FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Evidenční systém výkonů pro Farní charitu Jindřichův Hradec II
Student:	Bc. Jan Horyna
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem této práce je pokračovat v rozpracované realizaci evidenčního systému výkonů zaměstnanců Farní charity Jindřichův Hradec. Tato práce tematicky navazuje na diplomovou práci Bc. Davida Holkupa.

Postupujte v těchto krocích:
Proveďte analýzu současného stavu projektu.
Ve spolupráci s vedením Farní charity Jindřichův Hradec důkladně konzultujte potřeby organizace ohledně evidence výkonů pracovníků v návaznosti na již provedenou diplomovou práci Bc. Davida Holkupa.
Na základě analýzy a konzultací proveďte důkladný návrh dokončení realizace projektu.
Implementujte navrhované řešení. Neopomeňte řádnou dokumentaci implementace.
Správnost realizace ověřte vhodnými testy.
Nasaďte vaše řešení alespoň k testovacímu provozu pro Farní charitu Jindřichův Hradec.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 4. února 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Evidenční systém výkonů pro Farní charitu Jindřichův Hradec II

Bc. Jan Horyna

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

3. května 2022

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Jiřímu Hunkovi za pomoc při řešení problémů a za užitečné rady, které mi pomohly práci vylepšit. Dále bych chtěl poděkovat ředitelce Farní charity Jindřichův Hradec Mgr. Karolíně Píchové, DiS., se kterou se mi na návrhu nového systému velmi dobře spolupracovalo. Mé díky patří také dalším testerům Mgr. Petru Švepošovi a Bc. Petru Větrovskému. Děkuji také své rodině a přátelům a speciální dík míří k mé mamince Mgr. Petře Horynové za pomoc s korekturou této práce.

Věnováno mému dědovi Vláďovi, který je pro mne velkou inspirací a který se bohužel dokončení této práce o pár dnů nedožil.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 3. května 2022 . . . . . .

Odkaz na tuto práci

Horyna, Bc. Jan Horyna. *Evidenční systém výkonů pro Farní charitu Jindřichův Hradec II*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

V dnešní době se často setkáváme s tím, že naši práci narušuje přehnaná byrokracie. V rámci práce zaměstnance charity to znamená, že zaměstnanec může mnoho času trávit vyplňováním všemožných tabulek a formulářů a na pomoc klientům mu tak zbývá méně času. Tento problém zažívají také zaměstnanci Farní charity Jindřichův Hradec, kterým by vhodný informační systém mohl usnadnit práci. Tato práce se tedy věnuje analýze, návrhu, implementaci, testování a nasazení takového systému.

Klíčová slova Evidenční systém, Webová aplikace, Charita, Sociální služby

Abstract

Nowadays, we often find that our work is disrupted by too much bureaucracy. In the context of the work of charity workers, this means that they can spend a lot of time filling in all sorts of spreadsheets and forms. This leaves less time to help clients. This problem is also faced by the staff of the Farní charita Jindřichův Hradec, for whom a suitable information system could make their work easier. This thesis therefore focuses on the analysis, design, implementation, testing and deployment of such a system.

Keywords Evidence System, Web App, Charity, Social Services

Obsah

Seznam obrázků

Úvod

Závěrečné práce jsem vždy považoval jako prostředek, který by měl pomoci společnosti jako celku. Nedávalo by mi tedy osobně moc smysl pracovat na práci, jejíž výsledek by například využívala pro svůj zisk jen jedna určitá firma. Tuto vlastní podmínku neberu jako nějaké omezení, ale spíš jako příležitost pomoci s projekty, kterým by se jinak třeba nedostalo zpracování.

Jedním z generátorů podobných projektů je neziskový sektor, který se spoléhá na finance od státu nebo dárců a není tak vždy lehké zajistit plynulý chod příslušné organizace, natož ještě přicházet s možnými vylepšeními. Byl jsem tedy rád, když za mnou přišel můj kamarád a spolužák Bc. David Holkup s tím, že by rád s mojí pomocí vytvořil evidenční systém pro Farní charitu Jindřichův Hradec. Tato charita pokrývá široké spektrum služeb - poradna pro lidi v nouzi, noclehárna, výdejna potravin, materiální pomoc, nízkoprahové zařízení pro děti a mládež, sociálně aktivizační služba pro rodiny s dětmi a další.

Organizace typu charity potřebuje evidovat svou činnost nejen kvůli zpětné vazbě, ale některé údaje pochopitelně vyžaduje i stát, který si musí držet přehled o vynakládaných financích. Pro pracovníky charity se ale nesmí stát vytváření evidence komplikované a zdlouhavé, protože potom jim nezbude potřebný čas na pomoc svým klientům. Systém pro evidenci tak musí být hlavně jednoduchý a nemít zbytečné funkce navíc, které by dělaly pracovní proces složitější, než je nutné.

Práce jako celek navazuje na diplomovou práci výše jmenovaného Davida Holkupa. To pro mě byl také jeden z aspektů, proč jsem si tuto práci zvolil. Chtěl jsem si totiž vyzkoušet, jaké to je pracovat na větším projektu v rámci provázaných závěrečných prací. Tento postup dovoluje zpracovat větší projekt s možným zásadnějším přínosem. Na druhou stranu ale přináší rizika, která plynou třeba z větší potřeby komunikace a následné domluvy.

V této práci bych se chtěl nejdříve podívat na současný stav projektu, který bude potřeba analyzovat, abych mohl určit, jak projekt rozvinout dál.

V rámci projektu je také potřeba získat některé nové požadavky od zástupce organizace, takže konzultace s ním budou taktéž zásadním vstupem pro další práci. Všechny požadavky budou poté zapracovány v implementační fázi. V závěrečné fázi pak počítám s testováním a nasazením alespoň v testovacím provozu.

Kapitola 1

Cíl práce

Cílem této práce je přinést zaměstnancům Farní charity Jindřichův Hradec softwarový systém, ve kterém budou evidovat svou práci efektivněji než za aktuálního stavu. Tohoto cíle bude dosaženo díky analýze informací od zástupců charity a autora předchozí práce, která se věnovala stejnému tématu. Součástí splnění cíle je implementace a konfigurace podle zadání od zástupců charity. Výsledný produkt bude na konci procesu testován, aby mohl být úspěšně nasazen.

Kapitola 2

Analýza

2.1 Aktuální stav

Jak jsem již zmínil, práce navazuje na diplomovou práci Davida Holkupa [1]. Ten mne v září roku 2021 seznámil s tímto projektem a s ředitelkou Farní charity Jindřichův Hradec Mgr. Karolínou Píchovou, DiS. Původní domluva zněla tak, že vytvoříme pro potřeby zadavatele dva systémy, které spolu budou komunikovat. David se soustředil na část, kde by zaměstnanci Farní charity vykazovali svoji práci. Typicky tedy jde o práci s klienty ve formě různých konzultací v kanceláři nebo v terénu.

Druhou část jsem pak měl vytvořit já a zde byl kladen důraz na databáze klientů a zaměstnanců. Charita potřebuje mít přehled v tom, komu poskytuje svoje služby. U vlastních zaměstnanců se také vyplatí evidovat některé údaje. I když nejde o nějakou velkou firmu, tak databáze může pomoci třeba s vyhodnocováním, jak je daný zaměstnanec v práci efektivní a úspěšný. Pro ředitelku charity jsou pak přínosné i funkce, které například na jednom místě ukazují datum zdravotní prohlídky zaměstnance a podobně.

Rozdělení na dvě části prezentoval David [1, s. 23-25] i v architektuře zamýšleného systému. Ten se měl skládat ze dvou podprojektů, což odpovídalo výše uvedenému rozdělení práce na dvě části. Úpravy této architektury uvedu později v části s návrhem.

Charita potřebuje evidovat výkony svých zaměstnanců nejenom pro vyhodnocení v rámci vlastní organizace, ale i stát zajímá, jak si poskytovatelé těchto služeb vedou. Existují totiž zákony, které služby charit popisují. Stát je často také významným finančním přispěvatelem a musí tak na charity pohlížet z hlediska řádného hospodáře. Pro tyto účely je potřeba generovat různé statistické údaje o výkonech.

Přímo na pracovišti charity využívají pracovníci aktuálně zastaralého systému eQuip, který nabízí mnoho funkcí, z nichž některé jsou přebytečné a dělají tak interakci se systémem méně přehlednou. Stávající počítačový systém dále nabízí starší vzhled, který by si zasloužil aktualizovat. Navíc se ještě využívají prostředky jako papír a excelové tabulky, což také není vždy ideální řešení.

2.1.1 Vlastní řešení

Na začátku byla provedena analýza existujících řešení, z které vyšlo, že se pustíme do vlastního řešení. Tato analýza, na jejíž tvorbě jsem se podílel, je popsána v Davidově práci [1, s. 15-23]. Vývoj tak neměl stavět na žádném existujícím softwarovém řešení, ale velkou část zpracování jsme měli implementovat sami. Tento postup má výhodu v tom, že lze vše připravit na míru přímo podle pokynů zadavatele. Naopak nevýhodou často bývá delší čas vývoje a horší možnost budoucích rozšíření a úprav ze strany lidí, kteří nebyli součástí projektu od začátku.

Jak David popisuje v jeho práci, zvažovány byly systémy jako Jira, Trello nebo Microsoft Project. U všech se ale ukázalo, že jejich využití by bylo příliš komplikované, a proto jsme nakonec usoudili, že lepší bude zvolit cestu vlastního vývoje. David také do velké míry vypracoval jeho část systému a tu nasadil skrze službu Microsoft Azure.

2.1.2 Prototyp uživatelského rozhraní

David ve své práci zmiňuje Hi-fi prototyp [1, s. 49], který jsme ještě spolu s kolegy Bc. Lukášem Norkem a Bc. Martinem Bedrnou vypracovali v rámci předmětu NI-NUR. Prototyp byl vyvinut ve frameworku Angular [2]. V rámci vývoje tohoto prototypu proběhlo i testování [1, s. 79-83], z kterého lze také vycházet při dalším vývoji. Výhodou pro mé konkrétní využití je současně to, že jsem se na vývoji osobně podílel, přesněji jsem měl na starosti hlavně různá dialogová okna. Ostatní části jsou mi ale také známé, protože jsme návrh jednotlivých komponent probírali většinou v rámci celého týmu.

2.2 Zapojení Redminu

David v závěru jeho práce [1, s. 95] zmiňuje však i to, že se situace změnila v tu chvíli, kdy jsme se dozvěděli o systému Redmine [3]. Tento systém jsme nemohli brát v potaz, protože jsme ani nevěděli o jeho existenci. Redmine je open source projekt, vyvinutý v jazyce Ruby [4] a frameworku Ruby on Rails [5]. Po analýze jeho funkcí jsme dospěli k závěru, že do velké míry splňuje naše nároky. Nejde sice o řešení, které by se přímo věnovalo dané problematice, ale vzhledem k velmi širokému zacílení Redminu se ukázal tento nástroj jako vhodný kandidát i pro naše použití. Problémem a zároveň výhodou zde ale

je to, že toto řešení má mnoho funkcí. Jsou zde tedy funkce, které bychom využili, ale také je jich zde mnoho navíc, které by mohly pracovníka charity spíše plést. Bylo tedy potřeba vymyslet, jak se s tímto problémem vypořádat.

Redmine nabízí širokou paletu funkcí, které je možné využít:

Open source [6] Otevřenost celého systému znamená, že Redmine není jakýsi monolit, který si nelze nijak upravit. Naopak je zde přístup přímo ke zdrojovému kódu a lze ho upravovat.

REST API Redmine nabízí také komunikaci skrze REST API [7]. Díky tomu je tak možné se systémem komunikovat pomocí předem definovaného souboru akcí [8], které lze provádět nad jednotlivými objekty. Systém používá objekty jako projekty, úkoly, fronty a podobně. Nad nimi pak skrze API lze provádět CRUD operace [9]. To znamená, že objekty je možné vytvářet, číst, upravovat a mazat.

Aktivní komunita Systém si díky své otevřenosti získal pozornost skupiny lidí, která kolem něj vytvořila aktivní komunitu. Ta se o Redminu baví na diskuzních fórech [10], kde společně pomáhá s problémy ostatním uživatelům nebo například řeší možné nové funkce a jejich dopad na celý systém.

Pluginy Ne všechny nápady komunity se ale dostanou do Redminu jako takového. Systém je vyvíjen jako nezisková iniciativa malé skupinky programátorů. Ti tak musí zvažovat, jaká nová funkce bude přínosnější pro větší počet uživatelů. I tak si ale lze základní rozhraní vylepšit sadou předpřipravených rozšíření, které se v tomto světě označují jako pluginy [11]. Jejich jednoduché použití znamená vedle vlastního vývoje další možnost, jak chování Redminu upravit k vlastní potřebě.

Témata Prostředí Redminu nemusí vyhovovat všem. Další možnou cestou upravitelnosti jsou tak grafická témata [12]. Ta nepřidávají žádné nové funkce, ale přicházejí s novým vzhledem, který v některých případech může znamenat kladnější přijetí od finálních uživatelů.

Pluginy a témata spojuje jedna společná vlastnost. Za ty nejužitečnější si totiž autoři často účtují nemalé finanční prostředky. Například skrze nové pluginy lze využití Redminu dostat na opravdu novou úroveň. Příkladem tohoto vylepšení může být třeba paleta produktů od RedmineUP [13]. Při výběru těchto vylepšení je tak potřeba vybírat pečlivě i s ohledem na případné náklady.

2.3 Komunikace se zadavatelem

Jako komunikaci [14] označujeme proces, během kterého lidé mezi sebou sdílejí informace, nápady a pocity. Při vývoji softwarového produktu je předávání in-

formací mezi zhotovitelem a zadavatelem velmi důležité. Možnosti komunikace také ovlivňují dostupné metody vývoje. Cílem takové komunikace je, aby vedla k produktu, který bude zadavateli vyhovovat.

Obecně bychom mohli komunikaci rozdělit podle různých kritérií. Například podle metodického portálu rvp.cz [15] ji lze rozdělit na offline a online komunikaci. Vzhledem k tomu, že v době psaní práce nás omezuje situace s pandemií onemocnění Covid-19 a je potřeba omezit kontakty, je žádoucí, aby šlo komunikovat online. Čistě praktickým důvodem pak je geografická vzdálenost Farní charity Jindřichův Hradec a Prahy, kde práci zpracovávám. Při častější komunikaci by tak offline komunikace brala dost času kvůli dlouhé cestě.

V rámci online komunikace pak lze volit mezi přímou a nepřímou komunikací. Ta přímá má výhodu v tom, že máme okamžitou reakci druhé strany, řadí se sem tak prostředky jako telefon, videohovor nebo částečně i chat. Nevýhodou na druhou stranu je to, že obě strany musí být v jednu chvíli na společném komunikačním kanálu. Nepřímá komunikace to má obráceně. Reakce druhé strany jsou opožděné, ale zase neklade tak velké organizační nároky na komunikující. Vzhledem k tomu, že u vývoje softwaru je mnoho věcí, které je třeba domluvit, nepřímá komunikace by tak byla neefektivní a způsobovala by spíše zdržování celého projektu. Jako lepší se mi tak určitě jeví komunikace přímá a s vedením charity jsem se tak dohodl na tomto stylu komunikace.

Dalším členěním může být rozdělení podle pracovního portálu indeed.com [16], který rozlišuje celkem čtyři druhy komunikace, které se v různých úpravách vyskytují i v jiných zdrojích [17] nebo [18]:

Verbální Jde asi o nejpřirozenější formu komunikace, při které používáme mluvené formy jazyka. Požadavkem však je stejný dorozumívací jazyk. Komunikaci tímto způsobem lze provádět offline nebo online.

Neverbální Do této formy řadíme hlavně řeč těla, různá gesta nebo mimiku obličeje. Většinou jde o informace, které doplňují verbální formu a lze z nich lépe vyčíst třeba to, jaké má člověk v rámci komunikace pocity.

Vizuální V tomto případě jde hlavně o různé formy náčrtů, diagramů, map a podobně. Skrze tuto formu lze zadavateli například prezentovat návrh systému a odhalit případné nedostatky v rané fázi.

Psaná Poslední forma je v některých zdrojích slučována s verbální. Znovu jde o použití určitého jazyka, ale tentokrát je použití více nepřímé. Rozdělení do samostatné kategorie mi ale přijde užitečné, protože když je něco napsáno a obě strany se na tom shodnou, tak to má typicky větší váhu než ústní domluva.

Myslím, že všechny čtyři vyjmenované formy mohou být pro komunikaci se zadavatelem přínosné. Dokonce bych skoro řekl, že každá z nich je do jisté míry jen těžko zastupitelná. Budu se tedy snažit najít formát, který obsáhne všechny formy.

2.3.1 Vybraný styl komunikace

Vzhledem ke geografické vzdálenosti od místa využívání systému jsem zvolil k získávání informací od zadavatele sérii videohovorů. Hovory pokrývají verbální komunikaci, díky přítomnosti obrazové složky je pak přítomna i komunikace neverbální. Z vizuální komunikace plánuji probírat různé návrhy od základních náčrtů až po sofistikovanější wireframy. Psanou formu pak využiji pro přípravu na schůzku a také při zápisu ze schůzky. Toto ucelené řešení přináší hned několik výhod:

Zavedený kanál Autor předchozí práce komunikoval s ředitelkou charity víceméně stejným způsobem a lze z toho tedy vyvodit, že zadávající strana je na tento způsob zvyklá a zřejmě jí vyhovuje. Mým záměrem by tak rozhodně mělo být stavět na postupu, který se osvědčil.

Možnost interakce Komunikace by také mohla probíhat nepřímo. Mezi nepřímé způsoby komunikace lze v tomto případě zařadit třeba kontakt skrze e-mail nebo chat. Tyto formy ale neposkytnou interakci, která je velmi žádoucí. Ředitelka charity totiž nemá zkušenost s vývojem softwaru. Já na druhou stranu nemám dlouhodobou zkušenost s prací ve farní charitě. Mnoho problémů tak můžeme vyřešit společným přemýšlením o možném fungování výsledného řešení.

Průběžné vyhodnocování Díky pravidelným schůzkám se vyhnu tomu, že se u některé části systému špatně pochopíme a já pak přijdu s řešením, které nesplňuje očekávání zadavatele. Při pravidelných schůzkách a prezentaci průběžného stavu se včas odhalí případné nepřesnosti a já můžu učinit nápravu v dřívější fázi vývoje.

Větší zapojení zadavatele Pro zadávající stranu by pochopitelně bylo jednodušší sepsat zadání a po nějaké době obdržet hotový produkt. Při větším zapojení zadavatelské strany ale docílím toho, že sám zadavatel v průběhu projektu přijde na nové funkce nebo si vyhodnotí, že některé funkce nejsou tak potřebné, jak si původně myslel. Zadavatele tak stále udržuji součástí projektu.

Přínos obrazu Kromě zvukové stopy zde má velký přínos i vizuální stránka hovoru. Zaprvé se přímo vidíme, což dělá kontakt reálnější a obě strany to také více nutí držet se tématu. Zadruhé jsou zde možnosti jako sdílení obrazovky, kdy si můžeme ukazovat přesně to, co na obrazovce vidíme

sami a jak systém ovládáme. Lze se tak vyhnout větám typu: *Musíš zmáčknout tamto nad tím druhým.* nebo *To je divný, já to u sebe mám.*

Samozřejmě má ale tento přístup i svá negativa. Vzhledem ke komunikaci na dálku nemám třeba přímý kontakt s tím, jak budou uživatelé systém využívat. Nemám také vlastní zkušenost s tím, v jakém prostředí bude docházet k zadávání výkonů. Bylo by tak vhodnější být přímo na pracovišti. Jak jsem ale psal už výše, vzhledem k omezení kontaktů v rámci epidemie Covidu-19 jsem se rozhodl ke kontaktu skrze mikrofon, sluchátka, kameru a obrazovku.

2.3.2 Průběh videohovoru

Před každou virtuální schůzkou jsem si připravil sérii dotazů a témat, které jsem chtěl probrat. To do jisté míry formuje průběh každého videohovoru. Úplně na začátku rozhovoru ale ředitelce charity prezentuji pokrok, kterého jsem dosáhl od poslední schůzky. Už z této prezentace aktuálního stavu dostanu potvrzení toho, že se vývoj ubírá správným směrem, nebo korekci v případě, když jsme si v něčem posledně neporozuměli. V této první části také zadavatelskou stranu často napadnou různá vylepšení, která předtím nebyla požadována, ale vyplynula až z průběžného zpracování.

Dále otevírám jednotlivá témata, která mám poznamenaná a vytvářím si poznámky podle toho, jak si to zadávající strana představuje. Často přitom přijdeme ještě na další požadavky a možné scénáře využívání systému. Naopak u některých věcí je zase pro zadavatelskou stranu obtížné přesně určit, jaké zpracování by jí vyhovovalo. V té chvíli mohu nastínit, jak bych si danou funkci představoval já třeba podle znalosti jiných systémů.

Zápis ze schůzky pro mě poté slouží nejenom jako určitá osnova pro další práci, ale tento text využívám i k tomu, že například zadavatelce přečtu, jak jsem si to zapsal a ona to odsouhlasí nebo mi formulaci ještě opraví. Tento přístup tak slouží jako další kontrola toho, že se dobře chápeme a má eliminovat případná nedorozumění.

2.4 Požadavky na systém

V rámci této části práce bych se chtěl zaměřit na definování takzvaných funkčních a nefunkčních požadavků. Požadavky funkční [19] si lze představit poměrně snadno. Jedná se totiž o soubor funkcí, které zákazník potřebuje pro provozování své činnosti.

U nefunkčních (nebo také obecných) požadavků [19, s. 19] už to ale tak intuitivní není. Tato kategorie totiž zahrnuje nároky na systém, které běžného uživatele intuitivně nenapadnou. Řadíme sem tak například to, jak má být systém dostupný. Dále do této kategorie patří také nárok na výkonnost výsledného systému včetně nároků na časové odezvy různých úkonů. K výkonnosti se pak vážou také nároky na paměťovou náročnost. S tím pak může souviset

také finanční náročnost vývoje i následného provozu. Důležitým aspektem nefunkčních požadavků je i uživatelská přívětivost výsledného produktu. Často se zde také můžeme setkat se zkratkou FURPS [20, s. 618-620] [21], které označuje následující vlastnosti: Funkcionality (funkčnost), Usability (použitelnost), Reliability (spolehlivost), Performance (výkonnost) a Supportability (rozšiřitelnost / podporovatelnost).

U obou typů požadavků je nejdůležitější jejich popis, protože samotný název ne vždy dostatečně popisuje celý požadavek. U funkčních požadavků jsem ještě doplnil prioritu a složitost. Prioritu jednotlivých požadavků jsem získal z rozhovorů se zadavatelem. Složitost jsem si poté dovolil odhadnout sám, ale jedná se o odhad. V implementaci bude jasné, jak náročné bylo splnění konkrétního požadavku. Kombinace priority a náročnosti poté bude sloužit k rozhodnutí, v jakém pořadí funkce implementovat. Jako první tak dává smysl pracovat na požadavcích s vysokou prioritou a malou časovou náročností.

Požadavky, které zmiňoval už David, označuji **takto** a zcela nové požadavky ***takto***. I v už známých požadavcích ale byly provedeny změny, a proto je zmiňuji znovu.

2.4.1 Nefunkční požadavky

U nefunkčních požadavků mohu z velké části odkázat na Davidovu práci [1, s. 13]. U tohoto druhu požadavků nedošlo k velkým změnám. Výsledné nefunkční požadavky prezentuji v této podobě:

N1 - Dostupnost přes web
Systém má být přístupný pomocí webové aplikace, ke které budou zaměstnanci přistupovat skrze webový prohlížeč. Interakce se systémem pak bude prováděna pomocí počítače.

N2 - Nasazení v cloudu
Jelikož zákazník nemá vlastní server, tak nasazení výsledného řešení proběhne v cloudu.

N3 - Nezvýšení finanční náročnosti
Provoz celého systému nesmí být finančně náročnější než používání aktuálního systému eQuip [1, s. 14-15].

N4 - Jednoduché a přívětivé rozhraní
Uživatelské rozhraní je důležitým aspektem přijetí výsledného systému uživateli. Měřitelnost úspěchu splnění požadavku není úplně zřejmá. K posouzení splnění tohoto bodu budou sloužit data z uživatelského testování.

2.4.2 Funkční požadavky

U funkčních požadavků David [1, s. 11-13] opět shrnul požadavky pro celý systém a ne jen pro svoji část. Vzhledem k tomu, že některé požadavky přibyly a u některých se měnila priorita, rozepíšu všechny požadavku pro přehlednost znovu.

F1 - Evidence výkonů

Popis
Systém má evidovat výkony zaměstnanců, což jsou schůzky s klienty nebo jiná práce. Výkony budou zadávat sami zaměstnanci. Podle reálného členění charity budou výkony řazeny do služeb a pod aktivity, kdy jedna služba má pod sebou více aktivit. Jako příklad lze uvést třeba službu *Poradna pro lidi v nouzi*, pod kterou jsou aktivity jako *Právní poradenství* a *Finanční poradenství*. Výkony různých aktivit pak mohou mít různé uživatelsky definované atributy. Každý výkon pak lze po vytvoření zobrazit, upravit nebo smazat.

Priorita
Vysoká

Složitost
Vysoká

F2 - Evidence klientů

Popis
Klienti jsou zásadní entitou provozu celé charity. V rámci jejího provozu je potřeba uchovávat informace o klientech v rámci různých služeb. Atributy v různých službách se mohou lišit a opět zde požadujeme uživatelské vytváření atributů. Profily nebo karty klientů v různých službách spolu nejsou provázané. Provoz charity vyžaduje také možnost vytváření anonymních klientů. Informace o klientech pak musí být možno vytvářet, zobrazovat, upravovat a mazat.

Priorita
Vysoká

Složitost
Vysoká

F3 - Individuální plány

Popis
Individuální plán se váže vždy k jednomu klientovi a jeho cílem je skrze sérii výkonů pomoci klientovi s jeho problémem. Jako

příklady lze uvést například *Řešení dluhové situace* nebo *Hledání zaměstnání*. Výkony v rámci plánu jsou vždy pod jednou službou, zároveň ale mohou zasahovat do různých aktivit a tedy mít i různé atributy.

Priorita
Střední

Složitost
Střední

F4 - Evidence zaměstnanců

Popis
Zaměstnanci jsou zde vlastně v takové dvojí roli. Zaprvé jsou sami uživateli systému a vykazují do něj výkony. S tím souvisí možnost zaměstnanců se do systému přihlašovat pod vlastním účtem a upravovat si ho. Zadruhé ale o nich zaměstnavatel potřebuje sbírat informace a nakládat s nimi tak, aby se zlepšila efektivnost celé charity. Zaměstnanci tak například mají atributy jako *Platnost lékařské prohlídky* nebo *Platnost školení BOZP*. Splnění podobných podmínek bývá z hlediska zaměstnanců často povinné ze zákona a je tak žádoucí, aby o nich mělo vedení dostatečné informace.

Priorita
Vysoká

Složitost
Střední

F5 - Sdílení poznámek na nástěnce

Popis
Vedení charity vzneslo také požadavek na zpracování virtuální nástěnky, kam by zaměstnanci mohli dávat zprávy, které by byly určeny všem ostatním zaměstnancům.

Priorita
Nízká

Složitost
Střední

F6 - Vytváření statistik

Popis
Statistiky o výkonech jsou potřebné hlavně z důvodu zpětné kontroly. Tuto kontrolu na jedné straně vyžaduje zřizovatel, ale samotné vedení charity tyto statistiky ocení pro zlepšení své činnosti také.

Priorita
Vysoká

Složitost
Vysoká (v případě vlastní implementace), Nízká (v případě využití rozhraní systému Redmine)

2.5 Případy užití

Funkční a nefunkční požadavky dobře popisují celkovou funkčnost systému. Pro pochopení konkrétních funkcí jsou ale potřeba případy užití (use cases) [22]. K popsání případů užití nám poslouží jazyk UML (Unified Model Language) [23], který se používá pro grafické znázornění struktury různých systémů. Tato notace umožňuje pohled na celou sadu případů užití. Pro jejich správné pochopení je ale důležitý popis každého případu užití, který dává konkrétní informace o dané funkci.

