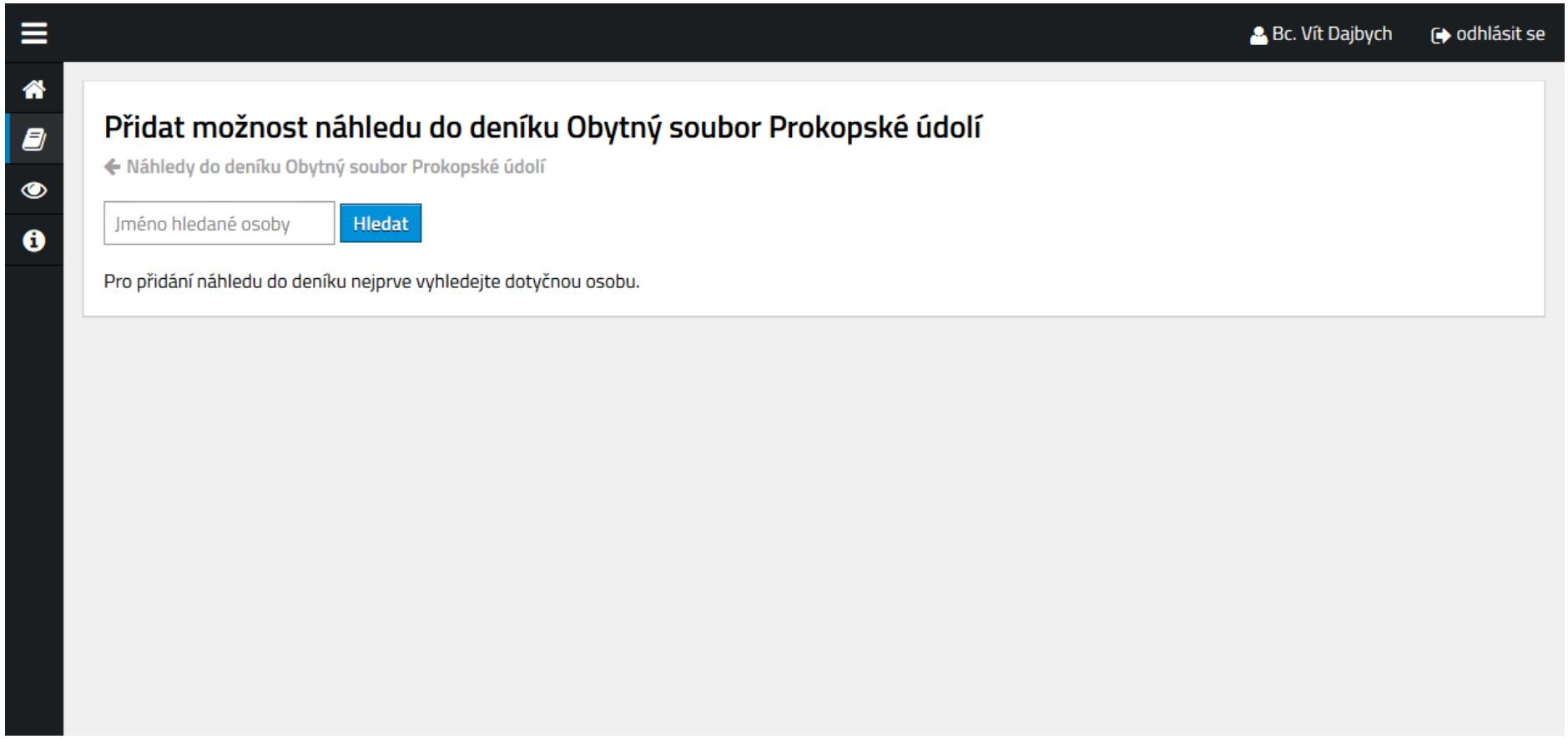
Příloha č. 9 Záznam do stavebního deníku



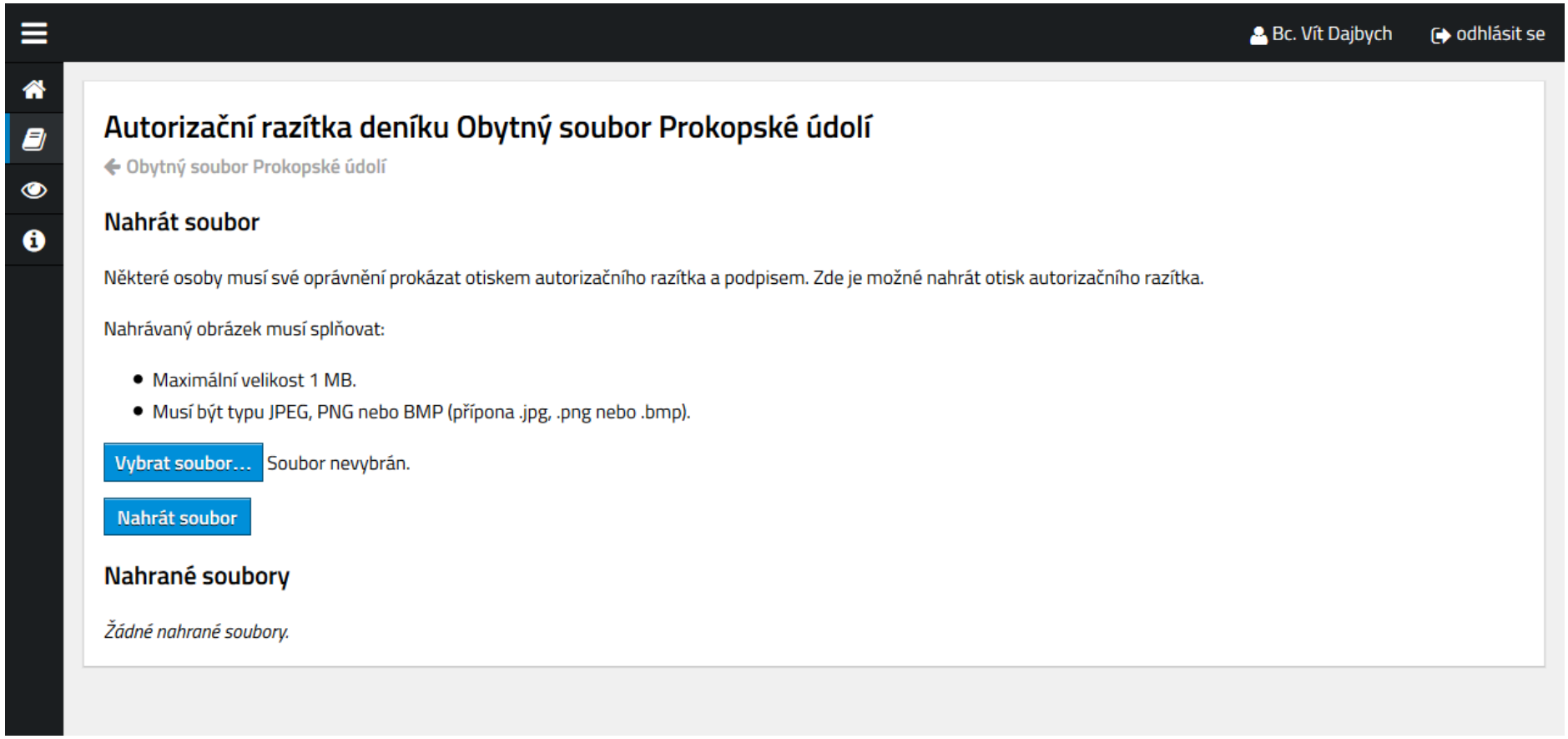
Příloha č. 10 Náhledy do deníku