Trochu speciální je případ užití PU29 Vytvořit statistiky o výkonech. Ten se totiž váže k využití uživatelského rozhraní Redmine. Tento postup byl zvolen z toho důvodu, že tento případ užití nebude využíván tak často a Redmine touto funkcí disponuje. Zároveň také tento případ užití bude využívat pouze vedení charity (typicky jeden člověk) a je tak výhodnější zaškolit uživatele než vyvíjet už existující funkci.

Pro modelování případů užití jsem použil kombinaci diagramu a strukturovaného popisu. Diagram odpovídá standardní notaci UML [23] a slouží hlavně pro základní pochopení případů užití, kterých není málo. Hlavní je ale popis jednotlivých případů. Jako první zde je popis samotného scénáře případu užití. Dále jsem popsal pre-conditions a post-conditions, což jsou vlastně podmínky a důsledky případu užití. V popisu pak používám raději zavedené anglické pojmy. Pre-conditions říkají, jaké podmínky musí být splněny, aby bylo možné případ užití použít. Post-conditions zase vyjmenovávají následky po provedení případu užití. Při vytváření této struktury jsem se inspiroval u té, kterou jsem našel v materiálech The University of Texas at Austin [24]. Pomohl mi také popis případů užití od společnosti Wrike [25], která se věnuje řízení projektů.

2.5.1 Role

Systém bude pracovat s různými rolemi podle toho, jaká budou mít zaměstnanci práva pro práci v systému. V aktuální fázi vedení charity požaduje rozdělení na tři role a to: Řadový zaměstnance, Vedoucí služby a Vedení charity. Rozdělení práv na využívání případů užití ještě není zcela jasné a bude se konkretizovat v rámci testovacího provozu. Z tohoto důvodu je tak potřeba, aby při implementaci nebyla práva jednotlivých rolí definována nijak složitě a napevno v rámci zdrojového kódu. Je tak nutné práva definovat v rámci nějaké konfigurace.

V rámci aktuální domluvy platí obecně to, že Řadový zaměstnance by měl mít práva hlavně na vykazování výkonů a jejich editaci. Vedoucí služby poté zodpovídá za celou službu, stará se o aktivity, jejich atributy a podobně. Vedení charity poté může celé služby vytvářet nebo mazat. U vyjmenovaných rolí také platí to, že práva nižší role přebírají automaticky role vyšší. Práva role Řadového zaměstnance tak má i role Vedoucího služby a práva Vedoucího služby zase přebírá role Vedení charity.

U těchto tří rolí dále platí, že jsou vztahovány ke konkrétním službám. Uživatel systému tak může mít k různým službám různá práva. Z tohoto nastavení profituje pracovník tím, že mu na obrazovce nepřekážejí věci, které nepotřebuje. Pro vedení charity je zde přínos v tom, že může lépe kontrolovat, kdo má v systému jaká práva.

V této části je ještě vhodné uvést, jaké role zde nemáme. Dalo by se přemýšlet třeba nad tím, že by měl klient přístup ke svým datům. Tato funkce ale není požadována a systém je čistě interní záležitostí charity. To platí i vzhledem k ostatním uživatelům internetu, kteří nebudou mít do systému přihlašovací údaje a neuvidí tak žádné informace. Toto je důležité i z toho hlediska, aby se k datům klientů nedostal nikdo mimo charitu.

2.5.2 Případy užití role Řadový zaměstnanec

Role Řadového zaměstnance využívá následujících případů užití:

PU1 Přihlásit se do systému

Popis

1. Systém si vyžádá od uživatele uživatelské jméno a heslo.
2. Uživatel vyplní požadované informace.
3. Systém vyhodnotí kombinaci uživatelského jména a hesla a případně uživatele přihlásí.

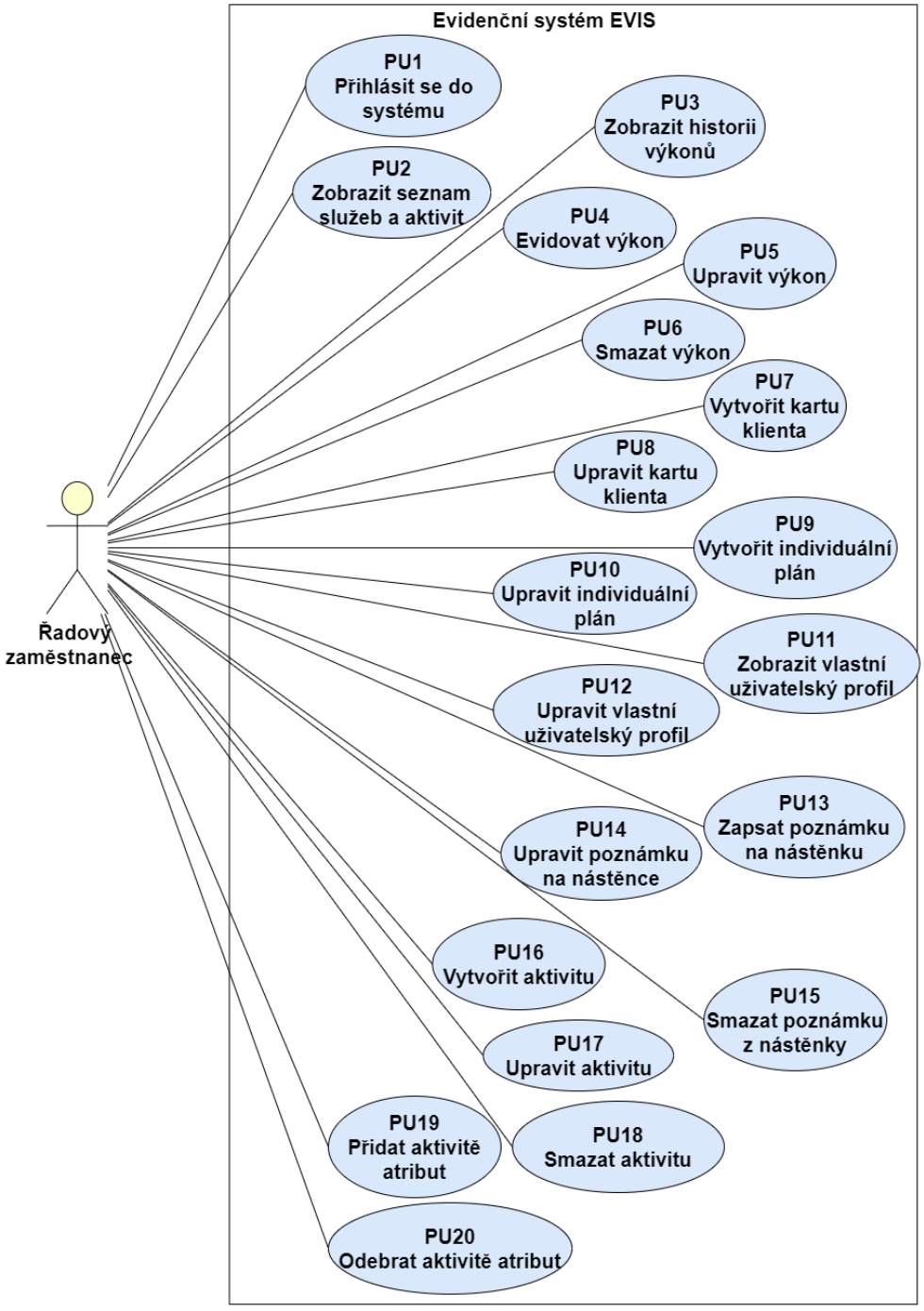
Pre-condition
Žádné

Post-condition
V případě správné kombinace uživatelského jména a hesla je uživatel přihlášen.

PU2 Zobrazit seznam služeb a aktivit

Popis
Uživatel si zobrazí přechodem na hlavní stránku seznam se službami a jejich aktivitami. Zobrazí se jen ty služby, ke kterým má podle svého účtu přístup.

Pre-condition
Uživatel je přihlášen.



Obrázek 2.1: Use case - Řadový zaměstnanec

Post-condition
Žádné

PU3 Zobrazit historii výkonů

Popis
Uživatel si zobrazí aktivitu, ke které chce vidět historii výkonů. Alternativně pak může přes individuální plán klienta některé ze služeb procházet výkony, které se vážou k tomuto plánu. V obou případech pak může uživatel historii výkonů filtrovat nebo řadit dle různých atributů.

Pre-condition
Uživatel je přihlášen.

Post-condition
Žádné

PU4 Evidovat výkon

Popis

1. Uživatel se dostane na stránku s aktivitou, u které chce vykázat výkon.
2. Poté uživatel zvolí možnost pro přidání nového výkonu.
3. Systém uživateli předloží formulář s údaji, které je pro nový výkon dané služby potřeba vyplnit. V rámci tohoto kroku lze vytvořit také vazbu mezi zaměstnance a výkonem nebo klientem (s individuálním plánem) a výkonem.
4. Uživatel formulář vyplní a potvrdí záměr vytvořit výkon se zadanými parametry.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna služba s aktivitou.

Post-condition
V případě validního vyplnění formuláře je vytvořen nový výkon.

PU5 Upravit výkon

Popis

1. Uživatel se dostane na stránku s aktivitou, u které chce editovat výkon.
2. Poté uživatel najde hledaný výkon a zvolí u něj možnost pro editaci.
3. Systém uživateli předloží formulář s údaji, které lze u daného výkonu upravit. Lze zde také upravit vazby výkonu na zaměstnance a klienta s individuálním plánem.

4. Uživatel formulář vyplní a potvrdí záměr upravit výkon se zadanými parametry.

Pre-condition
Uživatel je přihlášen. Je evidovaný alespoň jeden výkon.

Post-condition
V případě validního vyplnění formuláře je upraven existující výkon.

PU6 Smazat výkon

Popis

1. Uživatel se dostane na stránku s aktivitou, u které chce editovat výkon.
2. Poté uživatel najde hledaný výkon a zvolí u něj možnost pro smazání.
3. Smazání by nemělo uživatele zbytečně otravovat potvrzováním. Zároveň je potřeba mít možnost mazaný výkon vrátit zpět.

Pre-condition
Uživatel je přihlášen. Je evidovaný alespoň jeden výkon.

Post-condition
V případě úspěchu operace je výkon smazán.

PU7 Vytvořit kartu klienta

Popis

1. Uživatel si najde službu, kde chce vytvořit kartu klienta a otevře její sekci s klienty.
2. Následně uživatel vybere možnost vytvoření karty nového klienta.
3. Systém uživateli předloží formulář s údaji, které jsou pro vytvoření karty klienta potřeba. Je potřeba také zaručit možnost vytváření anonymních klientů.
4. Uživatel formulář vyplní a potvrdí záměr vytvořit kartu klienta se zadanými parametry.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna služba.

Post-condition
V případě validního vyplnění formuláře je vytvořena nová karta klienta.

PU8 Upravit kartu klienta

Popis

1. Uživatel si najde službu, kde chce upravit kartu klienta a otevře její sekci s klienty.
2. Následně uživatel vybere v seznamu klientů hledaného klienta a zvolí volbu pro upravení.
3. Systém uživateli předloží formulář s údaji, které lze u karty klienta upravit.
4. Uživatel formulář vyplní a potvrdí záměr vytvořit kartu klienta se zadanými parametry.

Pre-condition

Uživatel je přihlášen. Existuje alespoň jedna karta klienta.

Post-condition

V případě validního vyplnění formuláře je upravena existující karta klienta.

PU9 Vytvořit individuální plán

Popis

1. Uživatel si najde kartu klienta, u kterého chce vytvořit nový individuální plán.
2. Poté uživatel vybere možnost pro vytvoření nové plánu.
3. Systém uživateli předloží formulář, kde uživatel vyplní jméno a popis nového plánu.

Pre-condition

Uživatel je přihlášen. Existuje alespoň jedna karta klienta.

Post-condition

V případě validního vyplnění formuláře je vytvořen nový individuální plán.

PU10 Upravit individuální plán

Popis

1. Uživatel si najde kartu klienta, u kterého chce upravit existující individuální plán.
2. Poté uživatel najde v seznamu individuálních plánů ten hledaný a zvolí možnost pro jeho úpravu.
3. Systém uživateli předloží formulář, kde uživatel upraví jméno a popis existujícího plánu.

Pre-condition

Uživatel je přihlášen. Existuje alespoň jeden klientský plán.

Post-condition
V případě validního vyplnění formuláře je upraven existující individuální plán.

PU11 Zobrazit vlastní uživatelský profil

Popis
Uživatel si zobrazí profil s informacemi o své osobě.

Pre-condition
Uživatel je přihlášen.

Post-condition
Žádné

PU12 Upravit vlastní uživatelský profil

Popis

1. Uživatel si zobrazí vlastní profil.
2. Následně uživatel udělá případné úpravy. Vedle běžných informací jako úpravy e-mailové adresy nebo bydliště je zde možné upravit například datum vypršení školení BOZP nebo datum konání školení a kurzů, kterých se uživatel účastnil.
3. Po vyplnění uživatel změny potvrdí.

Pre-condition
Uživatel je přihlášen.

Post-condition
V případě validního vyplnění formuláře je upraven uživatelský profil.

PU13 Zapsat poznámku na nástěnku

Popis

1. Uživatel si otevře virtuální nástěnku.
2. Na nástěnce vytvoří novou zprávu, kterou chce sdílet s ostatními zaměstnanci.

Pre-condition
Uživatel je přihlášen.

Post-condition
V případě validního vyplnění je vytvořena nová poznámka.

PU14 Upravit poznámku na nástěnce

Popis

1. Uživatel si otevře virtuální nástěnku a najde si poznámku, kterou chce upravit.

2. Poté upraví její obsah.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna poznámka na nástěnce.

Post-condition
V případě validního vyplnění je upravena existující poznámka.

PU15 Smazat poznámku z nástěnky

Popis

1. Uživatel si otevře virtuální nástěnku a najde si poznámku, kterou chce smazat.
2. Poté vybere volbu pro smazání této poznámky.
3. Smazání je jednoduché a není komplikované potvrzováním.
4. V případě zvrácení příkazu ke smazání má uživatel možnost vrátit smazanou poznámku zpět.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna poznámka na nástěnce.

Post-condition
Poznámka je smazána.

PU16 Vytvořit aktivitu

Popis

1. Uživatel si najde službu, u které chce vytvořit aktivitu.
2. Následně uživatel zvolí možnost vytvoření nové aktivity.
3. Systém vyzve uživatele k zadání jména a popisu nové aktivity.
4. Uživatel vyplní požadované údaje a potvrdí vytvoření aktivity.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna služba.

Post-condition
Jsou-li vyplněné vstupní informace validní, nová aktivita je vytvořena.

PU17 Upravit aktivitu

Popis

1. Uživatel si najde v rámci služby aktivitu, kterou chce upravit.
2. Na stránce aktivity zvolí možnost pro upravení aktivity.
3. Systém vyzve uživatele k úpravě jména nebo popisu existující aktivity.
4. Uživatel vyplní požadované údaje a potvrdí upravení aktivity.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna aktivita.

Post-condition
Jsou-li vyplněné vstupní informace validní, nová aktivita je vytvořena.

PU18 Smazat aktivitu

Popis

1. Uživatel si najde v rámci služby aktivitu, kterou chce smazat.
2. Na stránce aktivity zvolí možnost pro smazání aktivity, možnost je dostupná jen v případě, když aktivita nemá žádné výkony. Mazání aktivit tak slouží typicky k tomu, když uživatel udělal například při vytváření aktivity nějakou chybu.
3. Systém vyzve uživatele k potvrzení požadavku.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna aktivita, která nemá žádné výkony.

Post-condition
Aktivita je smazána.

PU19 Přidat aktivitě atribut

Popis

1. Uživatel si najde v rámci služby aktivitu, které chce přidat atribut.
2. Na stránce aktivity zvolí možnost pro přidání atributu.
3. Dále může vybrat některý z existujících atributů. Pokud tak učiní, následující bod přeskakuje.
4. Pokud chce uživatel vytvořit nový atribut, zvolí si tuto volbu a systém ho vybídne k vybrání jména atributu, jeho datového typu a toho, jestli má být jeho vyplnění povinné.
5. Výběr následně uživatel potvrdí.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna aktivita.

Post-condition
Aktivita má nový atribut. V případě vytváření atributu vznikl nový atribut.

PU20 Odebrat aktivitě atribut

Popis

1. Uživatel si najde v rámci služby aktivitu, u které chce odebrat atribut.
2. Na stránce aktivity si najde daný atribut a zvolí možnost pro jeho odebrání.
3. Systém si vyžádá potvrzení této akce.

Pre-condition
Uživatel je přihlášen. Existuje alespoň jedna aktivita s uživatelem přidaným atributem.

Post-condition
V případě potvrzení přijde aktivita o atribut.

2.5.3 Případy užití role Vedoucí služby

Role Vedoucího pracovníka služby využívá stejných případů užití jako předchozí role a navíc se k ní váží i následující případy užití:

PU21 Upravit informace o službě

Popis

1. Uživatel si najde službu, kterou chce upravit.
2. Poté uživatel upraví jméno a popis služby.
3. Nakonec změny potvrdí.

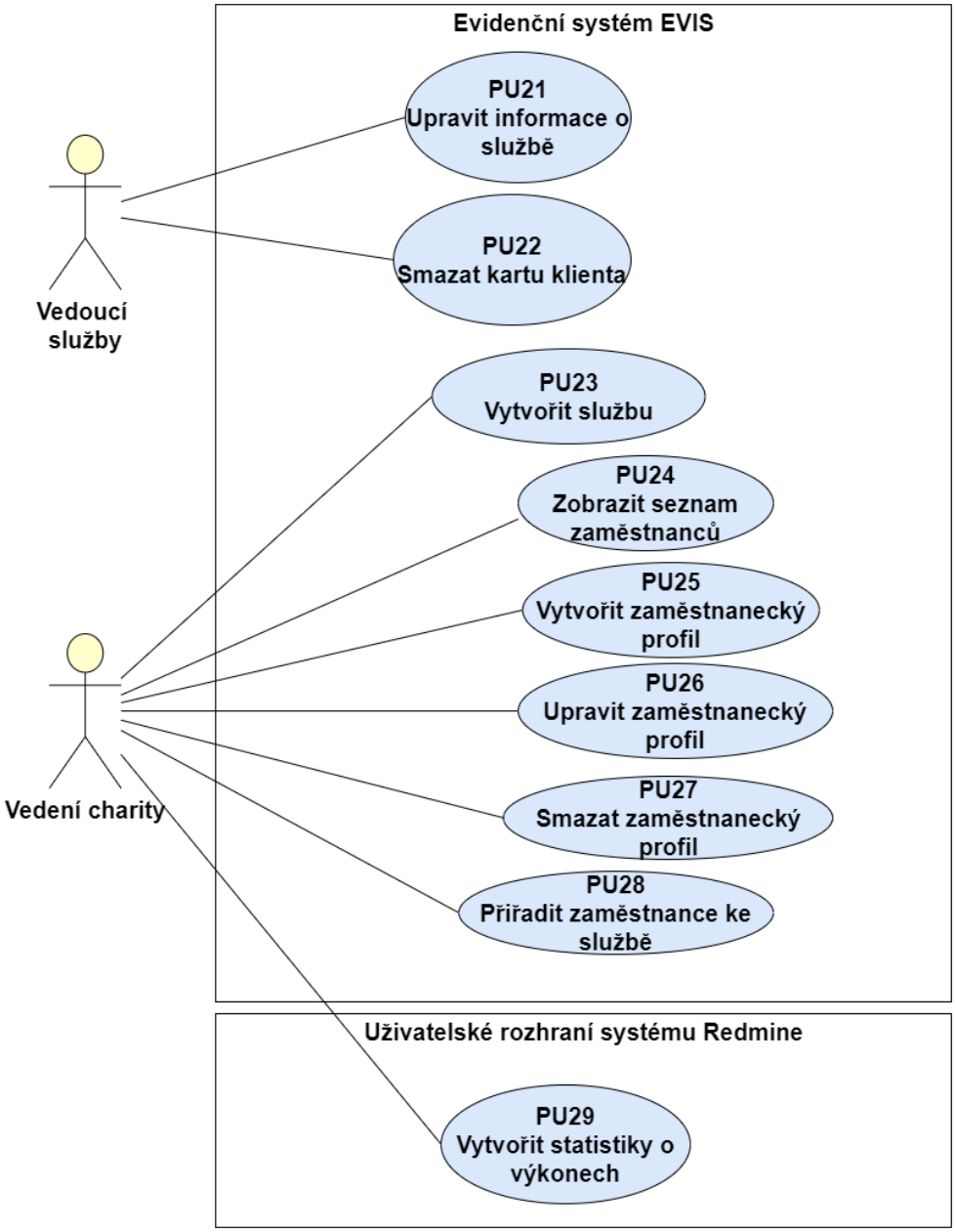
Pre-condition
Uživatel je přihlášen a má příslušnou roli. Existuje alespoň jedna služba.

Post-condition
Informace o službě jsou upraveny.

PU22 Smazat kartu klienta

Popis

1. Uživatel si najde službu, kde chce smazat kartu klienta a otevře její sekci s klienty.
2. Následně uživatel vybere v seznamu klientů hledaného klienta a zvolí volbu pro smazání.
3. Jelikož jde o zásadní akci, tak systém požádá uživatele o potvrzení požadavku. V rámci potvrzení je potřeba uživateli správně prezentovat, kterého klienta by se smazání mělo dotknout.
4. Po potvrzení dojde ke smazání.



Obrázek 2.2: Use case - Vedení

Pre-condition
Uživatel je přihlášen a má příslušnou roli. Existuje alespoň jedna karta klienta.

Post-condition
Karta klienta je smazána.

2.5.4 Případy užití role Vedení charity

Role Vedení charity využívá stejných případů užití jako předchozí role a navíc se k ní váží i následující případy užití:

PU23 Vytvořit službu

Popis

1. Uživatel na obrazovce se službami vybere možnost pro vytvoření nové služby.
2. Systém uživateli nabídne k vyplnění jméno a popis nové služby.
3. Uživatel položky vyplní a potvrdí záměr vytvořit novou službu.

Pre-condition
Uživatel je přihlášen a má příslušnou roli.

Post-condition
Nová služba je vytvořena.

PU24 Zobrazit seznam zaměstnanců

Popis
Uživatel si otevře obrazovku se zaměstnanci a libovolně si může procházet jejich profily. U zaměstnanců, kterým brzy vyprší například zdravotní prohlídka nebo školení BOZP, je poté uživatel na tuto skutečnost upozorněn.

Pre-condition
Uživatel je přihlášen a má příslušnou roli.

Post-condition
Žádná

PU25 Vytvořit zaměstnanecký profil

Popis

1. Uživatel na obrazovce se zaměstnanci vybere možnost přidání nového zaměstnance.
2. Systém uživateli nabídne k vyplnění informace o zaměstnanci.
3. Uživatel položky vyplní a potvrdí záměr vytvořit nový zaměstnanecký profil.

Pre-condition
Uživatel je přihlášen a má příslušnou roli.

Post-condition
Profil zaměstnance je vytvořen.

PU26 Upravit zaměstnanecký profil

Popis

1. Uživatel na obrazovce se zaměstnanci najde profil zaměstnance, který chce upravit a poté zvolí možnost úpravy informací o něm.
2. Systém uživateli nabídne úpravu informací o existujícím zaměstnanci.
3. Uživatel položky vyplní a potvrdí záměr upravit existující zaměstnanecký profil.

Pre-condition
Uživatel je přihlášen a má příslušnou roli. Existuje alespoň jeden zaměstnanecký profil.

Post-condition
Profil zaměstnance je upraven.

PU27 Smazat zaměstnanecký profil

Popis

1. Uživatel na obrazovce se zaměstnanci najde profil zaměstnance, který chce smazat a poté zvolí možnost smazání zaměstnaneckého profilu.
2. Systém si vyžádá potvrzení, protože jde o zásadní a ne tak často používaný příkaz.

Pre-condition
Uživatel je přihlášen a má příslušnou roli. Existuje alespoň jeden zaměstnanecký profil.

Post-condition
Profil zaměstnance je v případě potvrzení odstraněn.

PU28 Přiřadit zaměstnance ke službě

Popis

1. Uživatel na obrazovce se zaměstnanci najde profil zaměstnance, kterému chce přiřadit členství ve službě.
2. Poté uživatel zvolí možnost přidání nové vazby zaměstnance ke službě.
3. Systém následně nabídne seznam služeb a rolí.

4. Uživatel vybere kombinaci služby a role a potvrdí svůj výběr.

Pre-condition
Uživatel je přihlášen a má příslušnou roli. Existuje alespoň jeden zaměstnanecký profil. Existuje alespoň jedna služba.

Post-condition
Profil zaměstnance má nové členství ve službě a může v ní konat dle přiřazených práv.

PU29 Vytvořit statistiky o výkonech

Popis

1. Příslušný zaměstnanec se přihlásí do systému uživatelského rozhraní systému Redmine.
2. Otevře si obrazovku s výkony (v Redmine terminologii úkoly) a pomocí filtrů, řazení a souhrnů získá potřebné statistiky.

Pre-condition
Uživatel je přihlášen a má příslušnou roli.

Post-condition
Žádná

2.6 Slovník

David v jeho diplomové práci [1, s. 10-11] pracoval se slovníkem pojmů. Tento slovník je dobrý pro interakci se zadavatelem. Pro vlastní účely jsem si ho ale ještě rozšířil o české a anglické názvy, které jednotlivým slovům odpovídají v rámci Redmine terminologie. Toto rozšíření (obrázek 2.3) nemá pro zadavatele zásadní přímý přínos, ale při implementaci může dost pomoci při orientaci ve značení. Potřeba české i anglické verze plyne z toho, že API využívá anglické pojmy a uživatelské prostředí Redminu se bude využíváno v češtině. Využití API podrobněji popíši později.

Terminiologie charity	Redmine - česky	Redmine - english
Služba	Projekt	Project
Aktivita	Fronta	Tracker
Vlastnosti aktivity/zaměstnance/klienta	Uživatelská pole/atributy	Custom fields/attributes
Výkon	Úkol	Issue
Historie	výčet Úkolů	list of Issues
Individuální cíl	Úkol	Issue
Zaměstnanec	Uživatel	User
Klient	Úkol	Issue

Obrázek 2.3: Rozšířený slovník pojmů

Kapitola 3

Návrh

3.1 Využití Redminu

Z analýzy tedy vyplynulo, že v nějaké podobě se pro projekt využije systém Redmine. Teď je ale důležité rozhodnout to, jakým způsobem samotné zapojení provést. Možností je totiž hned několik.

První možností by bylo upravit existující prostředí Redminu. To se ukázalo jako neprůchodné vzhledem k tomu, že v jinak funkčním systému by se muselo dělat mnoho změn a kód není moc dobře zdokumentován. Dlouze bychom tedy zjišťovali vůbec to, jak spolu jednotlivé části komunikují a podobně. V rámci úprav Redminu se nabízela ještě možnost využít toho, že na systém lze aplikovat různá témata [12]. U těch je ale problém, že bezplatná témata přinášejí většinou jen málo změn a soustředí se hlavně na změny barev, ale upravené uživatelské rozhraní se téměř neliší od původní verze. Placená témata překreslují původní vzhled často docela radikálně a mění jej v modernější. Systém je po jejich aplikaci většinou přehlednější, ale více se zaměřuje na správu projektů, což není směr, který by nám vyhovoval.