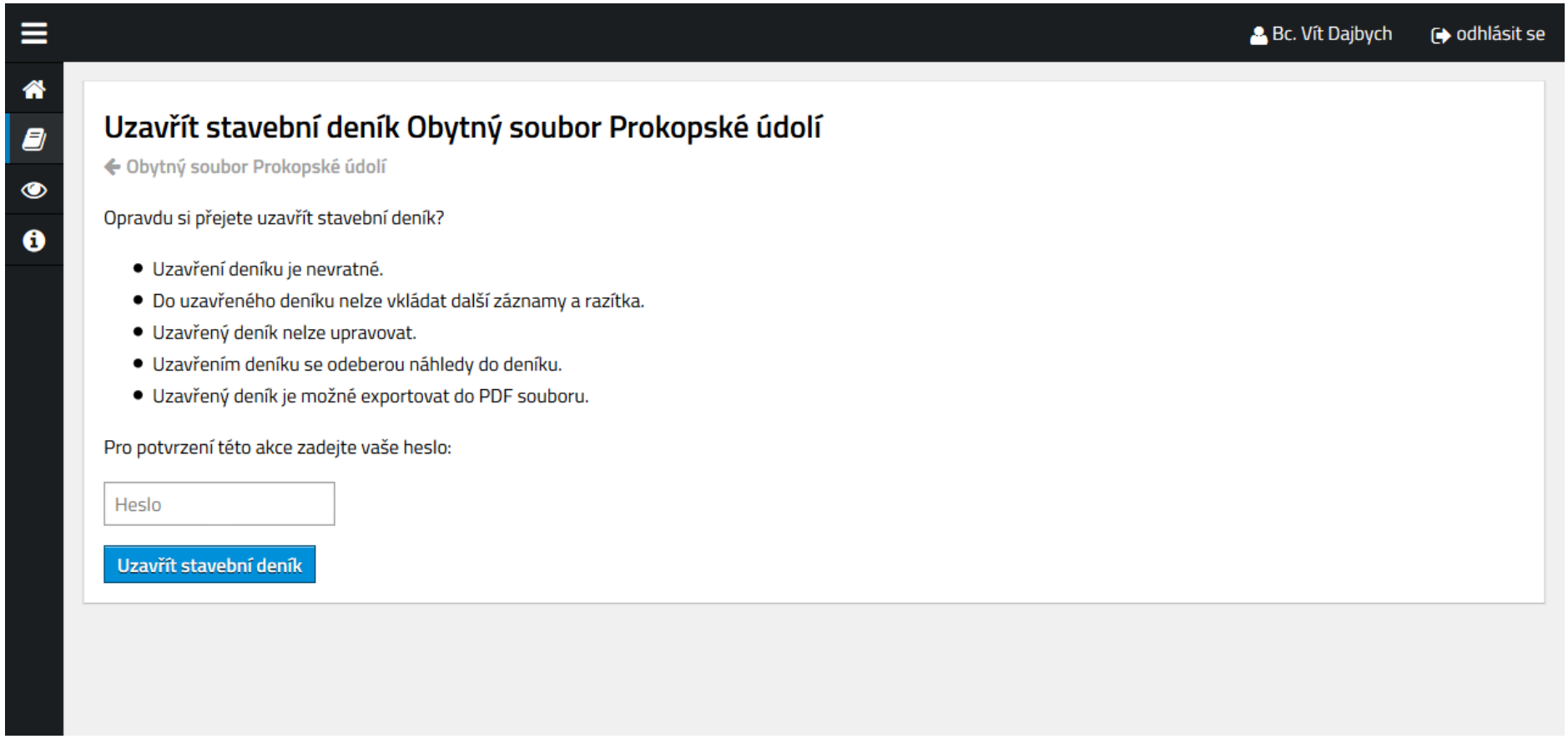
Příloha č. 11 Přidat možnost náhledu



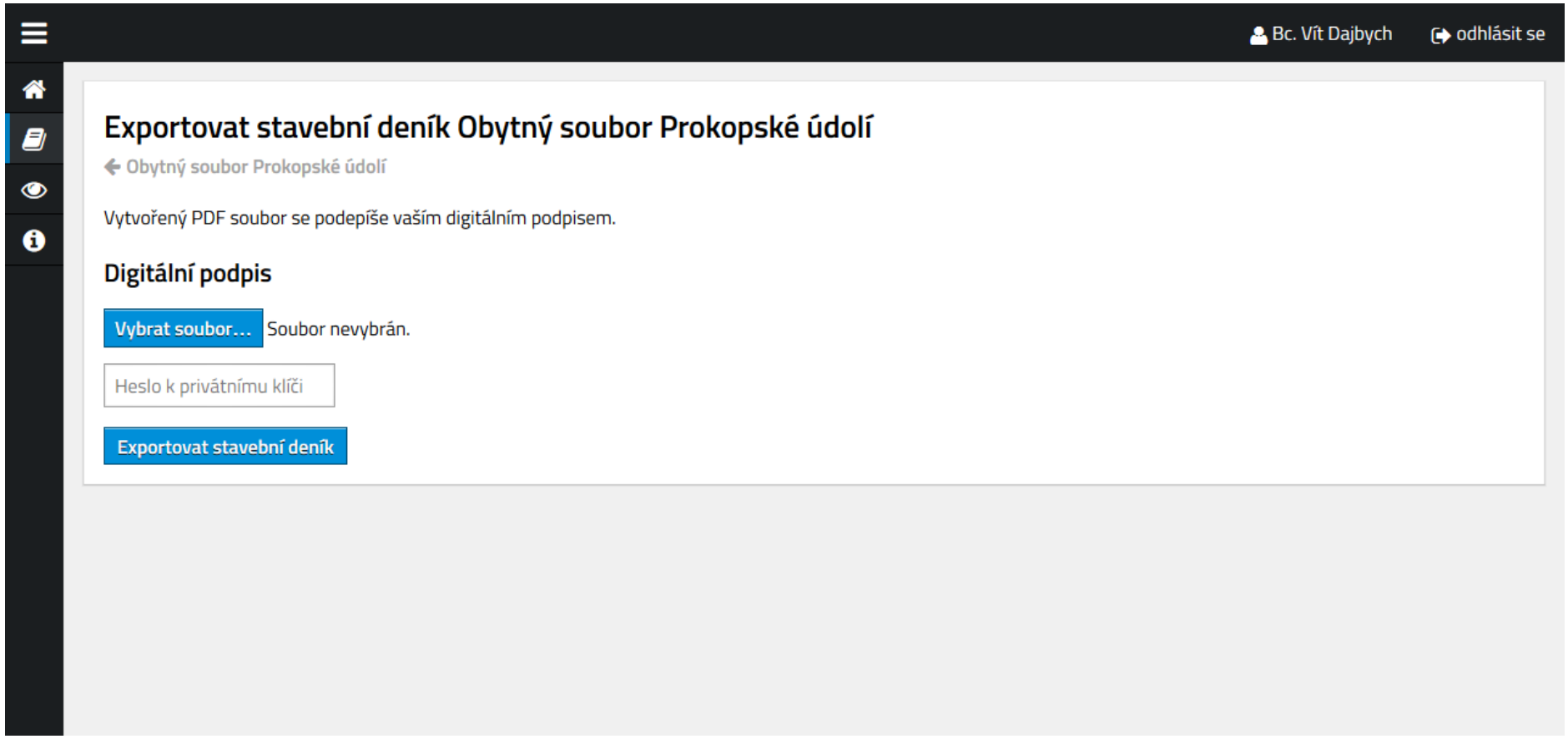