Dalším směrem, kterým by se mohl projekt ubírat, je využití REST API. To umožňuje vytvořit nad Redminem novou aplikaci, která bude používat data a funkce z Redminu. Výhodou tak je, že použijeme jen to, co potřebujeme a nepotřebné věci nebudou dělat pro uživatele práci v systému nepřehlednou. Otázkou pak také je, jestli API nabízí všechny potřebné funkce a některé tak naopak nechybí. Stav přístupu k jednotlivým objektům jako jsou projekty, úkoly nebo fronty je vidět v přehledné tabulce na stránkách Redminu [8]. API tak možná bude potřeba rozšířit a tomuto problému se budu dále věnovat v implementační části.

Systém Redmine řeší také potřebu získávat různá souhrnná data o výkonech. David se ve své práci [1, s. 63-73] věnoval také části, kde vytváří vlastní návrh nástroje pro vypočítávání těchto údajů. Redmine ale umí stejné informace o výkonech získat sám. Jelikož se tato funkce výsledného řešení nebude využívat na denní bázi a statistiky bude generovat pouze vedení charity,

ukázalo se jako výhodnější řešení využít pro tento účel rozhraní Redminu, které statistické údaje o výkonech umí spočítat také.

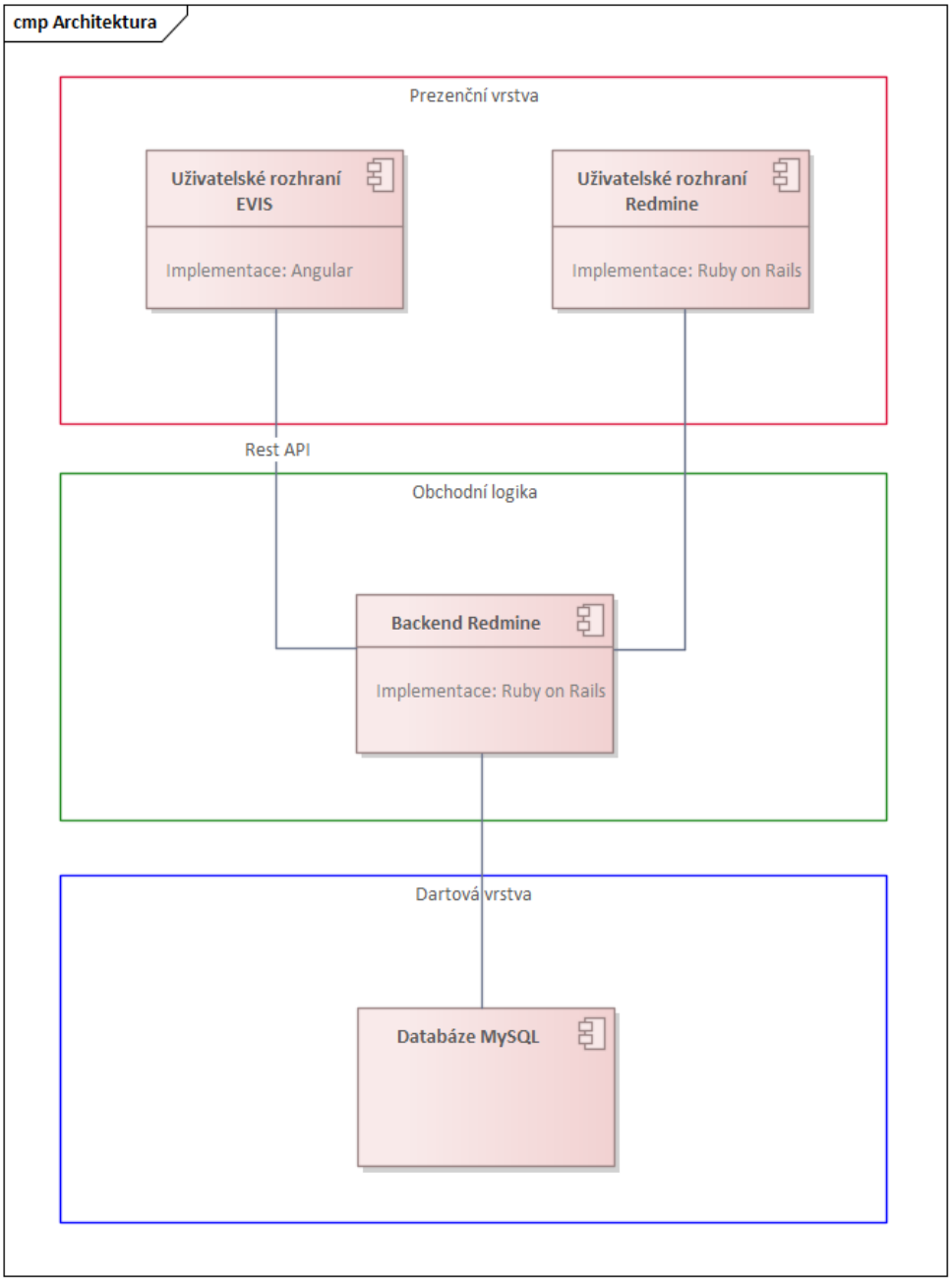
3.2 Architektura

Davidův návrh architektury jsem zmínil už výše. Nový návrh (na obrázku 3.1) bude vzhledem k zapojení Redminu přímočařejší, protože tento hotový systém obsahuje již sám některé části. Redmine jako samostatný systém je postaven na dvouvrstvé architektuře [26]. Je zde samostatná datová vrstva a poté sdružená vrstva pro obchodní logiku a prezentační vrstvu. Logiku i prezentaci zajišťuje jeden Ruby on Rails projekt, který bude v rámci implementace drobně upraven. Navíc zde ale máme zcela novou aplikaci v prezentační vrstvě a tím je nová webová aplikace vyvíjená v Angularu, které jsme dali označení Evis (ze spojení Evidenční systém). Na obrázku tedy zobrazuji třívrstvou architekturu a Redmine zde pro přehlednost rozděluji do dvou vrstev, i když se stále jedná o jeden projekt.

Redmine poskytuje hned několik možností pro výběr poskytovatele databáze. Tento systém poskytuje nativní podporu pro: MySQL [27], PostgreSQL [28], Microsoft SQL Server [29] a SQLite [30]. Všechny databáze se řadí mezi relační databáze [31]. V tomto typu databází jsou data uložena v tabulkách a jednotlivé tabulky jsou spolu propojeny podle pevného relačního schématu. Dále například u SQLite je problém s použitím pro více uživatelů, kteří by chtěli v jeden moment zapisovat data. Sám poskytovatel této databáze uvádí [32], že pro toto použití se databáze nehodí. Běžně ji ale můžeme najít třeba v některých mobilních aplikacích, kde slouží jako lokální úložiště. Využití si pak také často najde při testování.

Microsoft SQL Server není podporován nativně přímo v Ruby on Rails, což může být do jisté míry nevýhoda. Dále jsem se snažil rozhodovat také podle rychlosti jednotlivých řešení. Nakonec jsem našel porovnání [33] všech databází v Redminu 3.3 (dnešní aktuální verze je 4.2). Z porovnání vychází, že rozdíly mezi jednotlivými databázemi nejsou většinou velké. Nejrychlejší SQLite jsem vyřadil už kvůli komplikovanější práci více uživatelů. MySQL a PostgreSQL se drží na podobných hodnotách a s odstupem je poslední Microsoft SQL Server. Nakonec jsem se rozhodl pro databázi MySQL, kterou tvůrci Redminu uvádí v konfiguraci jako výchozí možnost a ve výčtech je vždy uváděna na prvním místě. Nic ale nebrání tomu, aby se později poskytovatel databáze změnil, jen by bylo potřeba migrovat data.

Backend zde zastává Redmine, který se stará o připojení k databázi a připravení dat pro frontend. Tato část má na starosti zpracovávat požadavky z frontendu, vyhodnotit je a poslat je do databáze. Při odpovědi frontend informuje o tom, jak byl požadavek vyhodnocen. K vyhodnocení požadavků se berou v potaz také práva uživatelů a Redmine tak může stejný požadavek od jednoho uživatele zamítnout a od druhého provést.



Obrázek 3.1: Architektura systému

Poté zde máme naši frontendovou aplikaci, která slouží jako zjednodušené rozhraní místo Redminu. Po funkční stránce tak bude tato aplikace poskytovat funkce pro každodenní vykazování výkonů a podobně. Podrobněji se této části budu věnovat později, protože zde velkou roli hraje návrh uživatelského rozhraní, které je dost důležitou součástí projektu a zaslouží si vlastní část. Souhrnná statistická data jsou pak generována přímo v rozhraní Redminu. Hlavní důraz je ale kladen na novou frontendovou aplikaci, v které budou uživatelé trávit drtivou většinu času.

3.3 Uživatelské rozhraní

Samotné uživatelské rozhraní vlastní aplikace vychází z prototypu, napsaném ve frameworku Angular [2]. Tento prototyp využívá grafických prvků Material designu od Googlu a díky tomu by mělo být rozhraní konzistentní a jednoduché. Před implementací je ale ještě dobré detailně navrhnout úpravy v uživatelském rozhraní, což je jedna z hlavních částí implementace. Druhou částí je napojení uživatelského rozhraní na systém Redmine, což Hi-fi prototyp v aktuální chvíli vůbec neumí.

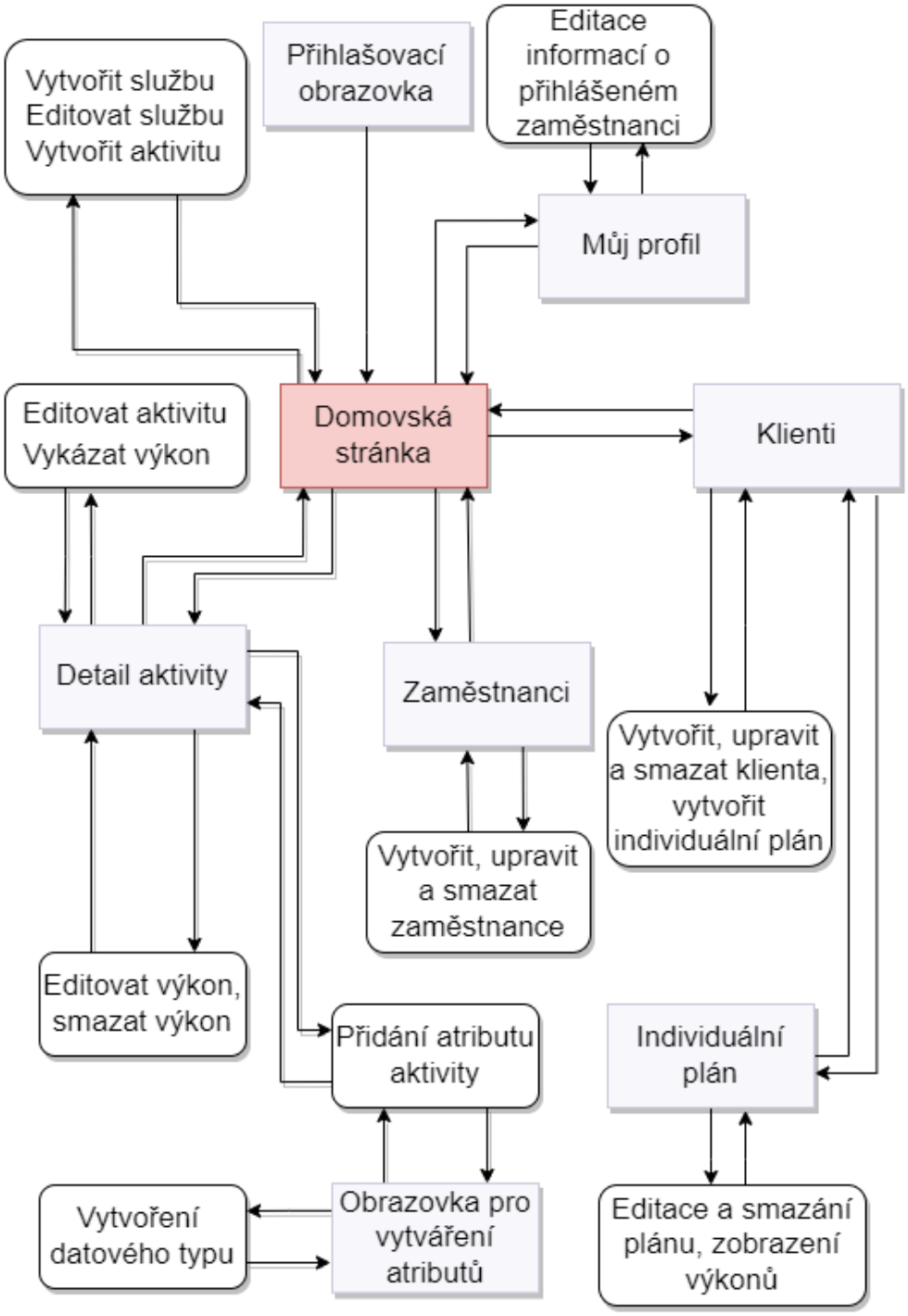
3.3.1 Task graph

Uživatelské rozhraní slouží jako prostředek přístupu uživatelů k vykonávání funkčních požadavků 2.4.2. Před návrhem samotného vzhledu a kompozice prvků na jednotlivých stránkách je dobré si udělat takzvaný Task graph. Tento diagram slouží k rozdělení systému na obrazovky, na kterých provádí uživatel požadované úkony. Vedle toho jsou zde zachyceny přechody mezi obrazovkami, které by ideálně měly co nejvíce odpovídat pracovnímu postupu uživatelů. Task graph vytvářeného systému ukazuje obrázek 3.2.

Hlavní obrazovkou je zde takzvaná *Domovská stránka*, do které se uživatel dostane po přihlášení z *Přihlašovací obrazovky*. Přímo na *Domovské stránce* pak může provádět úkony jako vytvoření a editace služeb nebo vytvoření aktivity. Hlavně ale tato stránka slouží jako takový rozcestník. Lze se odsud tedy dostat na obrazovky *Detail aktivity*, *Zaměstnanci*, *Klienti* nebo *Můj profil*. Tyto stránky pak mají další funkce a přechody, jak je zobrazeno na grafu.

3.3.2 Prototypy

Jelikož už mám pojmenované jednotlivé obrazovky a přechody, tak dalším krokem by mělo být prototypování. Výsledný prototyp slouží v komunikaci se zadavatelem, který na něm může poukázat na věci, které by rád viděl zpracované jinak. Pro mě pak prototyp bude užitečný pro samotnou implementaci, kdy ho použiji jako určitou osnovu a budu podle něj navrhovat jednotlivé stránky. Prototypy můžeme rozdělit do dvou kategorií a to Lo-fi (low-fidelity) a Hi-fi (high-fidelity) prototyp [34]. Tyto kategorie se mezi sebou liší úrovní



Obrázek 3.2: Task graph

přesnosti (fidelity). Zvolení přesnosti souvisí také s tím, kolik času budeme výrobě prototypu věnovat.

Lo-fi prototypy [35] často trvá vyrobit pouze několik hodin nebo i méně. Tento typ prototypu se často provádí skrze takzvaný papírový model, kde už název prezentuje, o co se jedná. Jde tedy o docela jednoduchý model, který si lze klidně nakreslit tužkou na papír. Při zapojení počítače je pak výhodou snadná upravitelnost. V rámci komunikace se zadavatelem si zde lze vyjasnit první nepřesnosti a tento typ návrhu pomůže i v týmu. Člověk má také často pocit, že má návrh perfektně vytvořený v hlavě, ale když dojde na další kroky, tak narazí na problémy, které by Lo-fi prototyp odhalil dříve. Na této úrovni prototypování se také lépe navrhuje celkové rozložení stránek, protože zde nejsou některé rušivé elementy jako animace, barvy, různé styly písma a podobně.

Původní Lo-fi prototyp jsme spolu s kolegy z týmu v předmětu NI-NUR vytvořili v nástroji Balsamiq [36]. Nejde o jediný nástroj, který se tvorbou podobných prototypů zabývá. Z konkurenčních řešení tak mohu zmínit například nástroj Figma [37]. Pro tvorbu Lo-fi prototypu pro tento projekt nebylo potřeba nějakých speciálních funkcí. Myslím si, že by oba nástroje posloužily stejně dobře. Velká část Lo-fi prototypu už byla hotová z práce ve výše jmenovaném předmětu. Já jsem poté dodělal obrazovky se zaměstnanci, klienty, klientskými plány, správou vlastního účtu nebo přihlašováním. Tyto obrazovky zpracovávají funkce uvedené v Task graphu 3.2. Celý papírový model je uveden v příloze [Appendix A].

Hi-fi prototyp [38] je na druhou stranu často téměř k nerozeznání od výsledného produktu. Jeho účelem je ukázat výsledné uživatelské rozhraní. Jsou zde tedy všechny grafické prvky včetně barev, stylů písma, přechodů a podobně. V této fázi si tak nevystačíme s tužkou a papírem, ale je potřeba využít služeb počítače nebo jiného zařízení, pro který je výsledná aplikace připravována. Rozdílem oproti výsledné aplikaci je zde hlavně nepřítomnost čehokoliv na pozadí uživatelského rozhraní. Systém tak nijak speciálně nepracuje s vkládanými daty a maximálně si udržuje aktuální stav, který se po opětovném načtení smaže. Na Hi-fi prototypu si lze ujasnit detaily, které by se měly změnit do finální aplikace. Typicky se zde už nejedná o koncepční změny, ale cílem změn jsou úpravy barev, velikosti jednotlivých prvků nebo animací.

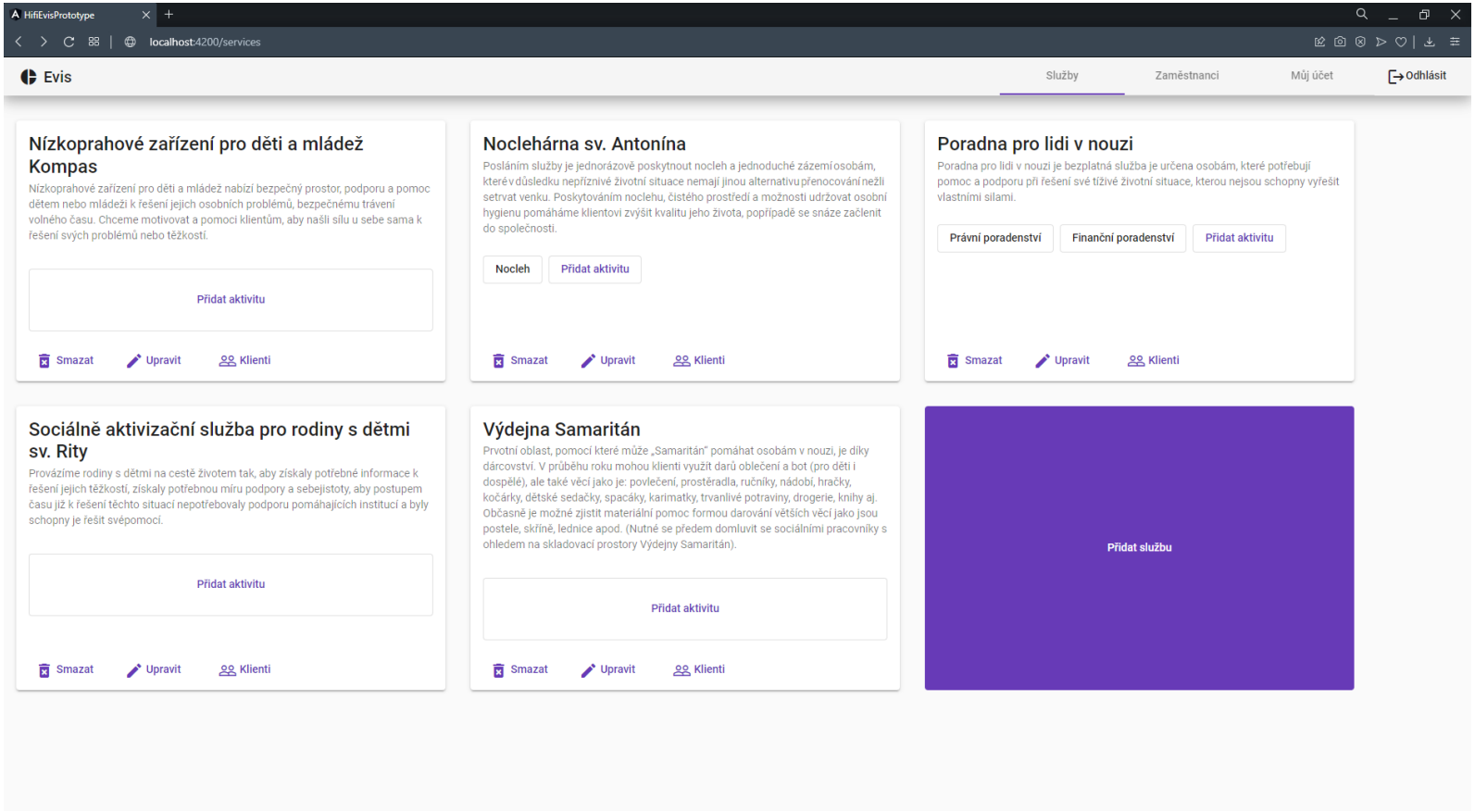
Prototyp s vyšší úrovní přesnosti můžeme navrhovat v různých nástrojích jako Axure, Adobe XD, Pixate a podobně. My jsme se ale ve stejném týmu jako u návrhu Lo-fi prototypu rozhodli navrhnout Hi-fi prototyp rovnou v programovacím frameworku Angular [2]. Pro využití tohoto nástroje v implementaci samotné webové aplikace jsme byli domluveni už předtím a tak nám přišlo praktické Hi-fi prototyp následně využít právě v implementační fázi.

V rámci této práce jsem se rozhodl na existující Hi-fi prototyp navázat z několika důvodů. Zřejmě největším z nich je to, že prototyp prošel uživatelským testováním, kterého se účastnili také zaměstnanci charity, a David ho popisuje v jeho diplomové práci [1, s. 79-82]. Vedle uživatelského testování

jsme provedli heuristickou analýzu, z jejichž závěrů lze vycházet pro další vývoj. Dalším důvodem je to, že na vývoji tohoto prototypu jsme se sám podílel a nebude pro mě tedy nijak složité vyznat se ve zdrojovém kódu a návrhu jednotlivých funkcionalit. Hi-fi prototyp jsem o nové obrazovky rozvíjel vždy před samotnou implementací. Prototyp je víceméně vizuálně totožný s uživatelským rozhraním výsledné aplikace, kterou představím v následující části s implementací. Pro demonstraci zde uvádím vzhled obrazovky s výpisem služeb charity 3.3.

3.3.3 Názvosloví

Už zde bych chtěl alespoň krátce zmínit jednu velmi důležitou věc, kterou jsme si odnesli z testování prototypu. Pro pohodlné používání je kriticky důležité, aby uživatelé rozuměli tomu, jak s nimi systém komunikuje. Když se dopustím určitého zobecnění, tak zaměstnanec charity typicky nerozumí pojmům jako atribut nebo datový typ. V rámci implementace tak bude velmi důležité zvolit pro cílovou skupinu srozumitelné popisky. Z části se budu snažit v této oblasti vylepšovat aktuální stav průběžně i podle výstupů ze schůzek se zadavatelem. Je ale také dost možné, že některé nedokonalosti se v tomto ohledu odhalí až v testování.



Obrázek 3.3: Ukázka Hi-fi prototypu

Kapitola 4

Realizace

4.1 Úprava Redminu

Realizaci tohoto projektu lze rozdělit do dvou částí. Tou první je úprava Redminu pro naše potřeby. Tato část bude v rámci realizace tou menší, ale pro následné fungování je zásadní. Vycházel jsem z Redminu ve verzi 4.2.3 [39].

4.1.1 Vylepšení API

Jak jsem již několikrát zmiňoval, Redmine je napsaný ve frameworku Ruby on Rails [5]. S tím jsem před touto prací neměl žádnou zkušenost a tak jsem k úpravám přistupoval dost opatrně. Další překážkou byla ne příliš dobrá dokumentace Redminu, kdy třeba v kódu jsou komentáře docela vzácné. Naštěstí je kód z velké části snadno pochopitelný. K tomu přispívá i použitý framework, který je přímo uzpůsoben k psaní API. Hlavními složkami, které mě zajímaly, byly *models*, *controllers* a *views*. V první jmenované jsou definovány kostry objektů jako projekt, úkol, uživatel a podobně. *Controllers* poté odpovídají za zpracování požadavků skrze vlastní webové rozhraní nebo skrze API. *Views* nakonec definují, jak bude vlastní webové rozhraní vypadat.

Cílem mých zásahů tak byly hlavně třídy v *controllers*, kde jsou funkce jako *index* (vrátí posloupnost objektů daného typu) *create* (vytvoří nový objekt) *update* (upraví existující objekt) nebo *destroy* (smaže existující objekt). Pro použití tohoto projektu byl nedostatečný přístup k objektům Custom_field a Tracker. U objektu User pak bylo pouze potřeba upravit práva na přístup k datům tohoto typu. Ve výchozím stavu totiž smí seznam všech zaměstnanců zobrazovat jen uživatel s administrátorskými právy, což v tomto projektu nestačí.

Při rozšíření funkčnosti pro přístup skrze API bylo vždy potřeba algoritmus rozdělit na větev pro standardní rozhraní Redminu (v kódu jako *format.html*) a na větev pro požadavky z API (v kódu jako *format.api*). Rozdíl v základní a vylepšené verzi pak lze vidět na obrázcích 4.1 a 4.2.

```
def destroy
  @tracker = Tracker.find(params[:id])
  unless @tracker.issues.empty?
    flash[:error] = l(:error_can_not_delete_tracker)
  else
    @tracker.destroy
  end
  redirect_to trackers_path
end
```

Obrázek 4.1: Původní verze funkce destroy v třídě TrackersController

```
def destroy
  @tracker = Tracker.find(params[:id])
  unless @tracker.issues.empty?
    flash[:error] = l(:error_can_not_delete_tracker)
  else
    @tracker.destroy
  end
  respond_to do |format|
    format.html {redirect_to trackers_path}
    format.api {render_api_ok}
  end
end
```

Obrázek 4.2: Vylepšená verze funkce destroy v třídě TrackersController

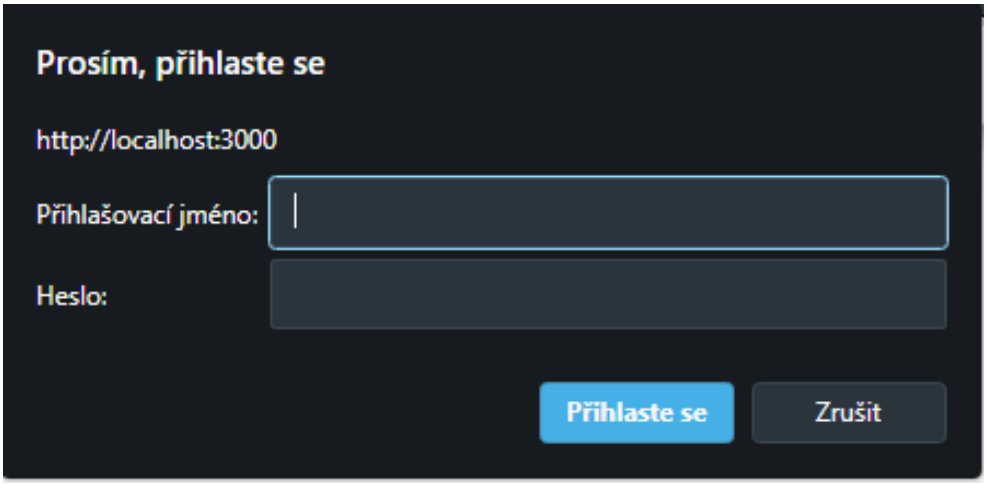
Konečně aby bylo výše popsané rozšíření opravdu dostupné, bylo potřeba přidat cesty k jednotlivým funkcím. To jsem provedl v souboru *config/routes.rb*, kde jsem provedl úpravy, které demonstruje obrázek 4.3

4.1.2 Technické úpravy

Další úpravy byly ještě drobnější, ale jejich využití bylo také důležité. Zmíním hlavně HTTP hlavičku WWW-Authenticate [40], kterou Redmine používá. Tato hlavička se vyšle spolu s chybovým kódem 401 Unauthorized v té chvíli, když se nepodaří ověřit přihlašovací údaje uživatele. Problém v tomto případě ale je s tím, že prohlížeč hlavičku WWW-Authenticate promítne tak, že vyhodí vlastní dialog pro získání přihlašovacích údajů. To může být v mnoha případech příjemným bonusem. Zde jsem však pro tento dialog nenašel využití

```
  get 'trackers', :controller => 'trackers', :action => 'inde
x'
  post 'trackers', :controller => 'trackers', :action =>
  ↪  'create'
  get 'trackers/:id', :controller => 'trackers', :action =>
  ↪  'show'
  delete 'trackers/:id', :controller => 'trackers', :action
  ↪  => 'destroy'
  put 'trackers/:id', :controller => 'trackers', :action =>
  ↪  'update'
```

Obrázek 4.3: Úprava cest v config/routes.rb

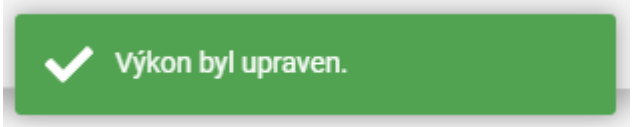


Obrázek 4.4: Ukázka výchozího přihlašovacího dialogu při špatných přihlašovacích údajích

a tedy jsem jej odstranil a nahradil vlastním řešením, které mi přijde uživatelsky přívětivější. Podobu původního řešení lze vidět na obrázku 4.4, moje řešení pak probírám v následující části, kde se věnuji implementaci nového uživatelského rozhraní Evis.

4.2 Nové uživatelské rozhraní Evis

V této části bych se chtěl zaměřit na nové uživatelské rozhraní, které budou využívat zaměstnanci charity v rámci běžných pracovních procesů. Jedná se o webovou aplikaci, která je napsaná v Angularu [2]. Jak jsem psal už



Obrázek 4.5: Notifikace - úspěch

v Nefunkčních požadavcích 2.4.1, aplikace musí mít přehledné a jednoduché uživatelské rozhraní, což je myšlenka, které jsem se snažil držet v rámci celé implementace. Pro lepší pochopení aplikace v následujících částech popíši jednotlivé obrazovky spolu s obrázky aplikace.

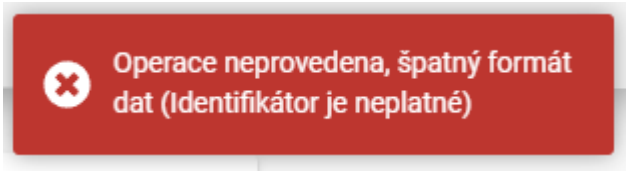
4.2.1 Výběr knihoven

V dnešní době existuje snad na každou složitější věc celá skupina knihoven. Problémem často není, jestli na danou věc existuje knihovna, ale kterou vybrat. Pro celkovou srozumitelnost je důležité jednotné grafické téma. Já zde používám styl Material Design a většina uživatelských prvků je přímo z této knihovny [41].

Chápání vzhledu je z velké části subjektivní záležitost. Zaprvé mám s tímto stylem sám pozitivní zkušenost z dřívějších projektů nebo z používání v telefonu s operačním systémem Google Android. Zadruhé za Material Designem stojí Google stejně jako za frameworkem Angular a tak už mi intuitivně dává smysl, že právě tento styl by nemusel být špatnou volbou. Zatřetí jde poté o styl jednoduchý a srozumitelný, což jsou požadované vlastnosti celé aplikace. Procházel jsem si i další designové styly jako Bootstrap, Vaadin, Clarity a další [42]. Většinou jsem nenašel nějaké zásadní nedostatky, které by dělaly jiný design nepoužitelným pro tento projekt. Vzhledem k předchozím důvodům jsem ale vybral Material Design.

Ne vše mi ale u standardních komponentů této knihovny vyhovovalo. Jako komponenty se ve světě Angularu označují různé prvky uživatelského rozhraní. V mém případě byl problém u tabulky. Tabulka je z mého pohledu velmi důležitým prvkem velké spousty projektů. Tabulky různých knihoven umí velkou spoustu věcí jako řazení, filtrování, stránkování, exportování, načítání na straně serveru a mnoho dalších. Dalším aspektem pro výběr té pravé knihovny je samotná práce ve zdrojovém kódu a zde je to zase spíš individuální.

Prošel jsem si několik z těch nejznámějších možností [43] a nakonec mě nejvíc zaujala tabulka z knihovny PrimeNG [44]. Tato tabulka mi vyhovovala z pohledu funkcí. Je zde podpora filtrování i řazení dat přímo na straně serveru při použití stránkování. Filtrování a řazení si lze pak poměrně jednoduše do značné míry upravit. Důležité pro tento výběr také bylo to, že tabulka od PrimeNG podporuje mino jiné i Material Design a nejde tak o komponent, který by do uživatelského rozhraní graficky neladil.



Obrázek 4.6: Notifikace - neúspěch

4.2.2 Notifikace a načítání

Jeden z rozdílů oproti Hi-fi prototypu je zavedení notifikací a zobrazení načítání. Tyto funkce jsou důležité kvůli tomu, že odpověď z Redminu nemusí být hned dostupná a dokonce může přijít negativní reakce s tím, že se operace nezdařila. Po odeslání požadavku na server tak uživatele informuji standardním načítacím proužkem u horní strany obrazovky. V případě přijetí odpovědi proužek zmizí. Odpověď pak může být taková, že uživatelský požadavek byl splněn. Poté ukáže systém zelenou notifikaci (obrázek 4.5) spolu se správou, co se vlastně stalo.

Na druhou stranu ale může dojít i k širokému spektru problémů. Může spadnout server, posílaná data nejsou v požadovaném formát a podobně. Redmine v rámci chybové odpovědi zároveň vrací i důvod(y) nezdaru požadavku. Já informaci o neúspěšném zpracování prezentuji jako červenou notifikaci (obrázek 4.6), kde je napsáno, co se stalo a v závorce pak uvádím ještě zprávy od Redminu. Ty často nejsou tak uživatelsky přívětivé, ale je to lepší než žádná informace.

Zpracování chyb jsem původně řešil na úrovni jednotlivých funkcích, které skrze HTTP požadavky volají funkce z Redmine API. Později jsem ale vše přepracoval na HTTP Interceptor (obrázek 4.7), skrze který jdou všechny požadavky. V případě chyby se nejdříve pokusí systém požadavek zopakovat, jestli ani to nevyjde, vypíše chybovou hlášku. K zobrazení chyby ale může dojít ještě jinak. Jde o ten případ, když se vrátí data, ale v aktuálním kontextu jsou například neplatná. To už si ale jednotlivé komponenty řeší samy.

4.2.3 Přihlašovací obrazovka

Nejdříve se uživatel setká s přihlašovací obrazovkou (obrázek B.1). Hlavním účelem je zde přihlásit uživatele do systému pomocí kombinace jeho uživatelského jména a hesla. Navíc je zde ještě krátké označení toho, na které webové stránce vlastně uživatel je. V případě zapomenutí nebo ztráty hesla je pak možné heslo obnovit. Samotné obnovení hesla probíhá skrze rozhraní Redminu, kde uživatel vyplní e-mailovou adresu, která je svázána s uživatelským účtem. Po chvíli obdrží e-mail s odkazem, kde si heslo obnoví.

```
export class HttpErrorInterceptor implements HttpInterceptor {
  constructor(private injector: Injector, private
  ↪  notificationService: NotificationService) { }

  intercept(request: HttpRequest<any>, next: HttpHandler):
  ↪  Observable<HttpEvent<any>> {
    return next.handle(request)
      .pipe(
        retry(1),
        catchError((error: HttpErrorResponse) => {
          let errorsString = "";
          if(error.error && error.error.errors) {
            errorsString += " (" + (error.error.errors as
            ↪  string[]).join(", ") + ")";
          }

          switch (error.status) {
            case 401: {
              this.notificationService.showError(config.error
              ↪  401 + errorsString);
              break;
            }
            case 403: {
              this.notificationService.showError(config.error
              ↪  403 + errorsString);
              break;
            }
            case 422: {
              this.notificationService.showError(config.error
              ↪  422 + errorsString);
              break;
            }
            default: {
              this.notificationService.showError(config.error
              ↪  Default + errorsString);
            }
          }
        })
      )
  }
}
```

Obrázek 4.7: HTTP error interceptor

4.2.4 Domovská stránka a menu

Tuto obrazovku (obrázek B.2) uživatel vidí hned po přihlášení a je hlavním rozcestníkem pro pohyb v rámci celé aplikace. Největší prostor zde zabírají karty s jednotlivými službami, což jsou části charity, které se věnují vždy určitému typu pomoci. Uživatel vždy vidí jen ty služby, které mu byly vedením přiřazeny a nemá zde tak něco, co by ke své práci nepotřeboval. Každá karta má v sobě název a popis dané služby. Poté je zde seznam aktivit, které se v rámci služby evidují. Nakonec jsou u každé karty i tlačítka pro smazání a úpravu služby nebo tlačítko pro přesměrování na obrazovku s klienty dané služby. Na konci seznamu služeb je pak tlačítko pro vytvoření nové služby.

Na této stránce je poprvé vidět horní menu, které uživatele doprovází celým prostředím (kromě přihlašovací obrazovky). Vlevo je logo systému, které standardně funguje jako odkaz na domovskou obrazovku. V pravé části zde pak jsou odkazy na různé obrazovky systému a tlačítko k odhlášení. Vedle funkce přesunu na jinou stránku slouží menu také jako pomocník při orientaci tím, že je vždy zvýrazněna stránka, na které uživatel je.

Z pohledu napojení na Redmine zde používám projekty jako služby v terminologii charity. Aktivity jsou poté reprezentovány pomocí front. Funkce vytvoření nové nebo úprava existující služby jsou zprostředkovány skrze jednoduchý dialog, kde uživatel vyplní název a popis služby. U mazání služby jsem přišel s řešením potvrzovacího dialogu, které může být sice zdržující a otravné, ale v případě mazání služby jde o dost zásadní věc, která si zaslouží potvrzení uživatelem. Mazání není však trvalé a v rámci Redmine terminologie je projekt pouze archivován. To znamená, že jej lze obnovit, ale v prostředí Evisu už se nebude služba nikde zobrazovat. Zrušení archivace je tak možné jen skrze rozhraní Redminu.

4.2.5 Detail aktivity

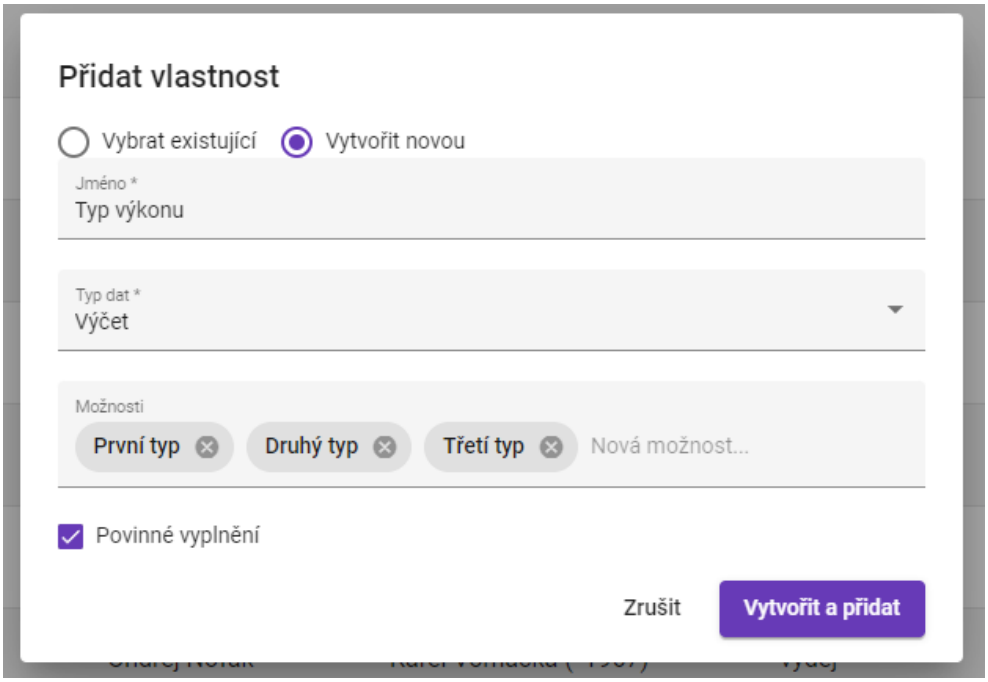
Na této obrazovce (obrázek B.3) by měl uživatel trávit velkou část doby, kdy bude systém používat. Na levé straně jsou informace o aktivitě. Editovat zde lze jméno a popis aktivity. Aktivitu tu lze také mazat. Mazání je ale dostupné jen v případě, že aktivita nemá žádné výkony. Proces práce s aktivitou totiž zahrnuje mazání typicky jen ve chvíli, kdy jí někdo vytvoří například omylem. Kdyby tato možnost byla dostupná i u aktivity s výkony, mohlo by dojít ke ztrátě cenných dat. Samozřejmě toto chování bude ještě prověřeno v rámci uživatelského testování.

Pod základními informacemi máme seznam vlastností aktivity. Už samotné slovo vlastnost je zde zajímavé. V původním návrhu jsem totiž pracoval s označením atribut aktivity. Toto označení mi přijde lépe popisující, o jakou věc jde. V rámci testování Hi-fi modelu jsme ale s týmem z předmětu Ni-NUR došli k tomu, že slovo atribut není dobře přijímáno cílovou skupinou, která bude program používat. Označení vlastnost by tak mělo být v tomto

ohledu lepší.

Zpět ale k samotným vlastnostem. Pro jejich pochopení si musíme vysvětlit některé souvislosti. Aktivita slouží v našem prostředí jako takový vzor pro výkony. Určuje tak, jaké vlastnosti (atributy) bude daná skupina výkonů mít. Vlastnosti můžeme rozdělit na výchozí a uživatelsky vytvořené. Výchozí vlastnosti jsou pro všechny aktivity stejné a jsou jimi: Datum, Pracovník, Klient, Předmět a Doba trvání. Těchto pět vlastností bylo po konzultacích s vedením charity zvoleno jako jakési minimum, které má mít každá aktivita a tedy i každý výkon.

Dále jsou zde vlastnosti uživatelsky vytvořené (obrázek 4.8). Zde si může uživatel vybrat mezi již existující vlastností (využívá se třeba u jiné aktivity) nebo vytvořit vlastnost novou. Vlastnosti podporují různé datové typy. Bohužel zde je opět trochu problém s názvoslovím a pojem datový typ se v předchozím testování Hi-fi prototypu neukázal jako vhodný. V tuto chvíli tedy používám sousloví typ dat, které snad bude mít lepší zpětnou vazbu. Uživatelsky vytvořené vlastnosti lze také odstraňovat. Každá uživatelsky vytvořená vlastnost má také možnost nastavení povinného vyplnění při vykazování.



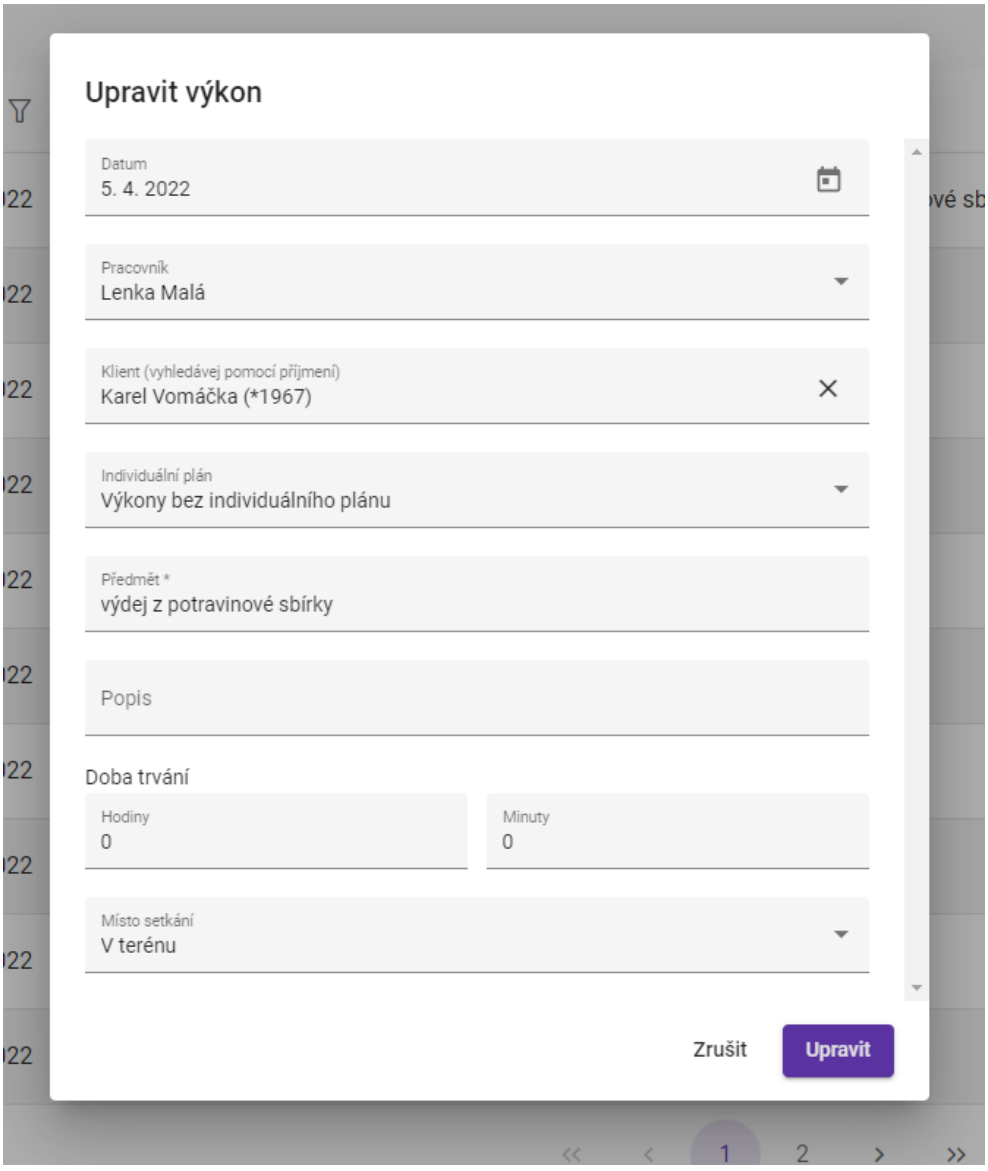
Obrázek 4.8: Přidání nové vlastnosti

Aktuálně aplikace podporuje tyto typy dat: Text, Číslo, Datum, Výčet a Ano/ne. Oproti tomu, co poskytuje Redmine jsem zde přišel se značným zjednodušením. Například čísla nerozděluji na celá a desetinná, protože si myslím, že to takto bude snazší na používání. Text a Datum jsou standardní typy dat. Pak zde máme Výčet, který umožňuje definovat konečný seznam textových možností, ze kterých poté uživatel u výkonu vybírá jednu možnost. Posledním typem je pak Ano/ne, což je klasická boolean hodnota, která nabývá pravdivé nebo nepravdivé hodnoty. Zde jsem nakonec po konzultaci s vedením charity nechal název, který používá Redmine, protože nám přišel dostatečně srozumitelný.

Většinu obrazovky s aktivitou zabírá tabulka s výkony. Ty se zobrazují v případě potřeby na více stránkách. Počítáme s tím, že výkonů budou během času tisíce, a proto je zde důležité nebrat najednou z Redminu seznam všech výkonů. Jako sloupce zde figurují vlastnosti aktivity. Podle nich lze výkony řadit nebo filtrovat. Obě tyto operace probíhají na straně serveru a neřadí a netřídí se tak jen malý úsek výkonů, který je na aktuální stránce. Pro filtrování používám standardní ikonku filtru, která možná nebude uživatelům srozumitelná na první pohled, ale jde o široce používaný symbol a dávat sem nějaké popisky by narušovalo jednoduchost. Zároveň počítám s tím, že za několik minut si každý uživatel tento symbol s filtrováním spojí. U řazení nepoužívám žádnou ikonu. Ta se zobrazí až při kliknutí na záhlaví sloupce. Když byly ikonky vidět stále, přišla mi tabulka méně přehledná a to zvláště při větším počtu sloupců. Řazení výkonů je pak dle požadavků zadavatelské strany možné maximálně nad jedním řádkem současně.

Speciální přístup je v tabulce věnován vlastnosti popis. Ta může být potenciálně docela dlouhá a tak není rozumné ji dávat do sloupečku. Místo toho jsem zvolil řešení, kdy lze každý výkon (řádek) rozbalit šipkou na levé straně řádku. Popis si takto lze zobrazit u více výkonů najednou a stav otevření/zavření se u konkrétního výkonu nezmění ani po filtrování nebo řazení.

Nové výkony lze přidat po kliknutí na tlačítko k tomu určené. Pro vyplnění nového výkonu se otevře v dialogovém okně (obrázek 4.9) stejný formulář jako v případě editace stávajícího výkonu. V tomto druhém případě je samozřejmě formulář předvyplněn aktuálními daty. K vyplnění jsou zde všechny výchozí i uživatelsky vytvořené vlastnosti. Datum uživatel vybírá přímo z kalendáře a nemusí tak řešit textový formát. Dobu trvání pak vyplňuje v tradičním formátu hodin a minut a nemusí tak například převádět minuty na desetinné číslo. Na tuto vlastnost jsem si vytvořil vlastní jednoduchý komponent, který se skládá ze dvou textových polí. Hledal jsem nějaké existující řešení, ale žádné mi nevyhovovalo. Většina z nich totiž byla uzpůsobena spíš k zadávání času v rámci dne a bylo zde tak omezení maximální dobou trvání 23 hodin a 59 minut. U výběru klienta systém napovídá podle příjmení klienta. U klienta pak lze ještě vybrat jeden z jeho individuálních plánů, které podrobněji rozeberu později.



Obrázek 4.9: Editace výkonu

Napovídání výběru klienta je děláno s ohledem na omezení počtu požadavků skrze API. Požadavek na Redmine se tedy neposílá po každém stisku klávesy ale naopak až zlomek vteřiny po tom, co byl zadán poslední znak. Jestli se během tohoto zlomku vteřiny mezitím zadá znak jiný, časovač se resetuje. Aktuálně je tato hodnota nastavena na 500 ms a uvidíme v testování, jestli je to rozumné nastavení. Kdyby byl časový úsek moc dlouhý, uživatel by zbytečně dlouho čekal na odpověď. Kdyby ale naopak byl krátký, posílal by se požadavek moc často, což by zbytečně zahlcovalo server. Snažil jsem se tak najít nějakou střední cestu. Ideální řešení často závisí na mnoha faktorech a jeho hledání by mohlo vydat na samostatnou několika stránkovou práci. Mnoho zdrojů zmiňuje údaj okolo 500 ms a třeba IBM má stejnou hodnotu jako výchozí v jednom ze svých systémů [45]. Evis tedy aktuálně používá také 500 ms.

Výkony lze logicky také mazat. Odstranění výkonu ale může být poměrně častý úkon, a proto není vhodné ho komplikovat potvrzováním skrze dialog. Zde byl zvolen jiný přístup. Uživatel klikne u výkonu na tlačítko odstranit a výkon ihned zmizí. Zároveň se mu ale v levém dolním rohu zobrazí notifikace s možností návratu smazaného výkonu. Tato notifikace je na svém místě 5 vteřin a poté se výkon reálně smaže i na úrovni Redminu. V případě mazání více výkonů se všechny uloží pod jednu notifikaci a jedním kliknutím je tak lze všechny vrátit zpátky.

4.2.6 Zaměstnanci

Na obrazovku se zaměstnanci (obrázek B.4) se dostane uživatel skrze tlačítko v menu v horní části obrazovky kdekoliv v systému. K datům o zaměstnancích se ale dostane jen uživatel s administrátorskými právy. Celá obrazovka má tři režimy. V prvním uživatel vidí jen několik tlačítek a tabulku se zaměstnanci. Tlačítka umožňují přidat nového zaměstnance a poté lze s jejich pomocí přidat nebo odebrat vlastnost. Přidání vlastností probíhá skrze dialog obdobně jako u aktivit, zde si však uživatel nemůže vybrat z již existujících vlastností a musí ji vždy vytvořit, Typy dat jsou ale stejné. Odebírání je zde řešeno formou dialogu, kde si uživatel jednoduše vybere, který uživatelsky vytvořený atribut chce odebrat.

Tabulka se zaměstnanci slouží hlavně k nalezení zaměstnance a detaily v jeho kartě si uživatel zobrazí později. Jsou zde tak jen sloupečky Jméno, Příjmení a Uživatelské jméno. Přes tyto tři sloupečky jde výsledky v tabulce řadit a v případě většího počtu zaměstnanců je k dispozici i stránkování. Navíc je v tabulce ještě čtvrtý sloupeček s označením Upozornění. Tento sloupeček může u zaměstnance ukazovat výstražný trojúhelník, který se ukáže v případě, když se některá z vlastností typu datum blíží. Pro lepší představu má zaměstnanec například vlastnost Lékařská prohlídka. Pokud je hodnota této vlastnosti dříve než za 14 dnů, je to signál pro upozornění. Tato funkce

je zde pro vedení charity, které může jednoduchým pohledem na seznam zaměstnanců zkontrolovat, zda není potřeba něco podobného zařídit.

Po kliknutí na tlačítko k přidání nového zaměstnance se dostaneme do druhého módu. Tabulka zmizí a na celé stránce je jen formulář s údaji o novém zaměstnanci. Po vyplnění a potvrzení je karta zaměstnance vytvořena a uživatel se dostane zpět k tabulce všech zaměstnanců.

Třetí mód je dostupný po kliknutí na jednoho ze zaměstnanců v tabulce. Vedle tabulky se zobrazí formulář pro editaci vybrané zaměstnanecké karty. Tento formulář svým rozložením odpovídá formuláři, skrze který se vytváří nová zaměstnanecká karta. Samozřejmě v tomto případě zde máme vyplněné informace o zaměstnanci. Je zde ale ještě třetí segment, ve kterém jsou služby zaměstnance. Zaměstnanec může figurovat ve více službách s různými rolemi. Právě zde lze tyto vazby na služby vytvářet nebo mazat.

4.2.7 Můj účet

Při znalosti obrazovky se zaměstnanci je tato část systému velmi jednoduchá. Je zde stejný formulář jako u vytváření nového zaměstnance. V tomto případě jsou v něm data z přihlášeného uživatelského účtu. K této části systému mají přístup všichni uživatelé. Dle požadavku vedení charity je totiž přínosné, aby si zaměstnanci mohli upravovat vlastní účty sami. Mohou si aktualizovat e-mailovou adresu, platnost lékařské prohlídky a podobně. Samozřejmě toto rozhodnutí klade větší zodpovědnost na zaměstnance, kteří zodpovídají za správnost údajů.

4.2.8 Klienti

Tato část implementace byla koncepčně asi nejsložitější. První návrh byl přistupovat ke klientům jako k objektům uživatel v rámci Redmine terminologie. Bohužel zde jsem se setkal s problémem, že by musel mít každý klient povinný a unikátní e-mail, což je nerealistické vzhledem k tomu, že u většiny klientů e-mailovou adresu neznáme.

Použité řešení nakonec pracuje s klientem jako s Redmine úkolem. Lze zde tak velmi dobře využít vlastnosti, kdy úkoly mohou mít jako potomky další úkoly. Dokonce pak například strávený čas rodičovského úkolu je součet všech jeho potomků. Charita potřebuje strukturu, kdy jeden klient má více individuálních plánů a ty zase mají více výkonů. Zde tedy možnost dceřinných úkolů funguje dobře. Vedení charity dále požadovalo možnost přidávat výkony rovnou ke klientům bez vazby na individuální plán. To jsem nakonec vyřešil tak, že každý klient má ve výchozím stavu připravený individuální plán s označením „Výkony bez individuálního plánu“. Do toho plánu tedy zaměstnanci mohou vykazovat všechny výkony, které nemají specifikovaný individuální plán, ale zároveň se tak nerozbije pěkná stromová struktura.