Příloha č. 12 Autorizační razítka



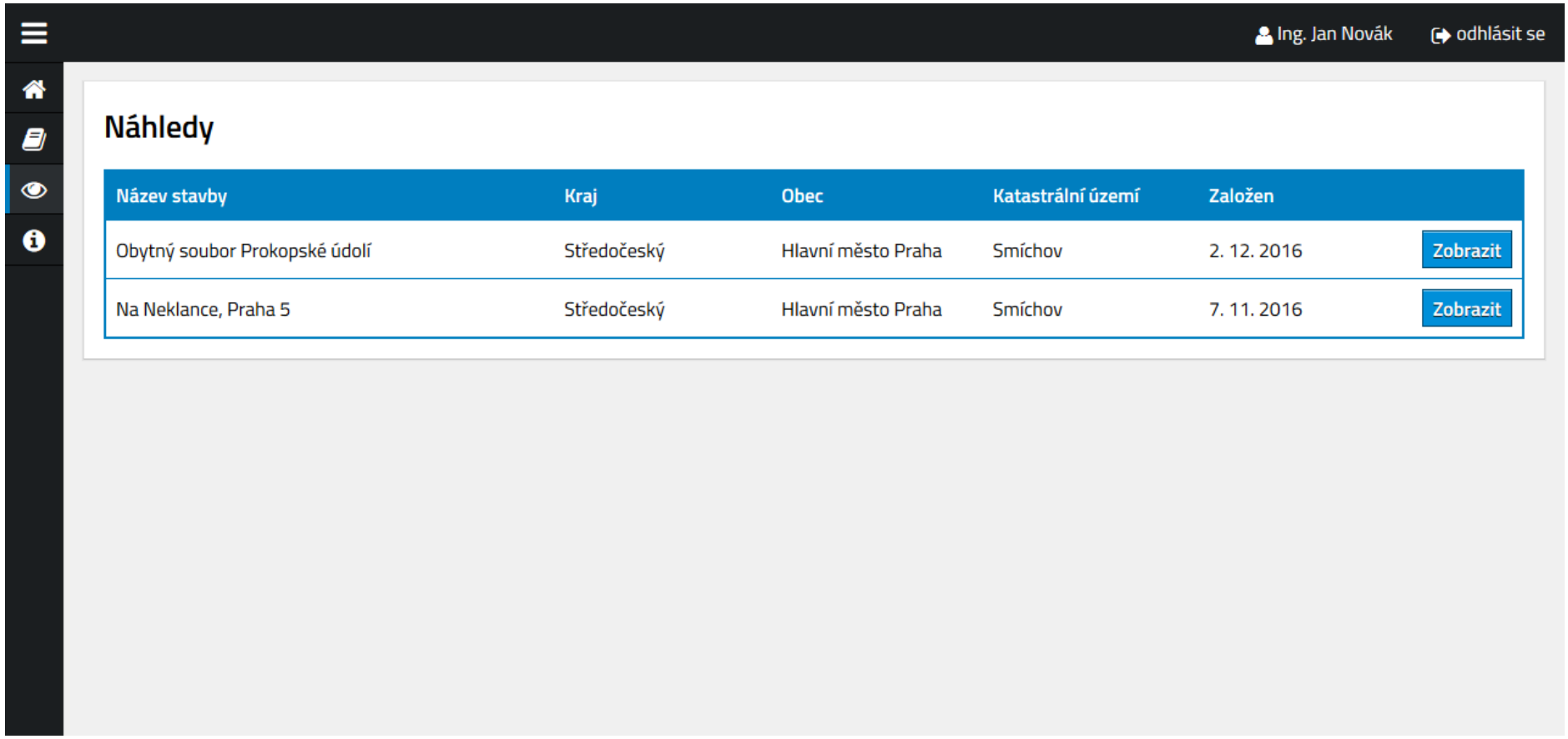
Příloha č. 13 Uzavření stavebního deníku



Příloha č. 14 Export stavebního deníku



Příloha č. 15 Náhledy



Příloha č. 16 Obsah přiloženého CD

```
├ thesis.pdf ..... text práce ve formátu PDF
└ program ..... složka s programem
   ├ packages ..... externí knihovny
   ├ Stavebni Denik ..... zdrojové kódy
   └ Stavebni Denik.sln ..... soubor projektu
```