Naopak problém je v tom, že příjmení klienta je namapováno na předmět úkolu a jméno zase na popis úkolu. Hlavně se jménem jsou pak problémy, protože přes popis úkolu nelze úkoly například řadit. Celkově se tak nejedná o dokonalé řešení a v další části práce bych chtěl řešit, co se s tím dá dělat.

Samotná obrazovka B.5 je velmi podobná předchozí obrazovce se zaměstnanci. Opět zde tak jsou tři módy - pouze seznam klientů, úprava existujícího klienta a vytvoření nového klienta. Opět lze klientům přidávat uživatelsky vytvořené vlastnosti. Místo vazeb na služby u zaměstnanců zde ale mají klienti seznam individuálních plánů.

4.2.9 Individuální plán

Zde se uživatel dozví vše o výkonech, které jsou nějak tematicky sdružené pod jeden individuální plán. Obrazovka (obrázek B.6) je podobná té s aktivitou. V levé části jsou informace o samotném plánu a pod ním ještě jméno klienta, ke kterému se výkon váže, a celkový čas, který zaměstnanci charity řešením tohoto plánu strávili. Se samotnými výkony pak nelze pracovat tak dobře jako v případě obrazovky s aktivitou. Nelze zde totiž vytvářet nové výkony a ty stávající lze jen zobrazovat nebo mazat. Původně zde byl požadavek na právě tuto funkčnost. Později ale byly vzneseny požadavky na další funkce, které podrobněji popíšu v dalších částech práce.

4.2.10 Dokumentace

Základní informace o implementaci jsou v souboru README.md, který se nachází ve složce se samotným projektem. Dále jsou v kódu komentovány části kódu, které jsou komplexnější. Pro vizualizaci vazeb mezi jednotlivými částmi kódu jsem pak použil nástroj CompoDoc [46], který umožňuje pěknou vizualizaci vazeb mezi komponenty, HTML strukturu jednotlivých stránek a mnoho dalších věcí.

Kapitola 5

Testování

Tato část vytváření softwarového projektu je jednou z nejdůležitějších. Právě zde můžeme zhodnotit, jak se produkt vydařil a jestli je potřeba opravit některé chyby. Už čistě z ekonomického hlediska se ukazuje testování jako zásadní, protože jen na americkém trhu by lepší testování mohlo ušetřit až desítky miliard dolarů [47] (údaj z roku 2007 - dnešní ztráty tak mohou být ještě vyšší).

Existuje široké spektrum různých typů testů [48]. Nejprve si můžeme testy rozdělit na manuální a automatizované. Manuální testy jsou prováděné uživatelem. Tyto testy jsou často velmi časově i finančně náročné. Je totiž potřeba připravit testovací prostředí, scénář a poté samotné testery. Na druhou stranu ale člověk často odhalí problémy, které by jinak odhalit nešly. Například na testování přívětivosti uživatelského rozhraní je člověk dost praktický tester. Automatizované testy jsou prováděny počítačem. Obecně lze říct, že počítač dostane sadu příkazů, které má provést a ty provede. Mezi přednosti automatizace patří hlavně rychlost a často snadné vytvoření.

Při podrobnějším členění narazíme na takzvané Unit testy. Tento typ testů se zaměřuje na ověření funkčnosti jednotlivých funkcí, metod, objektů, komponentů a dalších. Unit testy tedy testují kód více do hloubky a neberou ohled na vyšší úrovně a interakce různých částí kódu mezi sebou. Integrační testy jsou tak často vhodným doplňkem k předchozímu typu. Zde se totiž jedná právě o ověření toho, jak spolu jednotlivé části interagují.

Funkční testy mají ověřit, zda systém umí všechny požadované funkce. Nejde tedy o kontrolu funkcí a metod v kódu, ale jedná se o globální pohled. Akceptační testování probíhá při přebírání systému zadavatelem a má ověřit, zda systém splňuje funkční i nefunkční požadavky a je tedy připraven k nasazení. End-to-end testy simulují chování uživatele v systému. Jde o průchod od začátku do konce s využitím všech různých cest, kam se může uživatel v systému dostat. Tyto testy jsou velmi silné, zároveň jde ale o dost drahé testování. Navíc jsou poměrně náchylné na změny v kódu, když jsou automatizovány.

```
import { Project } from "./project";
import { config } from '../config';
import { IssueCategory } from "../issueCategory/issueCategory
";

describe('Project', () => {
  it('should have a defined category', () => {
    let project = new Project(10, "Sluzba", "Sluzba pro lidi
    ↪  v˜nouzi", [], [new IssueCategory(11,
    ↪  config.issueCategoryClient)]);
    let projectCategory = project.getIssueCategoryForName(con
    ↪  fig.issueCategoryClient);
    expect(projectCategory).toBeDefined();
    expect(projectCategory?.id === 11).toBeTruthy();
    expect(projectCategory?.name ===
    ↪  config.issueCategoryClient).toBeTruthy();
  });

  it('should have no category', () => {
    let project = new Project(10, "Sluzba", "Sluzba pro lidi
    ↪  v˜nouzi", [], []);
    let projectCategory = project.getIssueCategoryForName(con
    ↪  fig.issueCategoryClient);
    expect(projectCategory).toBeUndefined();
  });
})
```

Obrázek 5.1: Příklad Unit testu

5.1 Unit testy

Jako první jsem si vybral Unit testy. Tento typ testů jsem si zvolil hlavně pro jejich snadnou správu a rychlé výsledky. Moje práce se ale ve velké míře točila okolo návrhu uživatelského rozhraní a tak nešlo o primární způsob testování.

5.2 Uživatelské testování

Pro projekt s velkým důrazem na srozumitelné a jednoduché uživatelské rozhraní je snad nejdůležitější kladné přijetí uživateli. Ideální tedy je, když systém mohou otestovat přímo zaměstnanci - budoucí uživatelé. Pro testování jsem si v systému připravil testovací data, která by měla uživateli připomínat pracovní prostředí.

Dále jsem vypracoval scénář [49], jehož body měli testeři splnit. Scénář (příloha E) má opět testera co nejvíce zasadit do role pracovníka charity. Snažil jsem se začlenit hlavně ty funkce, které budou pracovníci využívat nejčastěji. Dále jsem se také zaměřil na detaily, u kterých jsem si nebyl tak úplně jistý, že je uživatel bude schopný používat intuitivně. Scénář jsem se snažil psát tak, aby uživatele zaujal. Aby scénář nebyl seznamem mnoha kroků, rozdělil jsem je na dvě části. První by vykonávalo vedení charity a druhou zaměstnanci. Obě části na sebe navazují, což působí při vyplňování více reálně.

Většinu času jsem nechával testery čistě se zadáním, protože jsem jim nechtěl nijak napovídat a ovlivnit tak testování. Když ale svůj úkol dokončili, ptal jsem se, proč se rozhodli zrovna pro tento postup nebo proč se zrovna zde zasekli. Snažil jsem se je také uvést do globálního pohledu na systém, když zkoušeli jednotlivé funkce. Většinou ale chápali užitečnost jednotlivých úkonů sami.

Na konci testování jsem testery poprosil o vyplnění krátkého dotazníků a i ústně jsme pak ještě rozebírali jejich pohled na systém. Dotazník jsem použil z toho důvodu, že jsem se nechtěl odvolávat pouze na mou interpretaci jejich pocitů a postřehů. Dotazník ale nebyl nijak složitý a každý se mohl libovolně rozepsat v celkem třech otázkách. V první otázce jsem se ptal, co se uživateli na systému líbilo. Druhá otázka měla naopak zjistit, co se uživatelům nelíbilo. Poslední dotaz pak směřoval k získání návrhů pro možná budoucí vylepšení.

Čistě z technického hlediska probíhalo testování na serverech služby Microsoft Azure [50]. Tento hosting byl už v předchozí práci Bc. Davida Holkupa vybrán jako nejlepší možnost pro dlouhodobé nasazení. Chtěl jsem tedy ověřit, jak si zde systém povede ve skutečnosti. Hostovány zde byly celkem tři služby a to: databáze MySQL, systém Redmine a nový systém Evis.

Testování jsem provedl celkem na třech lidech, z nichž dva pracují v charitě, pro kterou se systém vytváří. Nejde o extra velký vzorek, ale i tak jsem se z něj pokusil dostat maximum informací. Jelikož jde o menší vzorek testerů, rozhodl jsem se nejdříve rozebrat výstupy testerů postupně a poté se zaměřit na shrnutí. Moje autentické zápisky z testování jsou v Příloze F.

5.2.1 První tester

Prvním testerem byl můj kamarád Bc. Petr Větrovský. Petr sice nemá zkušenost s prací v charitě, ale používá také plno informačních systémů a jeho pohled také může přinést zajímavé postřehy. Navíc jsem chtěl mít někoho, na kom si nacvičím vymyšlený scénář testování. U Petra bylo celkově vidět, že tolik nerozumí tomu, jak fungují procesy charity. Dobře však zvládal ovládání jednotlivých prvků uživatelského rozhraní, které se vyskytují i v jiných systémech.

V některých částech se snažil do systému psát různé nesprávné vstupy. Na některých místech to nešlo, u zápisu času se mu ale podařilo do políčka pro počet hodin napsat například „2 + 10“. Systém tento vstup následně uložil jako 0, což není nejlepší chování systému. Ideální by zde bylo uživateli podobné

výrazy nedovolit. Na druhou stranu je dobré, že to systém nijak nerozhodí a uživatel po úpravě vždy vidí uloženou hodnotu, takže si ji může zkontrolovat. Dále si Petr stěžoval na umístění notifikace pro možnost vrácení smazaného výkonu. Tato volba se totiž objeví vlevo dole a ostatní informační notifikace se objevují naopak v horním pravém rohu. Mým záměrem bylo rozdělit funkční a informační notifikace, ale minimálně Petrovi to nevyhovovalo.

5.2.2 Druhý tester

Dále jsem dostal možnost testovat systém přímo na zadavatelce a ředitelce charity Mgr. Karolíně Píchové, DiS. Podle mého pochopení fungování této charity jde o člověka, který nejlépe ví, jak vypadají procesy charity a to hlavně z globálního hlediska. Z hlediska testování měla zadavatelka jasnou výhodu v tom, že jsme se o systému pravidelně bavili a postupně ho společně upravovali. Až při testování měla ale poprvé možnost si vyzkoušet sama, jak se systém ovládá.

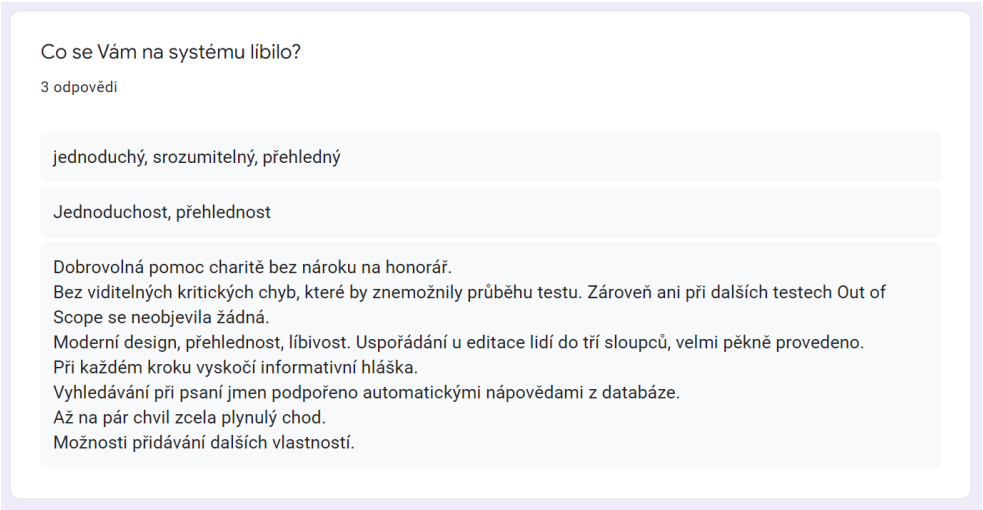
Při testování samotném bylo vidět, že zadavatelce nedělá problém se v systému vyznat a většinou postupovala docela rychle. Nedošlo zde k žádným větším zádrhelům, ale potvrdilo se, že některé funkce chybí a jsou opravdu potřeba. V tomto případě ale šlo o funkce, které v původním zadání nebyly a podrobněji je rozeberu později.

5.2.3 Třetí tester

U třetího testera jsem měl největší obavy. S Mgr. Petrem Švepošem jsme prováděli testování už u původního Hi-fi prototypu, který vznikl ještě v rámci předmětu NI-NUR (část 2.1.2). Pan Švepoš je zaměstnancem charity a jeho pohled je tak velmi důležitý. Oproti zadavatelce se totiž na celý systém dívá víc lokálně a právě tento pohled také potřebuji. Důvodem mých obav bylo to, že při minulém testování se ukázalo, že panu Švepošovi prototyp moc nevyhovoval a se systémem si nerozuměl. Vyplňoval špatná políčka, klikal na špatná tlačítka a tak celkově se zdál v systému ztracený.

V rámci testování další verze systému byla ale situace mnohem lepší. Tentokrát bylo vidět, že vždy po chvíli většinou našel správné řešení. U několika případů se pak zasekl nebo si nevěděl rady. Občas jsem mu tak napověděl a poté už nebyl problém úkol dokončit. V některých případech pak udělal úkol jiným způsobem, než jsem původně zamýšlel, ale po vysvětlení mu bylo jasné, jak by daný úkol udělal rychlejším postupem. I zde bylo vidět, že postrádal některé funkce, které by udělaly práci přímočařejší a rychlejší. Šlo o stejné funkce jako v předchozím případě.

Celkově jsem byl ze zlepšení hodně pozitivně překvapený. Lepší interakci se systémem u pana Švepoše si vykládám třemi způsoby. Zaprvé pan Švepoš měl už se starší verzí systému zkušenost. Šlo sice o dost rozdílnou verzi bez velké části funkcí a také už je to několik měsíců od prvního testování, ale i tak



Obrázek 5.2: Pozitivní zpětná vazba

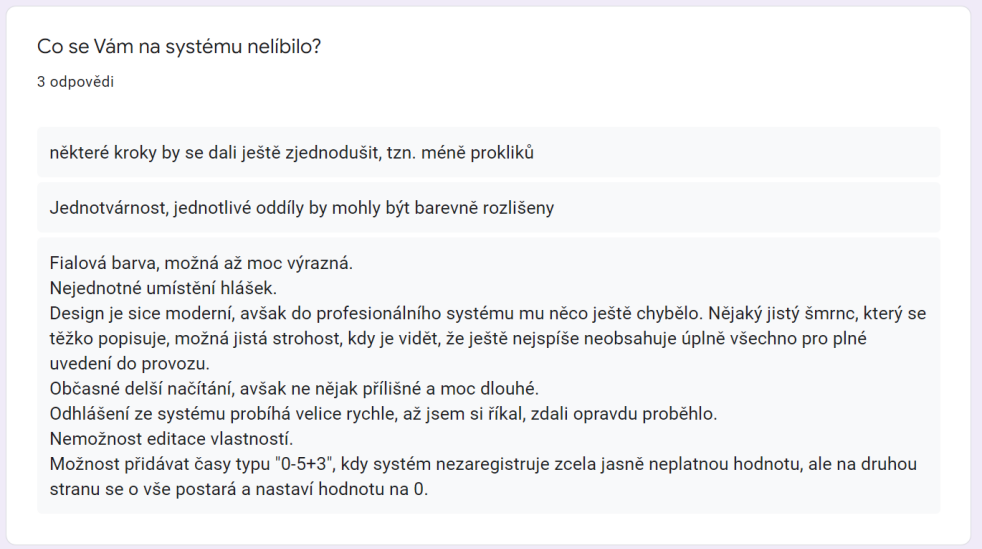
to mohlo nyní pomoci. Zadruhé mohl být scénář napsán víc srozumitelněji. Zatřetí je zde také důvod, že systém dostal vylepšení, která zlepšují uživatelský zážitek.

Celkově jde mnohdy o drobnosti, ale například zobrazení toho, že se něco načítá nebo ukázání notifikace o výsledku požadavku dává uživateli nějakou jistotu, kterou ani sám často nedokáže popsat. U tohoto konkrétního testera pak hraje roli ještě změna některých termínů. Právě pan Švepoš nás totiž minule upozorňoval, že termíny jako atribut nebo datový typ jsou pro něj a jeho kolegy často neznámé. V aktuální verzi tak používám místo atributu vlastnost a místo datového typu typ dat. I další termíny a hlášky jsem se snažil formulovat s myšlenkou právě na to, aby bylo vše srozumitelné a jsem tak rád, že tentokrát zde nebyl problém.

5.2.4 Shrnutí

Testování jako celek bych chtěl shrnout v této části. Postupně se zaměřím na jednotlivé otázky z dotazníku, kde je dobře vidět, na čem se například testeři shodli.

Všichni uživatelé kvitovali přehlednost a jednoduchost systému (obrázek 5.2), což docela dobře ukazuje na splnění jednoho z nefunkčních požadavků v části 2.4.1. Dále zde bylo zmíněno, že je systém moderní, což zase trefně naráží na to, že systém by měl z části nahradit starší aplikaci. Srozumitelnost je také velmi požadovanou vlastností systému s uživatelským rozhraním a i tento termín se v dotazníku vyskytuje. Nakonec uživatelé kladně hodnotili přímo některé funkce jako třeba možnost přidávání dalších vlastností nebo

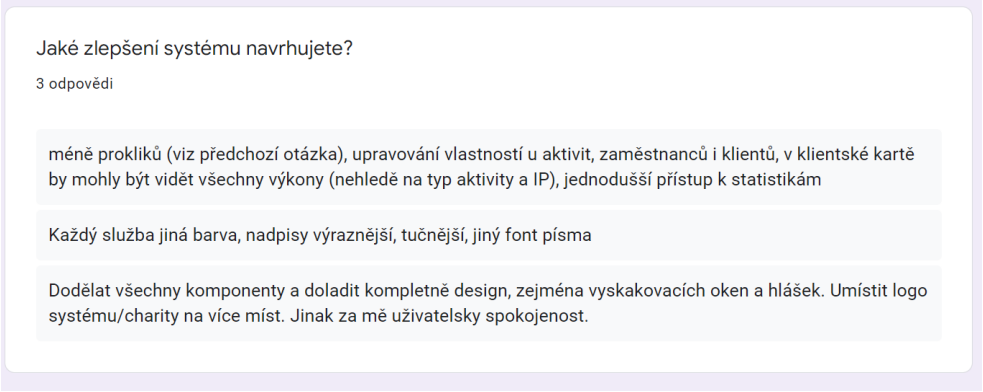


Obrázek 5.3: Negativní zpětná vazba

vytváření vlastních uživatelských vlastností k různým objektům v systému.

Ne vše ale bylo dle testerů dokonalé a tak jsem se v dotazníku dočkal i nějakých výtek (obrázek 5.3). Zajímavé například byly připomínky k barvám, kdy byla kritizována jednotvárnost nebo použití moc výrazné fialové barvy. Jako možné řešení je pak navrhováno dát různým částem systému různou barvu. Pak jsou zde problémy s tím, že uživatelé nenašli požadovanou funkci. Jako dvě hlavní chybějící funkce mohu jmenovat nemožnost vytváření a editace výkonů přímo v individuálních plánech a klientských kartách. Dále se ještě hledala možnost úpravy vlastností jednotlivých objektů.

Problém byl také s občas delším načítáním, které ale přisuzuji hostingu, který probíhal v rámci zkušebního období na službě Microsoft Azure. Hosting totiž systém při delší nečinnosti uspal a na začátku testování tak docházelo k drobným problému. Tato věc je ale řešitelná správným nastavením. Dále přineslo nasazení na hosting ještě problém s tím, že nebylo možné aktualizovat stránku, což je zřejmě způsobeno trochu odlišným způsobem spouštění aplikace. Posledním problémem, který souvisí s hostingem je nemožnost odesílat z Redminu e-maily. Došlo zde totiž k tomu, že Google (jehož e-mailovou schránku jsem použil) nebral přihlášení z Azure jako dostatečně bezpečné a odmítl přihlášení.



Obrázek 5.4: Návrhy na zlepšení

Uživatelské návrhy na zlepšení (obrázek 5.4) se často točí okolo zlepšení předchozích nedostatků a jsou zde tak tipy na různé barvy, výraznější písmo, jiný font nebo přidání výše uvedených funkcí, které by zjednodušily vykazování a celkovou práci se systémem. Zadavatelka pak separátně ještě zkoušela možnost vytváření statistik v prostředí Redmine, ale to jí přišlo moc složité a tak nakonec navrhovala, jestli by nešlo generovat statistiky přímo v prostředí Evis.

5.3 Hodnocení zadavatelky

V této části se nebudu věnovat nějakému standardizovanému testu. Mám ale pocit, že slovo zadavatele by zde mělo být. V jistém slova smyslu chápu vyjádření zadavatele jako velmi důležitou formu testu, která doplňuje pohled na to, zda projekt uspěl. Z mého pohledu není úplně šťastné někoho parafrázovat, protože se vždy forma i obsah sdělení trochu posune od původní verze. Chtěl bych zde tedy uvést nezkreslené hodnocení projektu ze strany zadavatelky a ředitelky Farní charity Jindřichův Hradec Mgr. Karolíny Píchové Dis.:

Jsem moc ráda za iniciativu studentů Davida Holkupa a Jana Horyny, oboru Softwarové inženýrství. Potěšil mne zájem těchto studentů zpracovat v rámci diplomové práce smysluplný a prospěšný projekt. Z mého úhlu pohledu byl tento prvotní nepsaný cíl naplněn. Studenti se s nadšením pustili do nelehkého a pro nás potřebného úkolu, zpracování systému, v němž bychom byli schopni evidovat naši činnost, tak abychom poskytovali kvalitní služby a naplnili požadavky našich donátorů. Sociální služby využívají různé systémy evidence, a to v závislosti na typu poskytované služby. Pro naše potřeby využíváme systému e-quip. Tento systém je sice nastaven, tak aby jej mohlo využívat široké spektrum sociálních služeb, nicméně je málo přehledný a zbytečně složitý.

Zároveň umožňuje evidovat pouze registrované sociální služby. Z těchto důvodů jsme se tedy s výše uvedenými studenty domluvili na spolupráci. Zadáním bylo vytvoření uživatelsky jednoduchého, přehledného prostředí, ve kterém by bylo možné evidovat práci s klienty (karta klienta, individuální plány, výkony, ...) , dále evidovat údaje o zaměstnancích a také generovat potřebné statistiky. Samozřejmě je zde ještě několik dílčích věcí, které by se daly vylepšit, nicméně domnívám se, že se tento úkol studentům podařilo splnit.

Kapitola 6

Budoucnost projektu

6.1 Nové funkce

Jak jsem psal už u testování, ze strany zaměstnanců i vedení charity se v průběhy vypracování ukázaly nové požadavky, které velmi usnadní každodenní práci při vykazování schůzek s klienty. K těmto novým požadovaným funkcím jsme došli spolu se zadavatelkou až ke konci implementace aktuální verze. Domluvili jsme se tedy, že nové funkce budou řádně analyzovány a implementovány až v rámci příští verze. Z tohoto důvodu jsem tak práci nesměřoval k nasazení v reálném pracovním prostředí.

Když si rozebereme nové funkce, velkým požadavkem zde byla možnost vytvářet a editovat výkony v rámci karty klienta nebo jeho individuálních plánů. Od současného řešení se tak bude muset několik věcí změnit. Vytváření a editace výkonů je prováděna pomocí dialogových oken. V aktuální verzi jsem se snažil dělat tato okna co nejjednodušší, aby je šlo použít v různých částech systému. Nyní to ale vypadá spíš na to, že se každá z funkcí bude využívat na více místech a dávalo by pak smysl dialogy předělat. Aktuálně si představuji, že by dialogy dostaly více vlastní logiky a přímo v nich by se provádělo volání požadavků na API.

Tato změna by přinesla ještě jednu výhodu, kterou jsem si původně neuvědomil. Když uživatel vytvoří nebo upraví výkon a klikne na Vytvořit/Uložit, dialog se zavře a při negativní odpovědi uživatel přijde o rozepsaný výkon. Kdyby se požadavek prováděl v rámci dialogu, mohl by uživatel v případě chyby opravit některou z vlastností a udělat požadavek nový. Celá funkcionalita by si ale samozřejmě zasloužila podrobnější analýzu.

Druhou větší novou funkcí by měla být editace vlastností, kterou testeři hledali marně. Původní myšlenka byla taková, že vlastnosti bude typicky nutné editovat hlavně hned po vytvoření, kdy se člověk například překlepne v názvu vlastnosti. V tomto případě pak jde vlastnost smazat a vytvořit novou. K úpravám využívaných vlastností tak mělo docházet velmi zřídka a v těchto momentech jsem spoléhal na rozhraní Redminu.

Je otázkou, jakým způsobem by bylo vhodné tuto funkci implementovat. Například v rámci aktivity se nabízí vedle ikonky koše přidat i ikonku tužky pro editaci a editovat skrze podobné dialogové okno, ve kterém se vlastnosti vytváří. U klientů a zaměstnanců už ale není umístění tlačítek pro editaci tak jasné. Jako druhá možnost mě napadá udělat samostatnou stránku, která by se věnovala speciálně správě vlastností. Vlastnosti mají totiž v systému různé typy, vazby na různé objekty a tak by možná dávalo smysl vidět a mít možnost upravit tyto věci na jednom místě. Těžko ale říct, jak by uživatelé podobnou stránku přijali. Znovu je zde tedy potřeba podrobnější analýza.

Nezpracovanou funkcí také zůstává virtuální nástěnka s poznámkami, která se do aktuální verze systému nedostala z důvodu nízké priority. V budoucnu by tak možná i na tuto funkci mohla přijít řada.

6.2 Práce se statistikami

Samostatnou kapitolou jsou statistiky, u kterých jsem věřil, že se využije rozhraní Redminu. To by zvládlo požadované funkce velmi jednoduše. Je zde ale problém s nepřehledností oproti rozhraní Evis (což bylo i jedním z výstupů uživatelského testování). Nabízí se tak možnost přidat do Evisu i možnost generování statistik, i když by to mohlo být časově dost náročné. Opět bude třeba analyzovat i různé další možnosti, které třeba nejsou na první pohled zřejmé.

6.3 Problémy s Redminem

Redmine je v mnoha věcech přínosná část celého řešení, ať už jde o řešení přihlašování uživatelů, správu jejich účtů, široké možnosti práce s úkoly a další. V interakci se systémem jsem ale narazil i na některé problémy.

Hlavní zádrhel vidím v práci s klienty, kde Redmine nativně něco takového nepodporuje a je tak potřeba „ohýbání“ existujících funkcí. To s sebou nese například nemožnost klienty řadit podle křestního jména, protože reálně je tato informace namapována na popis u úkolu a přes popis v Redminu řadit úkoly nejde. Problematické je také rozdělení vlastností na ty pro výkony a na ty pro klienty. Redmine totiž vidí obojí jako atributy úkolů. Tyto problémy tak přinášejí omezení funkcí v prostředí Evis nebo některé části implementace nemohou být tak přehledné a jednoduché.

Dále jde o drobnosti. Například jsem se setkal s tím, že fronta může mít maximálně 30znakový název (stejný problém řešil už před 12 lety jiný uživatel u jiného objektu [51]). Problém je pak i s filtrováním úkolů podle jejich kategorií skrze API. S touto funkcí jsem počítal, ale nakonec jsem nejen já zjistil, že nefunguje [52]. V těchto případech nejde o zásadní věci, ale jsou to problémy, které znesnadňují už aktuální funkce prostředí Evis. Před jeho rozšiřováním

se tak musí dát velký důraz na to, aby podobné nebo ještě větší problémy nenastaly při implementaci nových funkcí.

Nakonec vyvstává otázka, jestli je Redmine vůbec potřeba. Možná by bylo lepší vzhledem k problémům a potenciálnímu přesunu statistik do systému Evis přijít s vlastní backendovou částí. Na druhou stranu ale pořád Redmine přináší výše jmenované funkce, které jsem nemusel implementovat sám. Kolem něj je také komunita, která přichází s různými pluginy a návrhy na vylepšení. Rozhodnutí na případné zahození a nahrazení Redminu bych osobně dělal až po analýze nově požadovaných funkcí. Aktuálně totiž pro jasný pohled nemám dost informací.

6.4 Souhrn

Pro dokončení systému do bodu, kdy bude připraven k nasazení, je nutné podrobně analyzovat nové funkce a jejich implementaci do stávajícího řešení. Při tomto procesu je potřeba dávat speciální pozor na to, jak bude fungovat integrace se systémem Redmine, u kterého je potřeba vyhodnotit, jestli stále výhody jeho používání převažují nad nevýhodami a problémy.

Závěr

Cílem této diplomové práce bylo usnadnit zaměstnancům Farní charity Jindřichův Hradec práci pomocí nového evidenčního systému. Tohoto cíle nebylo úplně dosaženo. Splněny ale byly dílčí cíle, díky kterým je systém ve verzi, která je poměrně blízko nasazení. Splnění hlavního cíle zabránily hlavně nové požadavky, které byly vzneseny při implementaci a nemohly být dostatečně podrobně analyzovány a zpracovány. Myslím, že bylo rozumné netlačit na nasazení nehotového produktu. Výchozí stav pro dokončení jsem definoval v 6. kapitole a doufám, že s její pomocí se celý projekt podaří dokončit a bude zdárně nasazen. Osobně bych se rád na dalším vývoji tohoto projektu podílel.

Literatura

[1] Holkup, D.: *Evidenční systém výkonů pro Farní charitu Jindřichův Hradec*. Diplomová práce, Fakulta informačních technologií ČVUT v Praze, únor 2022, https://dspace.cvut.cz/handle/10467/99558.

[2] The modern web developer's platform. *Angular*, [cit. 2022-02-27]. Dostupné z: `https://angular.io/`

[3] Redmine. *Redmine*, [cit. 2022-02-19]. Dostupné z: `https://www.redmine.org/`

[4] Ruby is... *Ruby*, [cit. 2022-02-21]. Dostupné z: `https://www.ruby-lang.org/en/`

[5] Compress the complexity of modern web apps. *Rails*, [cit. 2022-02-21]. Dostupné z: `https://rubyonrails.org/`

[6] Team, I.: *The Rise of Open-Source Software*. IntroBooks. Dostupné z: `https://books.google.cz/books?id=AgXfDwAAQBAJ`

[7] Masse, M.: *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, 2011, ISBN 9781449319908. Dostupné z: `https://books.google.cz/books?id=eABpzyTcJNIC`

[8] Redmine API. *Redmine*, [cit. 2022-02-19]. Dostupné z: `https://www.redmine.org/projects/redmine/wiki/rest_api`

[9] King, G.; Harris, T.; Kuate, P.; aj.: *NHibernate in Action*. Manning, 2009, ISBN 9781638354765. Dostupné z: `https://books.google.cz/books?id=AzozEAAAQBAJ`

[10] Forums. *Redmine*, [cit. 2022-03-19]. Dostupné z: `https://www.redmine.org/projects/redmine/boards`

[11] Plugins Directory. *Redmine*, [cit. 2022-03-19]. Dostupné z: https://www.redmine.org/plugins

[12] Redmine theme list. *Redmine*, [cit. 2022-02-19]. Dostupné z: https://www.redmine.org/projects/redmine/wiki/theme_list

[13] Professional Redmine Plugins. *RedmineUP*, [cit. 2022-03-19]. Dostupné z: https://www.redmineup.com/

[14] Munsaka, T.: *Communication is Complex. Definitions, Types and Problems*. GRIN Verlag, 2014, ISBN 9783656862956. Dostupné z: https://books.google.cz/books?id=UojgBQAAQBAJ

[15] Druhy komunikace. *rvp.cz*, [cit. 2022-03-19]. Dostupné z: https://wiki.rvp.cz/Knihovna/1.Pedagogick%C3%BD_lexikon/K/Komunikace/Druhy_komunikace

[16] 4 Types of Communication and How to Improve Them. *indeed.com*, [cit. 2022-03-19]. Dostupné z: https://www.indeed.com/career-advice/career-development/types-of-communication

[17] 3 Main Types of Communication. *Arkansas State University*, [cit. 2022-03-19]. Dostupné z: https://degree.astate.edu/articles/undergraduate-studies/3-main-types-of-communication.aspx

[18] What are Communication Strategies? - Definition, Types and Examples. *study.com*, [cit. 2022-03-19]. Dostupné z: https://study.com/academy/lesson/what-are-communication-strategies-definition-types-examples.html

[19] Chung, L.; Nixon, B.; Yu, E.; aj.: *Non-Functional Requirements in Software Engineering*. International Series in Software Engineering, Springer US, 2012, ISBN 9781461552697. Dostupné z: https://books.google.cz/books?id=MNrcBwAAQBAJ

[20] Sage, A.; Rouse, W.: *Handbook of Systems Engineering and Management*. Wiley series in systems engineering and management, Wiley, 2014, ISBN 9780470083536. Dostupné z: https://books.google.cz/books?id=eFRwQuzPnEcC

[21] FURPS – Model kvality testování. *Deník testerky*, [cit. 2022-04-30]. Dostupné z: http://deniktesterky.cz/furps-model-kvality-testovani/

[22] Bittner, K.; Spence, I.; Jacobson, I.: *Use Case Modeling*. The Addison-Wesley object technology series, Addison Wesley, 2003, ISBN 9780201709131. Dostupné z: https://books.google.cz/books?id=zvxfXvEcQjUC

[23] Fowler, M.; Kobryn, C.; Online, S. T. B.: *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Object Technology Series, Addison-Wesley, 2004, ISBN 9780321193681. Dostupné z: https://books.google.cz/books?id=nHZslSr1gJAC

[24] Login. *The University of Texas at Austin*, [cit. 2022-04-15]. Dostupné z: https://www.cs.utexas.edu/~mitra/csFall2011/cs312/lectures/login.html

[25] What Is a Use Case? *Wrike*, [cit. 2022-04-15]. Dostupné z: https://www.wrike.com/blog/what-is-a-use-case/

[26] Three-Tier Architecture. *IBM*, [cit. 2022-04-16]. Dostupné z: https://www.ibm.com/cloud/learn/three-tier-architecture

[27] Why MySQL? *MySQL*, [cit. 2022-02-27]. Dostupné z: https://www.mysql.com/why-mysql/

[28] PostgreSQL: The World's Most Advanced Open Source Relational Database. *PostgreSQL*, [cit. 2022-02-27]. Dostupné z: https://www.postgresql.org/

[29] Co si oblíbíte na systému SQL Server 2019. *Microsoft*, [cit. 2022-02-27]. Dostupné z: https://www.microsoft.com/cs-cz/sql-server/sql-server-2019

[30] What Is SQLite? *SQLite*, [cit. 2022-02-27]. Dostupné z: https://www.sqlite.org/index.html

[31] What is a Relational Database? *Amazon AWS*, [cit. 2022-02-27]. Dostupné z: https://aws.amazon.com/relational-database/

[32] Appropriate Uses For SQLite. *SQLite*, [cit. 2022-02-27]. Dostupné z: https://www.sqlite.org/whentouse.html

[33] Lesyuk, A.: *Mastering Redmine - Second Edition*. Community experience distilled, Packt Publishing, 2016, ISBN 9781785881305. Dostupné z: https://subscription.packtpub.com/book/networking_and_servers/9781785881305/1/ch01lvl1sec10/mysql-postgresql-sqlite-or-microsoft-sql-server

[34] Benyon, D.: *Designing User Experience*. Pearson Educación, 2019, ISBN 9781292155531. Dostupné z: https://books.google.cz/books?id=MXqFDwAAQBAJ

[35] Lo-fi prototypování: kdy se hodí a jak na něj. *Medium*, [cit. 2022-04-15]. Dostupné z: https://medium.com/design-kisk/lo-fi-prototypov%C3%A1n%C3%AD-kdy-se-hod%C3%AD-a-jak-na-n%C4%9Bj-5a965914ae39

[36] Life's too short for bad software! *Balsamiq*, [cit. 2022-04-15]. Dostupné z: https://balsamiq.com/

[37] Prototype while you design, and vice versa. *Figma*, [cit. 2022-04-15]. Dostupné z: https://www.figma.com/prototyping/

[38] Lo-fi prototypování: kdy se hodí a jak na něj. *blog.prototypr.io*, [cit. 2022-04-15]. Dostupné z: https://blog.prototypr.io/high-fidelity-prototyping-what-when-why-and-how-f5bbde6a7fd4

[39] Download. *Redmine*, [cit. 2022-04-16]. Dostupné z: https://www.redmine.org/projects/redmine/wiki/download

[40] WWW-Authenticate. *Mozilla*, [cit. 2022-04-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/WWW-Authenticate

[41] Angular Material. *Angular*, [cit. 2022-04-16]. Dostupné z: https://material.angular.io/

[42] 11 Popular Angular UI Libraries To Try In 2021. *Medium*, [cit. 2022-04-16]. Dostupné z: https://javascript.plainenglish.io/10-popular-angular-ui-libraries-to-try-in-2021-5a24c8aaa902

[43] 10 Best Angular DataTables with Pagination, Sorting and Filter. *NgDevelop*, [cit. 2022-04-16]. Dostupné z: https://www.ngdevelop.tech/best-angular-tables/amp/

[44] Table. *Prime Faces*, [cit. 2022-04-16]. Dostupné z: https://www.primefaces.org/primeng/table

[45] Setting the debounce timeout. *IBM*, [cit. 2022-04-16]. Dostupné z: https://www.ibm.com/docs/en/spm/7.0.9?topic=navigator-setting-debounce-timeout

[46] The missing documentation tool for your Angular, Nest, Stencil application. *Compodoc*, [cit. 2022-04-26]. Dostupné z: https://compodoc.app/

[47] McLeod, R.; Everett, G.: *Software Testing: Testing Across the Entire Software Development Life Cycle*. IEEE Press, Wiley, 2007, ISBN 9780470146347. Dostupné z: https://books.google.cz/books?id=z8UdPmvkBHEC

[48] The different types of software testing. *Atlassian*, [cit. 2022-04-26]. Dostupné z: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

[49] How to write an effective usability testing script (+ example). *Maze*, [cit. 2022-04-28]. Dostupné z: `https://maze.co/guides/usability-testing/script/`

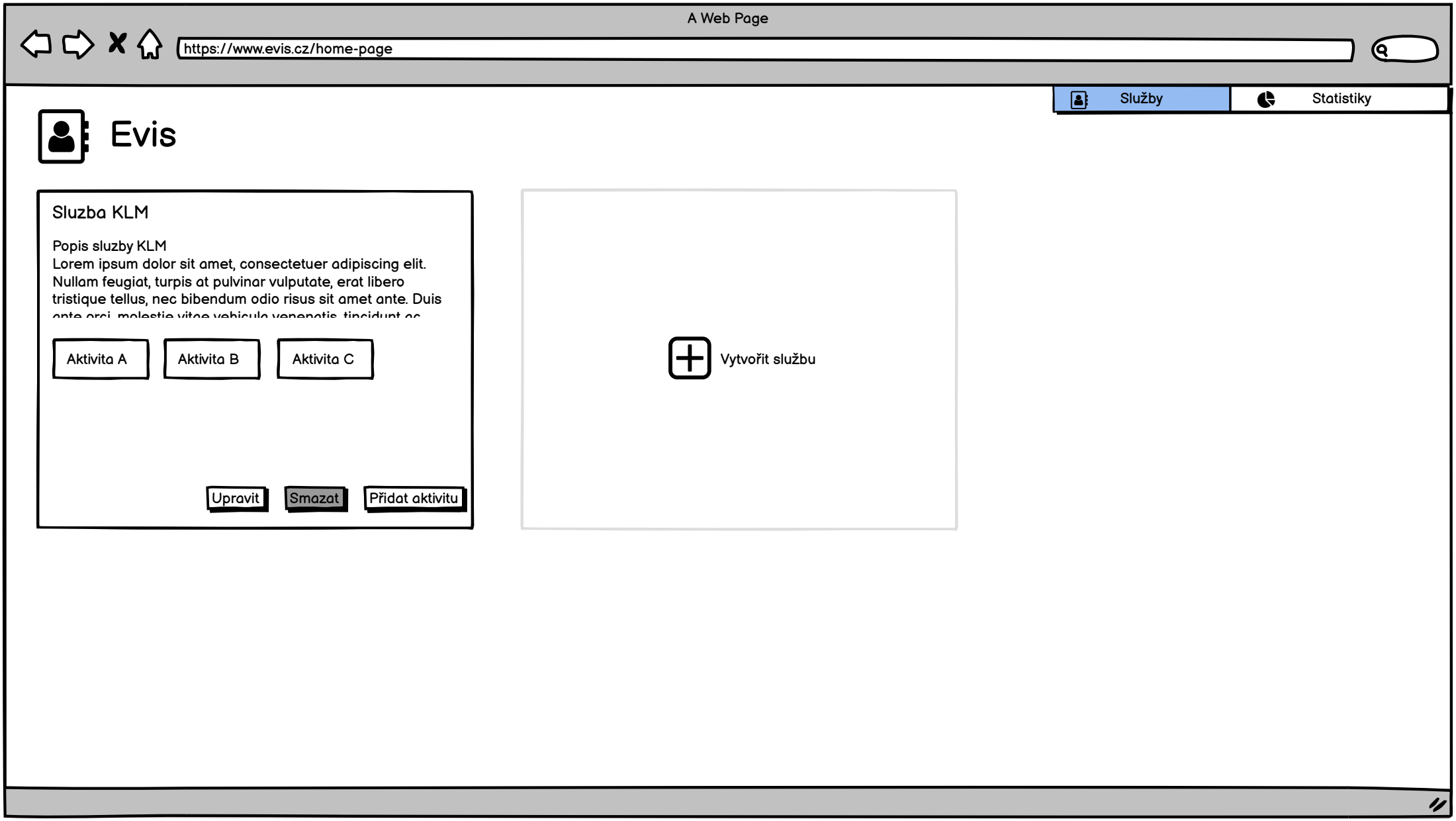
[50] Don't just migrate. Modernize. *Microsoft Azure*, [cit. 2022-04-28]. Dostupné z: `https://azure.microsoft.com/en-us/`

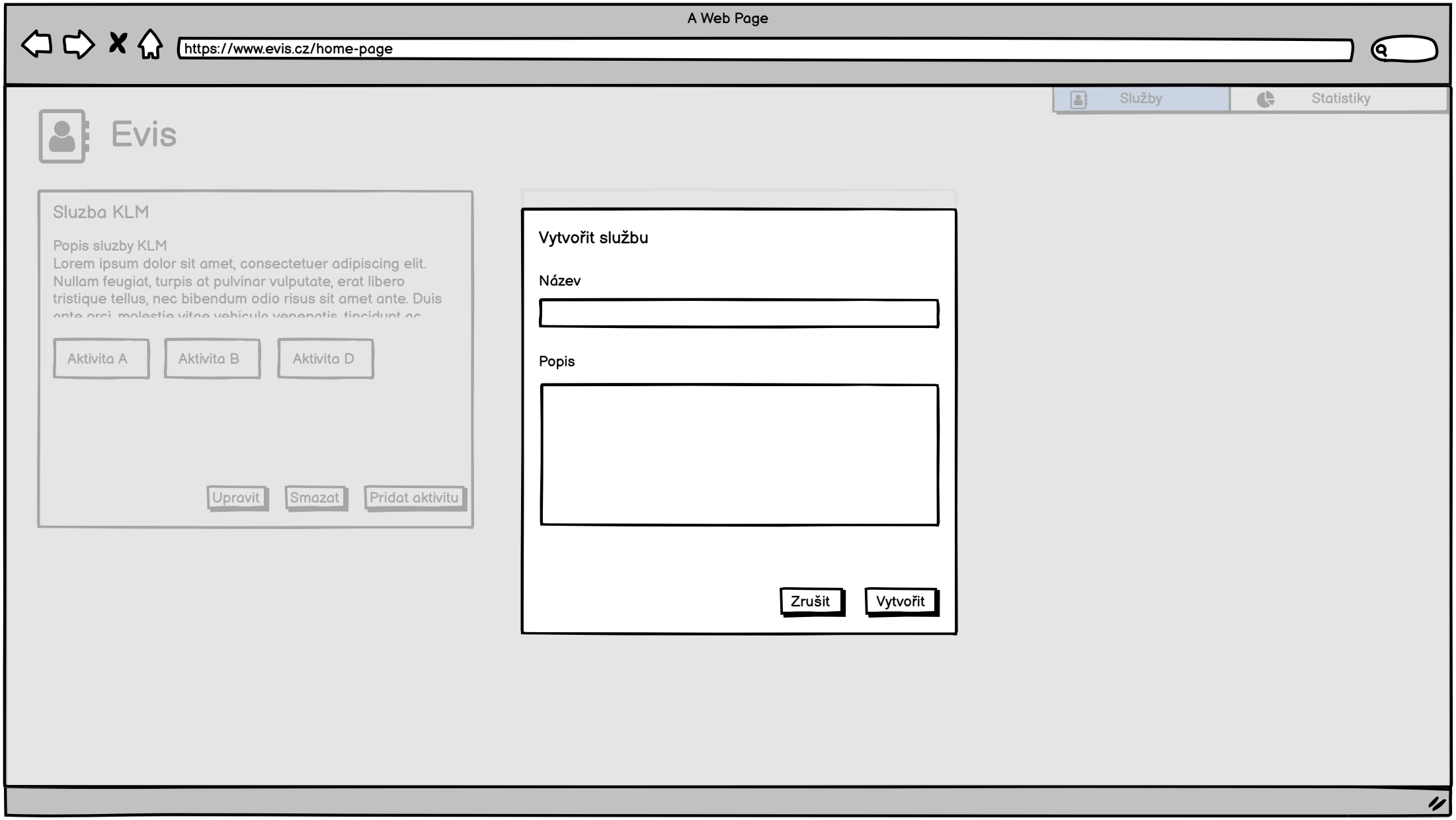
[51] Number of characters. *Redmine*, [cit. 2022-04-29]. Dostupné z: `https://www.redmine.org/boards/2/topics/13170`

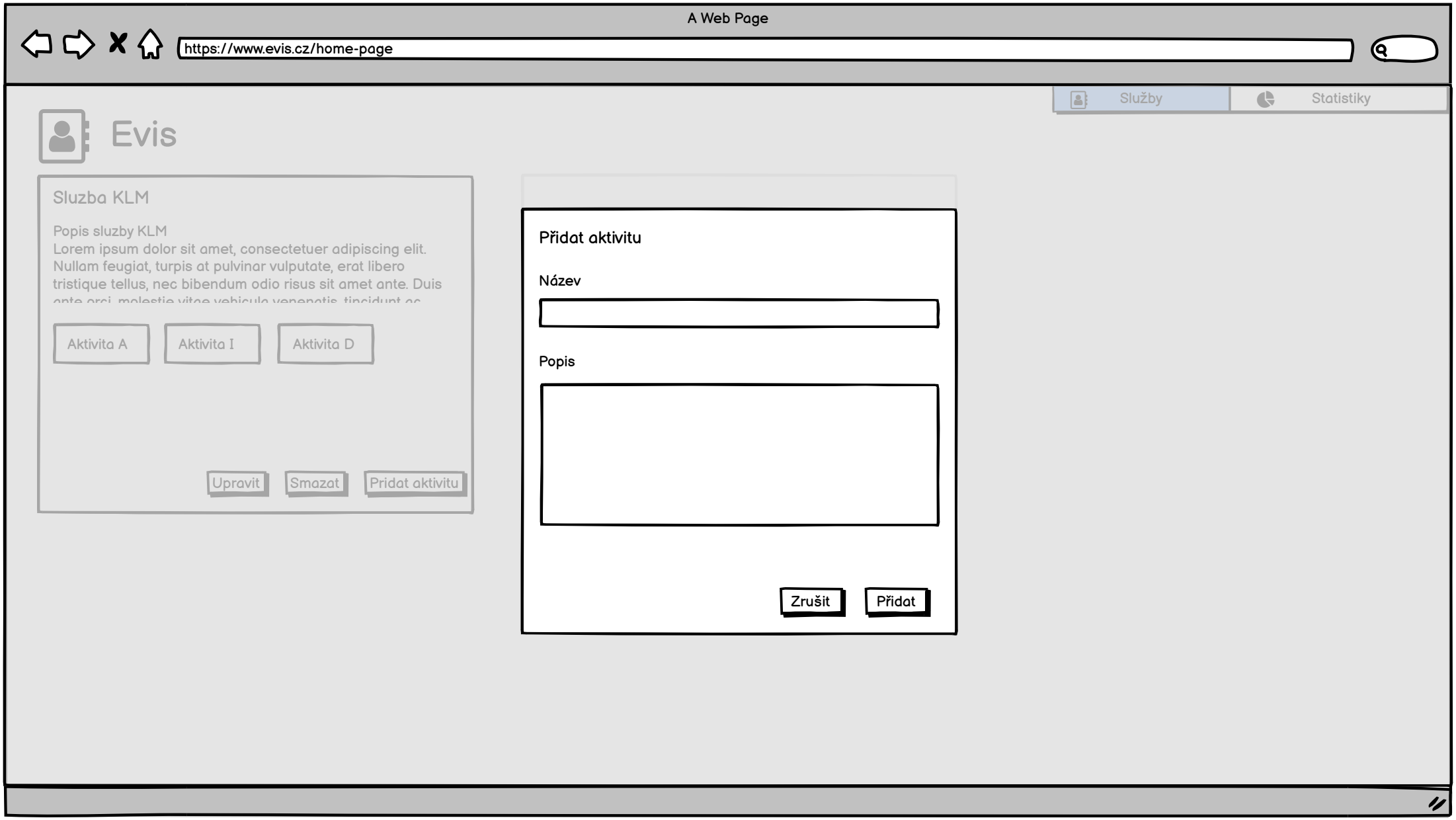
[52] Rest Api: filter by issue category. *Redmine*, [cit. 2022-04-29]. Dostupné z: `https://www.redmine.org/issues/35843`

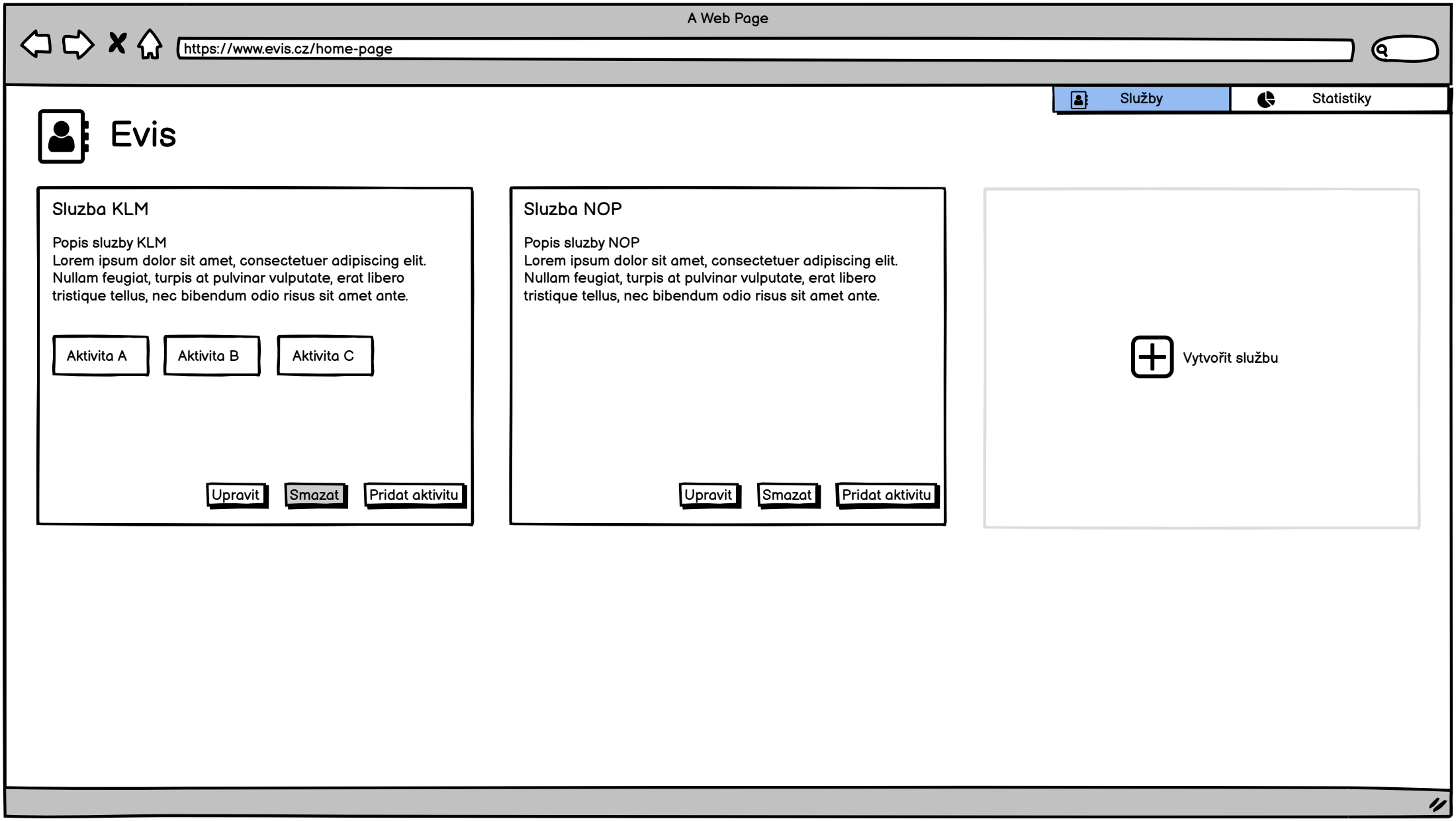
Příloha A

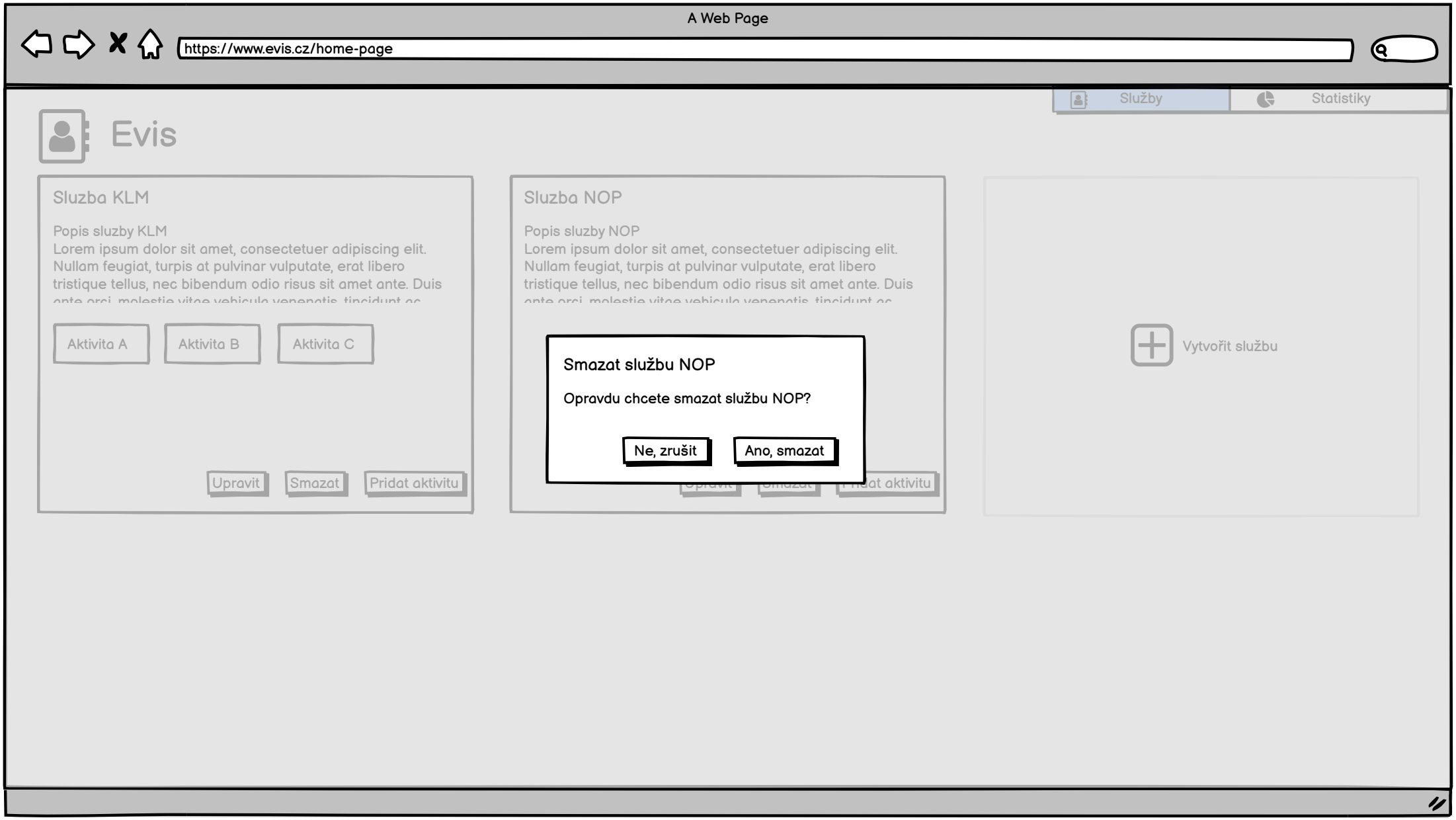
Papírový model

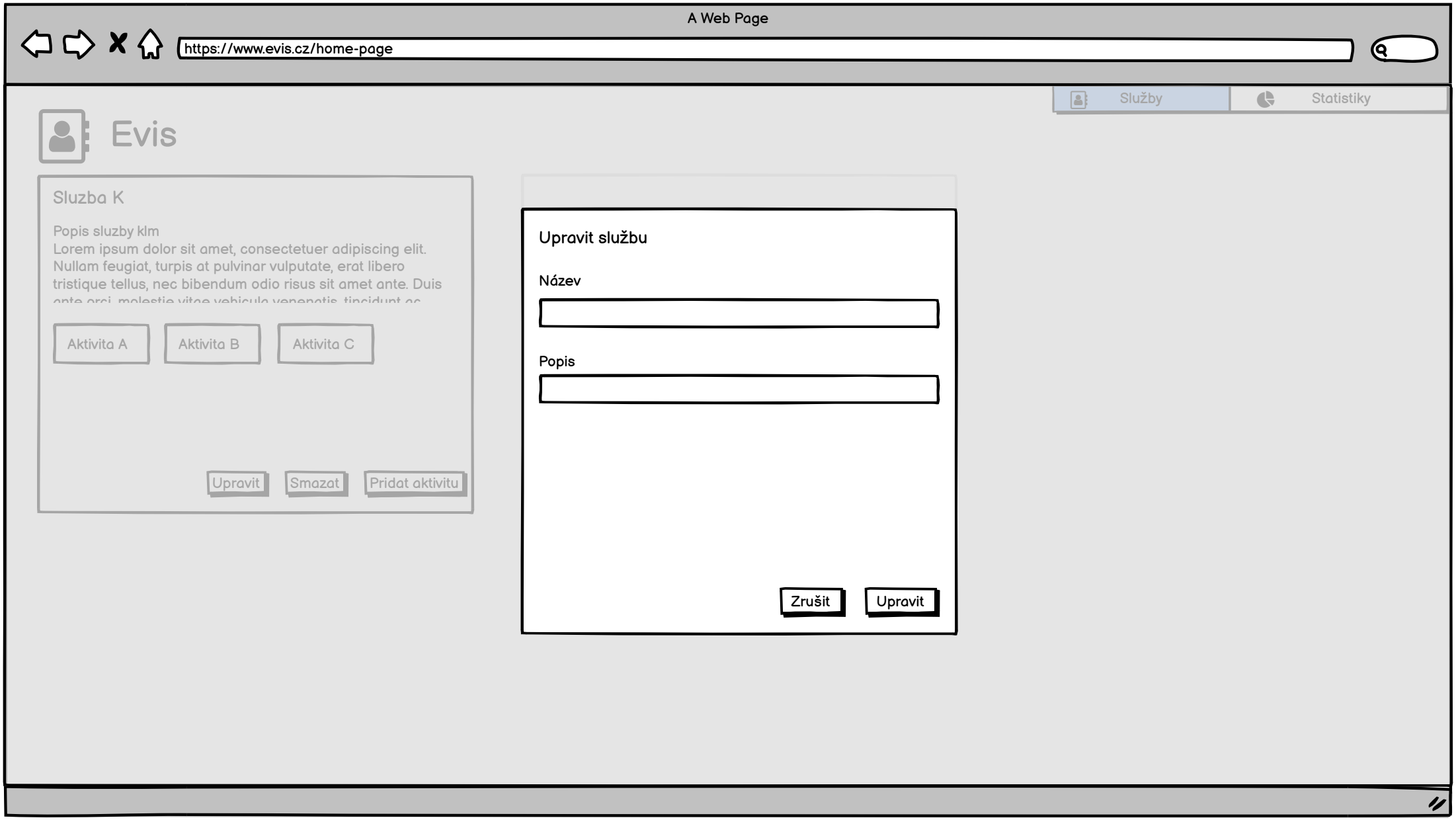

A Web Page
https://www.evis.cz/home-page
Služby
Statistiky
Evis
Sluzba KLM
Popis sluzby KLM
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Duis
Aktivita A
Aktivita B
Aktivita C
Upravit
Smazat
Přidat aktivitu
Vytvořit službu


A Web Page
https://www.evis.cz/home-page
Služby
Statistiky
Evis
Sluzba KLM
Popis sluzby KLM
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Duis
Aktivita A
Aktivita B
Aktivita D
Upravit
Smazat
Pridat aktivitu
Vytvořit službu
Název
Popis
Zrušit
Vytvořit


A Web Page
https://www.evis.cz/home-page
Služby
Statistiky
Evis
Sluzba KLM
Popis sluzby KLM
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Duis
Aktivita A
Aktivita I
Aktivita D
Upravit
Smazat
Pridat aktivitu
Přidat aktivitu
Název
Popis
Zrušit
Přidat


A Web Page
https://www.evis.cz/home-page
Služby
Statistiky
Evis
Sluzba KLM
Popis sluzby KLM
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante.
Aktivita A
Aktivita B
Aktivita C
Upravit
Smazat
Pridat aktivitu
Sluzba NOP
Popis sluzby NOP
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante.
Upravit
Smazat
Pridat aktivitu
Vytvořit službu


A Web Page
https://www.evis.cz/home-page
Služby
Statistiky
Evis
Sluzba KLM
Popis sluzby KLM
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Duis
Aktivita A
Aktivita B
Aktivita C
Upravit
Smazat
Pridat aktivitu
Sluzba NOP
Popis sluzby NOP
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Duis
Smazat službu NOP
Opravdu chcete smazat službu NOP?
Ne, zrušit
Ano, smazat
Vytvořit službu


A Web Page
https://www.evis.cz/home-page
Služby
Statistiky
Evis
Sluzba K
Popis sluzby klm
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Duis
Aktivita A
Aktivita B
Aktivita C
Upravit
Smazat
Pridat aktivitu
Upravit službu
Název
Popis
Zrušit
Upravit

A Web Page

https://www.evis.cz/activity/1

Služby | Statistiky

Evis | Aktivita A

Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.

Evidovat výkon

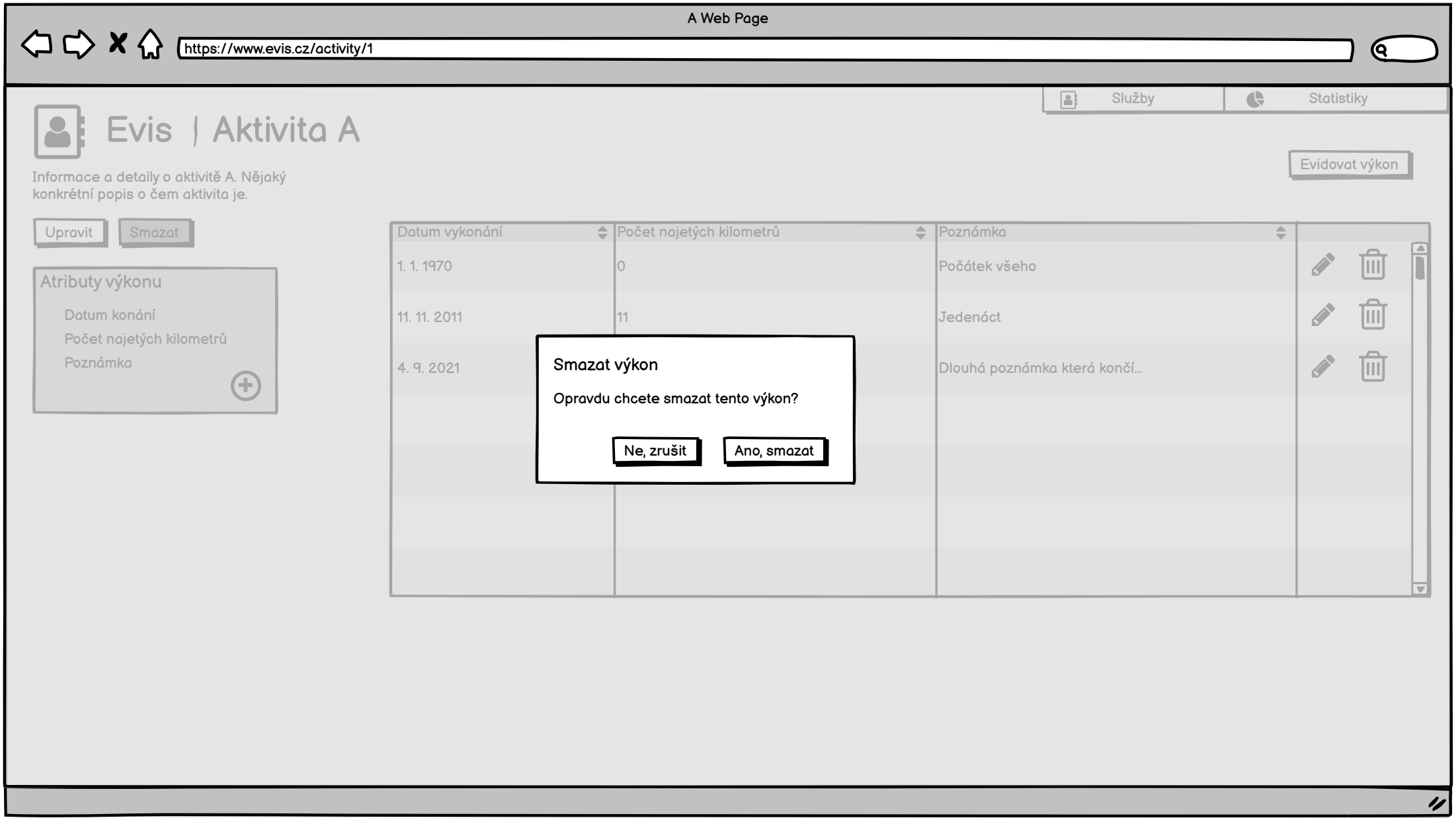
Upravit | Smazat

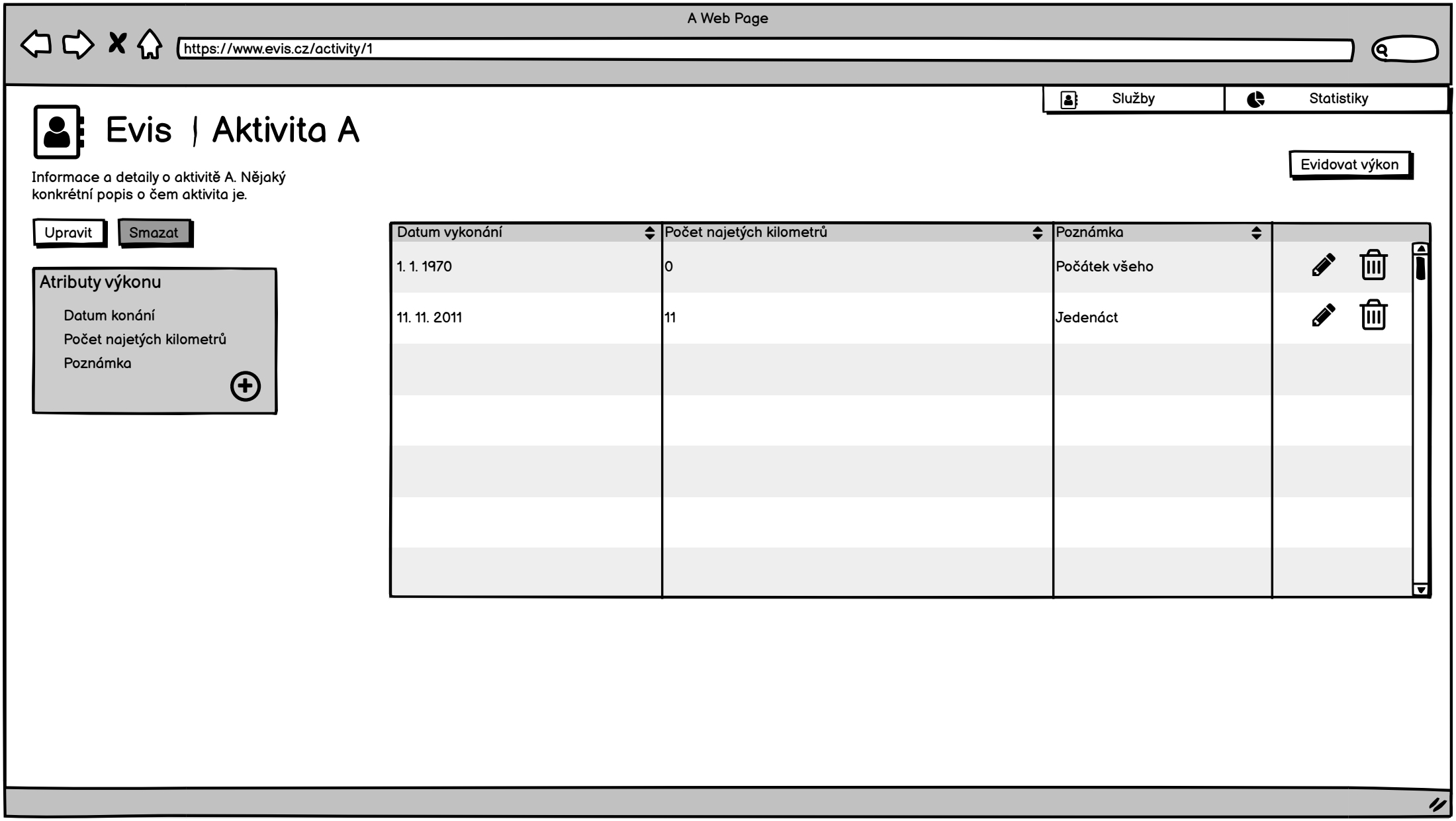
Atributy výkonu

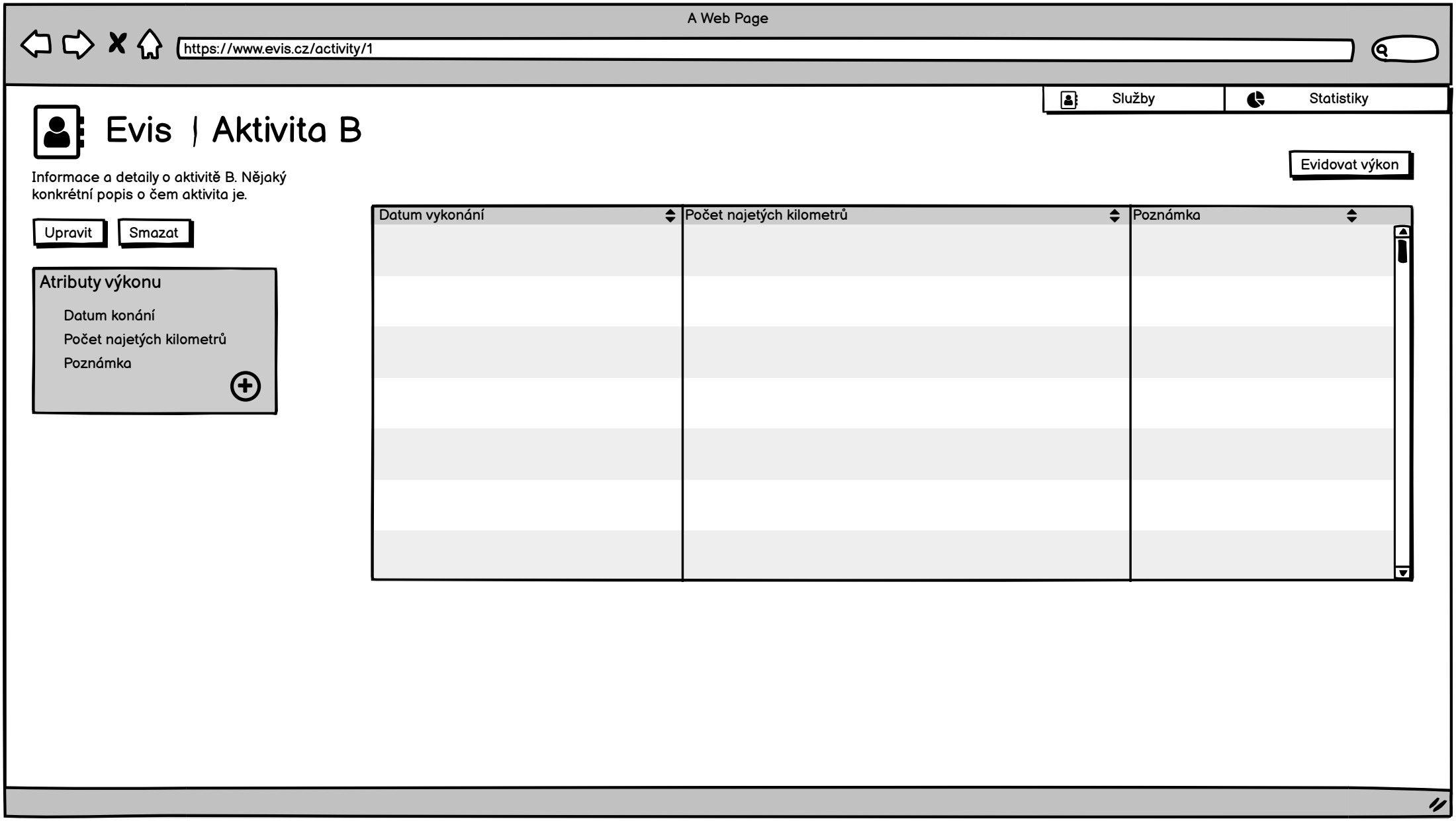
- Datum konání
- Počet najetých kilometrů
- Poznámka

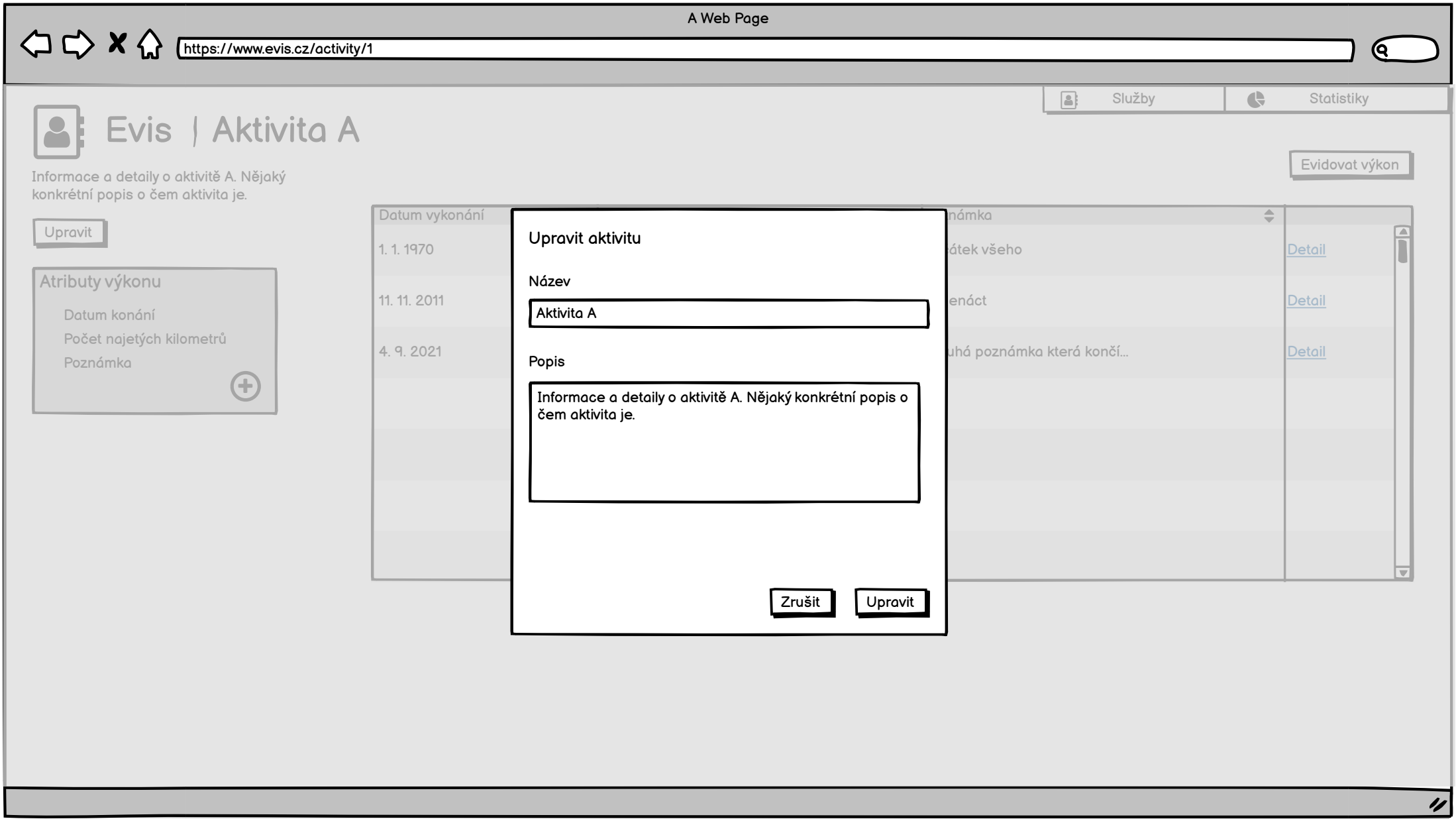
⊕

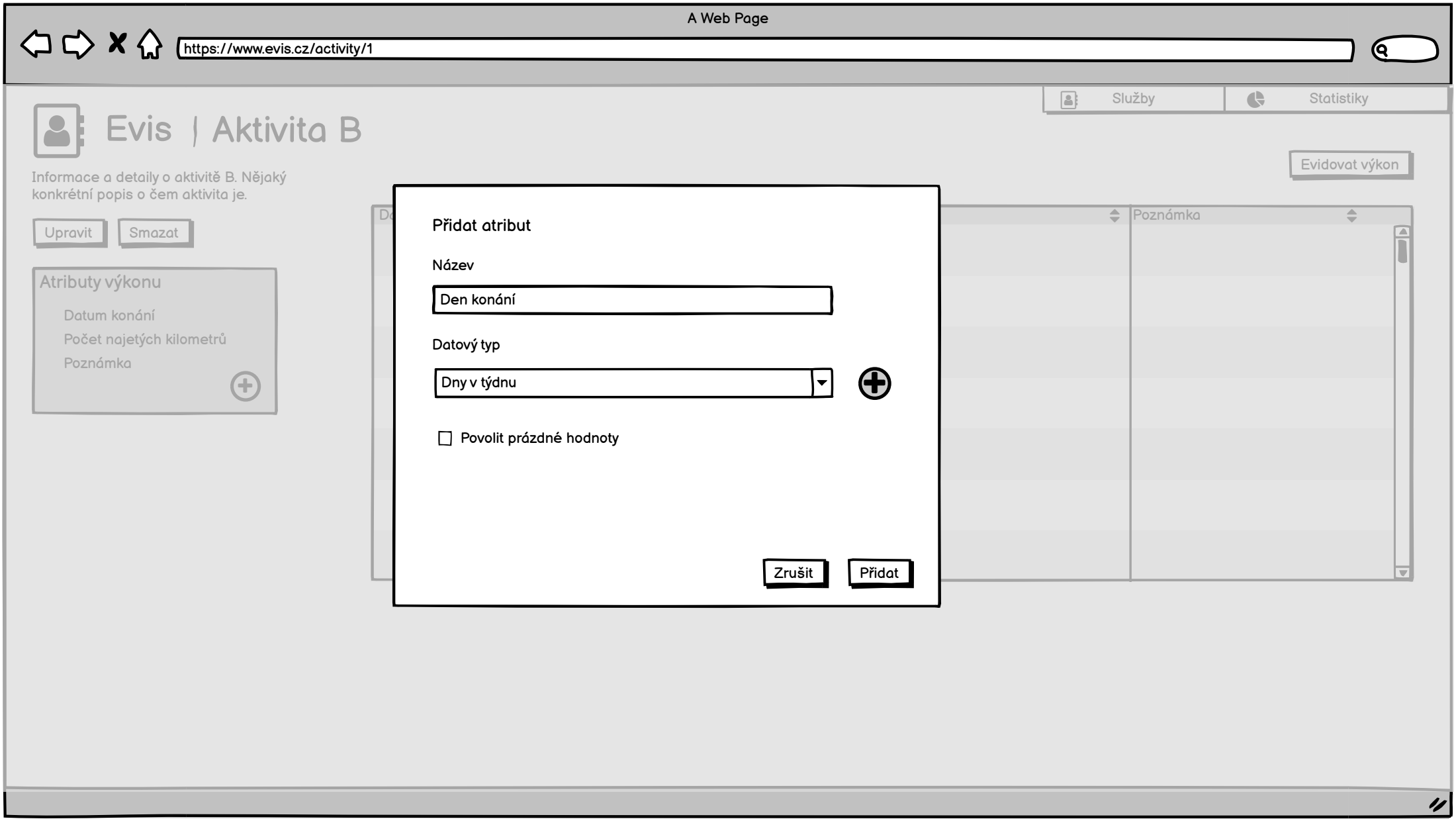
Datum vykonání	Počet najetých kilometrů	Poznámka	
1. 1. 1970	0	Počátek všeho	✎ 🗑
11. 11. 2011	11	Jedenáct	✎ 🗑
4. 9. 2021	69	Dlouhá poznámka která končí...	✎ 🗑

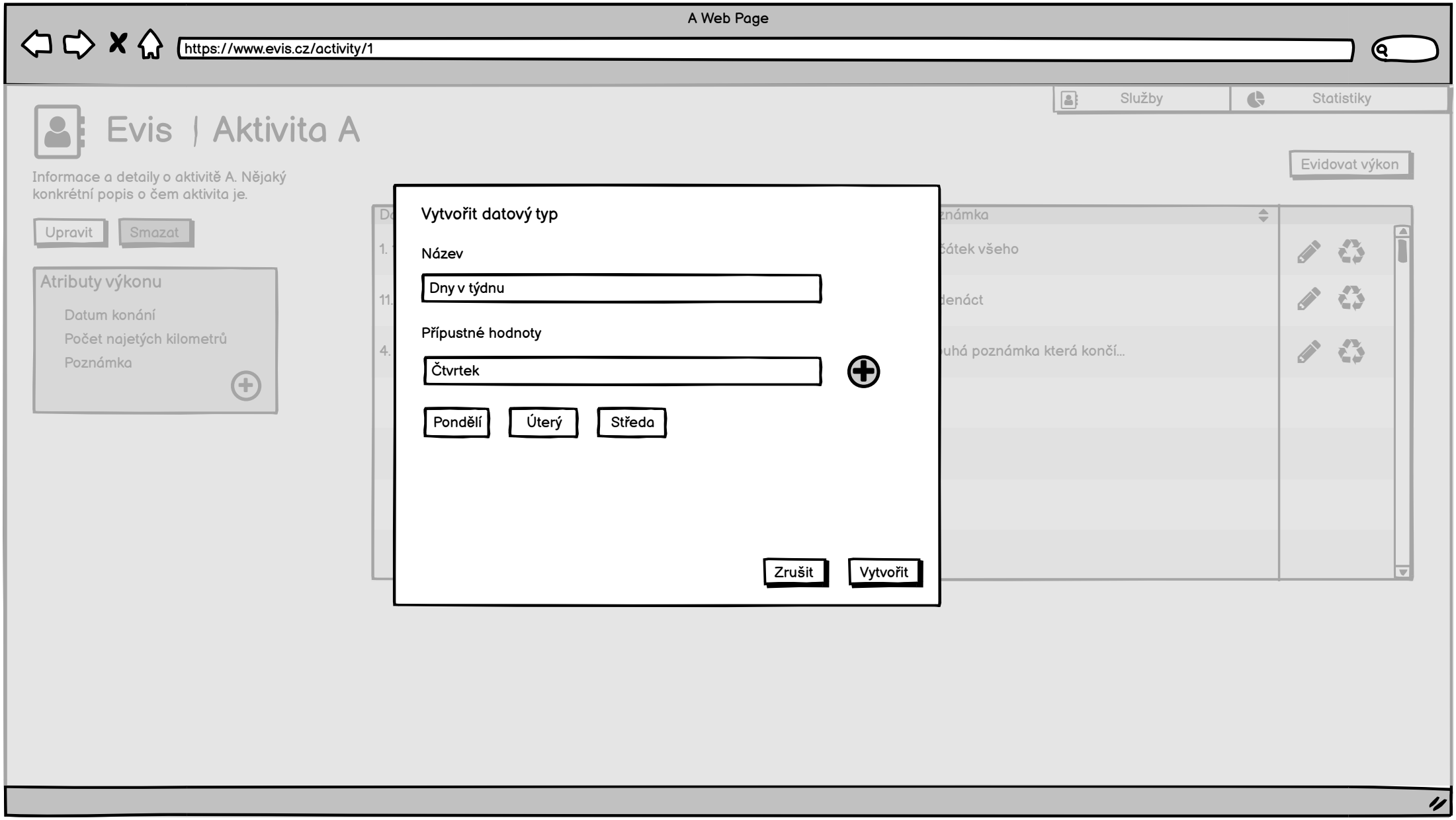

A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Evidovat výkon
Upravit
Smazat
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Datum vykonání
Počet najetých kilometrů
Poznámka
1. 1. 1970
0
Počátek všeho
11. 11. 2011
11
Jedenáct
4. 9. 2021
Dlouhá poznámka která končí...
Smazat výkon
Opravdu chcete smazat tento výkon?
Ne, zrušit
Ano, smazat

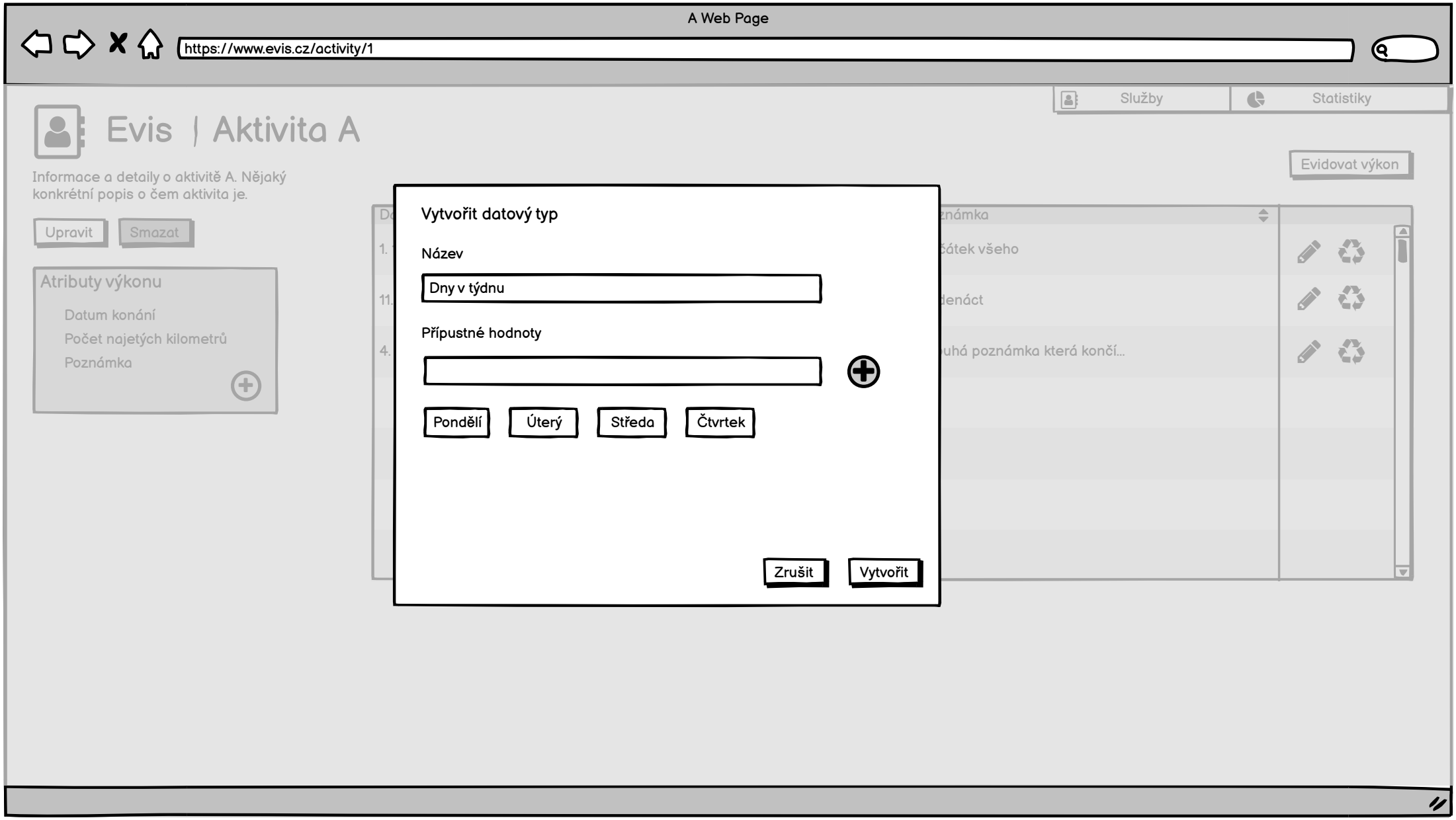

A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Evidovat výkon
Upravit
Smazat
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Datum vykonání
Počet najetých kilometrů
Poznámka
1. 1. 1970
0
Počátek všeho
11. 11. 2011
11
Jedenáct

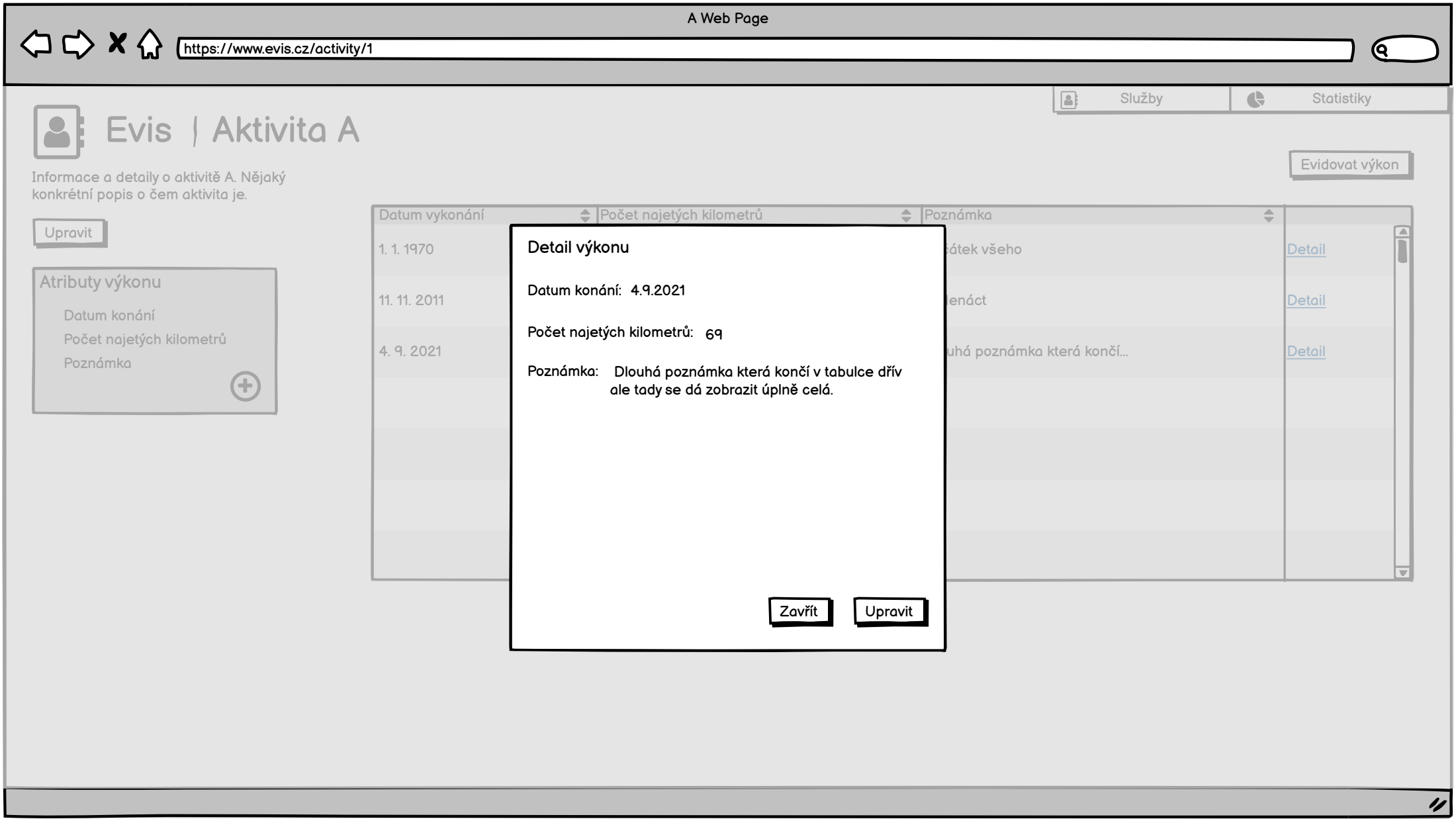

A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita B
Informace a detaily o aktivitě B. Nějaký konkrétní popis o čem aktivita je.
Upravit
Smazat
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Evidovat výkon
Datum vykonání
Počet najetých kilometrů
Poznámka

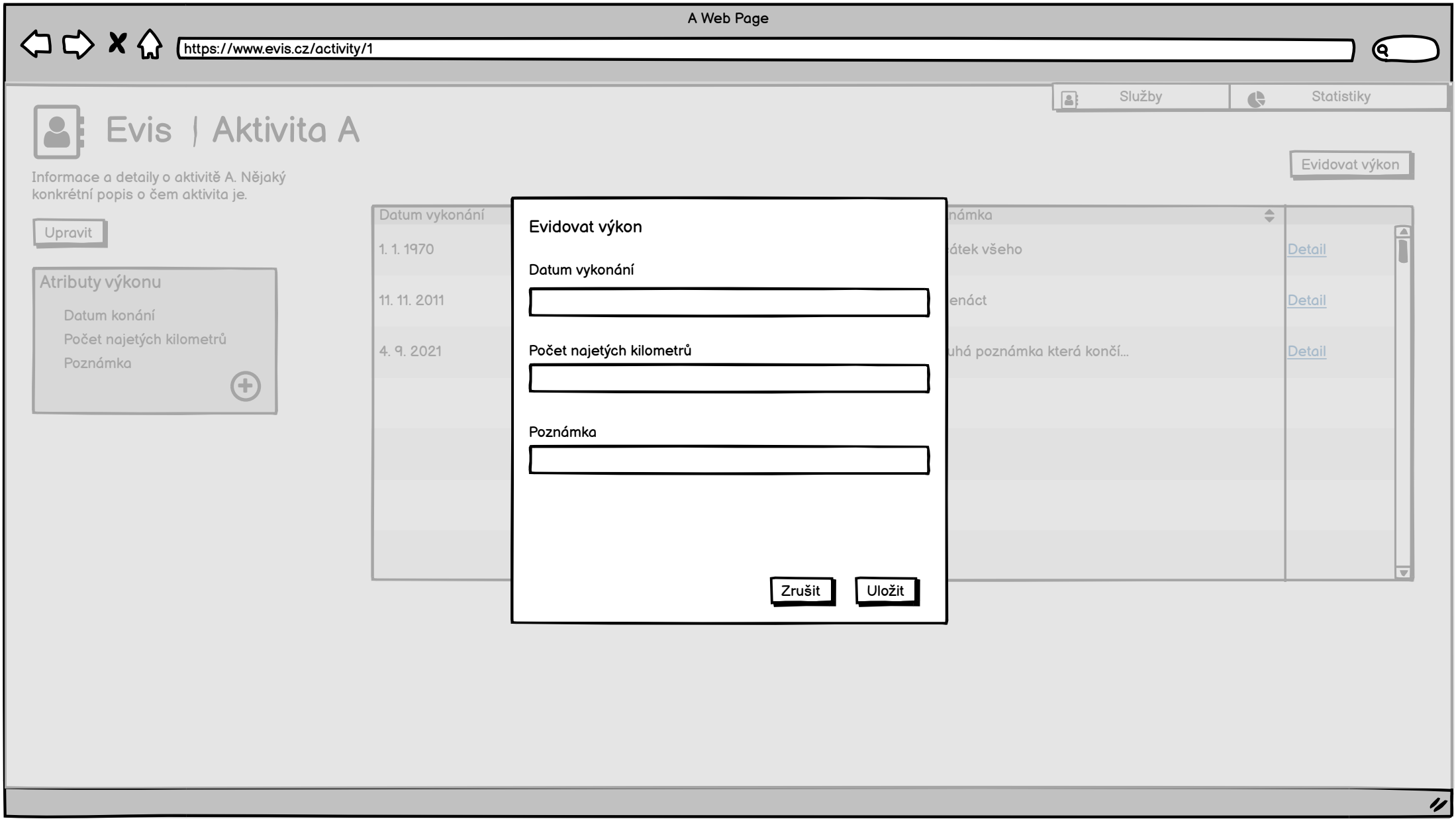

A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Upravit
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Evidovat výkon
Datum vykonání
1. 1. 1970
11. 11. 2011
4. 9. 2021
Detail
Upravit aktivitu
Název
Aktivita A
Popis
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Zrušit
Upravit

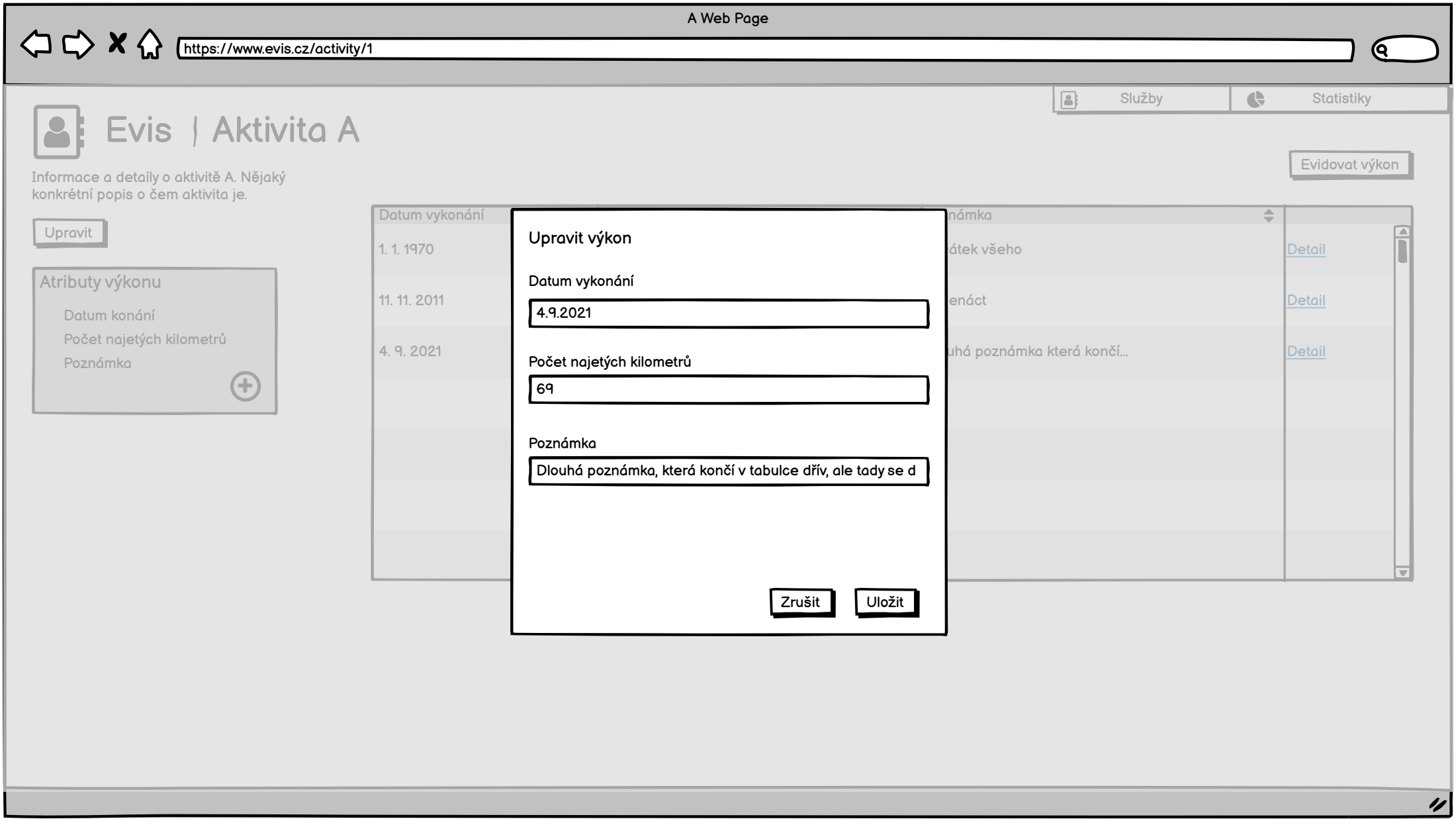

A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita B
Informace a detaily o aktivitě B. Nějaký konkrétní popis o čem aktivita je.
Upravit
Smazat
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Evidovat výkon
Poznámka
Přidat atribut
Název
Den konání
Datový typ
Dny v týdnu
Povolit prázdné hodnoty
Zrušit
Přidat


A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Upravit
Smazat
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Evidovat výkon
Vytvořit datový typ
Název
Dny v týdnu
Přípustné hodnoty
Čtvrtek
Pondělí
Úterý
Středa
Zrušit
Vytvořit

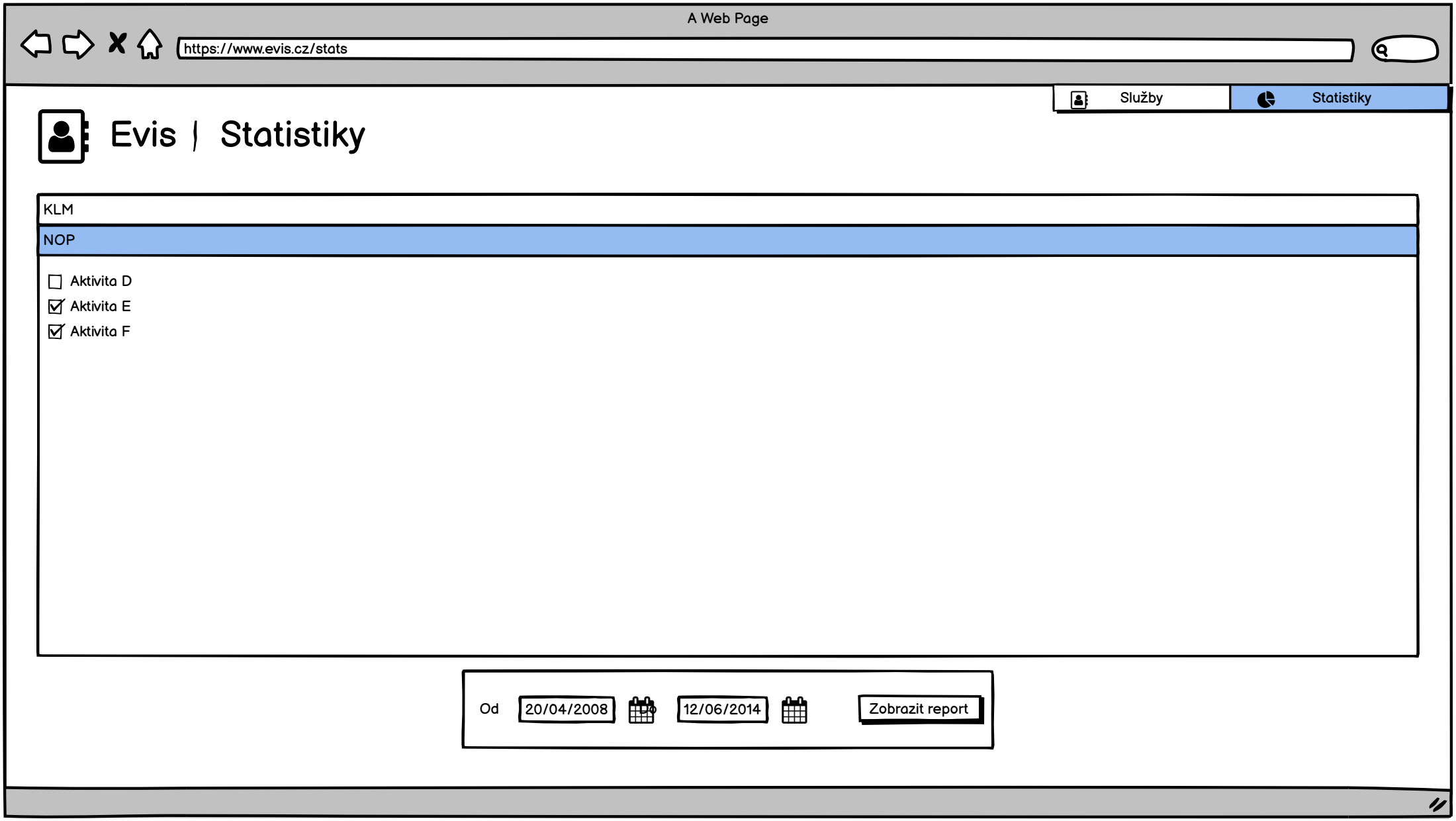

A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Upravit
Smazat
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Evidovat výkon
Poznámka
Vytvořit datový typ
Název
Dny v týdnu
Přípustné hodnoty
Pondělí
Úterý
Středa
Čtvrtek
Zrušit
Vytvořit


A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Upravit
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Evidovat výkon
Datum vykonání
Počet najetých kilometrů
Poznámka
1. 1. 1970
Detail
11. 11. 2011
Detail
4. 9. 2021
Detail
Detail výkonu
Datum konání: 4.9.2021
Počet najetých kilometrů: 69
Poznámka: Dlouhá poznámka která končí v tabulce dřív ale tady se dá zobrazit úplně celá.
Zavřít
Upravit


A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Upravit
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Evidovat výkon
Datum vykonání
1. 1. 1970
11. 11. 2011
4. 9. 2021
Detail
Detail
Detail
Evidovat výkon
Datum vykonání
Počet najetých kilometrů
Poznámka
Zrušit
Uložit


A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Evis | Aktivita A
Informace a detaily o aktivitě A. Nějaký konkrétní popis o čem aktivita je.
Upravit
Evidovat výkon
Atributy výkonu
Datum konání
Počet najetých kilometrů
Poznámka
Datum vykonání
1. 1. 1970
11. 11. 2011
4. 9. 2021
Detail
Upravit výkon
Datum vykonání
4.9.2021
Počet najetých kilometrů
69
Poznámka
Dlouhá poznámka, která končí v tabulce dřív, ale tady se d
Zrušit
Uložit


A Web Page
https://www.evis.cz/stats
Služby
Statistiky
Evis | Statistiky
KLM
Aktivita A
Aktivita B
Aktivita C
NOP
Od
20/04/2008
12/06/2014
Zobrazit report


A Web Page
https://www.evis.cz/stats
Služby
Statistiky
Evis | Statistiky
KLM
NOP
Aktivita D
Aktivita E
Aktivita F
Od
20/04/2008
12/06/2014
Zobrazit report

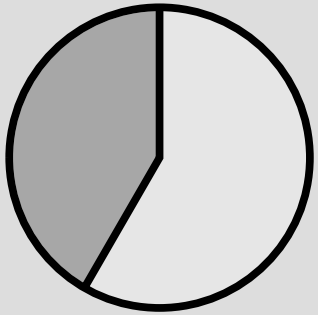
A Web Page

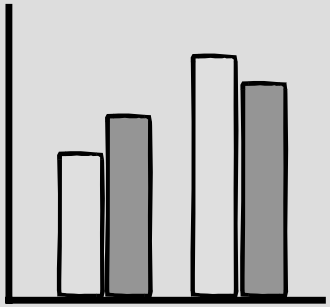
https://www.evis.cz/report

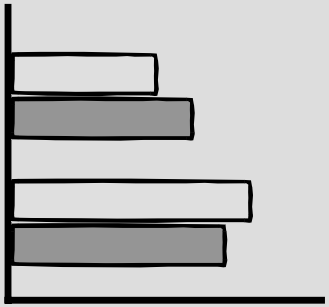
Služby | Statistiky

Evis | Report

Zpráva 20.04.2008 - 12.06.2014







Stáhnout

A Web Page

https://www.evis.cz/employees

Zaměstnanci

Přidat nového zaměstnance | Atributy zaměstnance

Jméno	Datum narození	Upozornění
Jan Amos Komenský	28. března 1592	☑
Ema Destinnová	26. února 1878	☐
Karel IV.	14. května 1316	☐
Bedřich Smetana	2. března 1824	☑
Václav Havel	5. října 1936	☑
Božena Němcová	4. února 1820	☑
Marie Terezie	13. května 1717	☐

Jméno: Václav
Příjmení: Havel
Datum narození: 5. října 1936
Platnost lékařské prohlídky: 4. listopadu 2021
Typ smlouvy: Hlavní pracovní poměr
Platnost pracovní smlouvy: -
Platnost školení BOZP a PO: 4. listopadu 2021
Platnost školení řidičů referend: 4. listopadu 2021
Zdravotní pojišťovna:
Rodné číslo:
Místo narození:
Bydliště, telefon, e-mail,...
Služby: Sociálně aktivizační služba pro rodiny s dětmi sv. Rity
Výdejna Samaritán
Noclehárna sv. Antonína
Role: .....
Vzdělání: .....
Další vzdělávání (kurzy,...): .....

Odstranit | Upravit

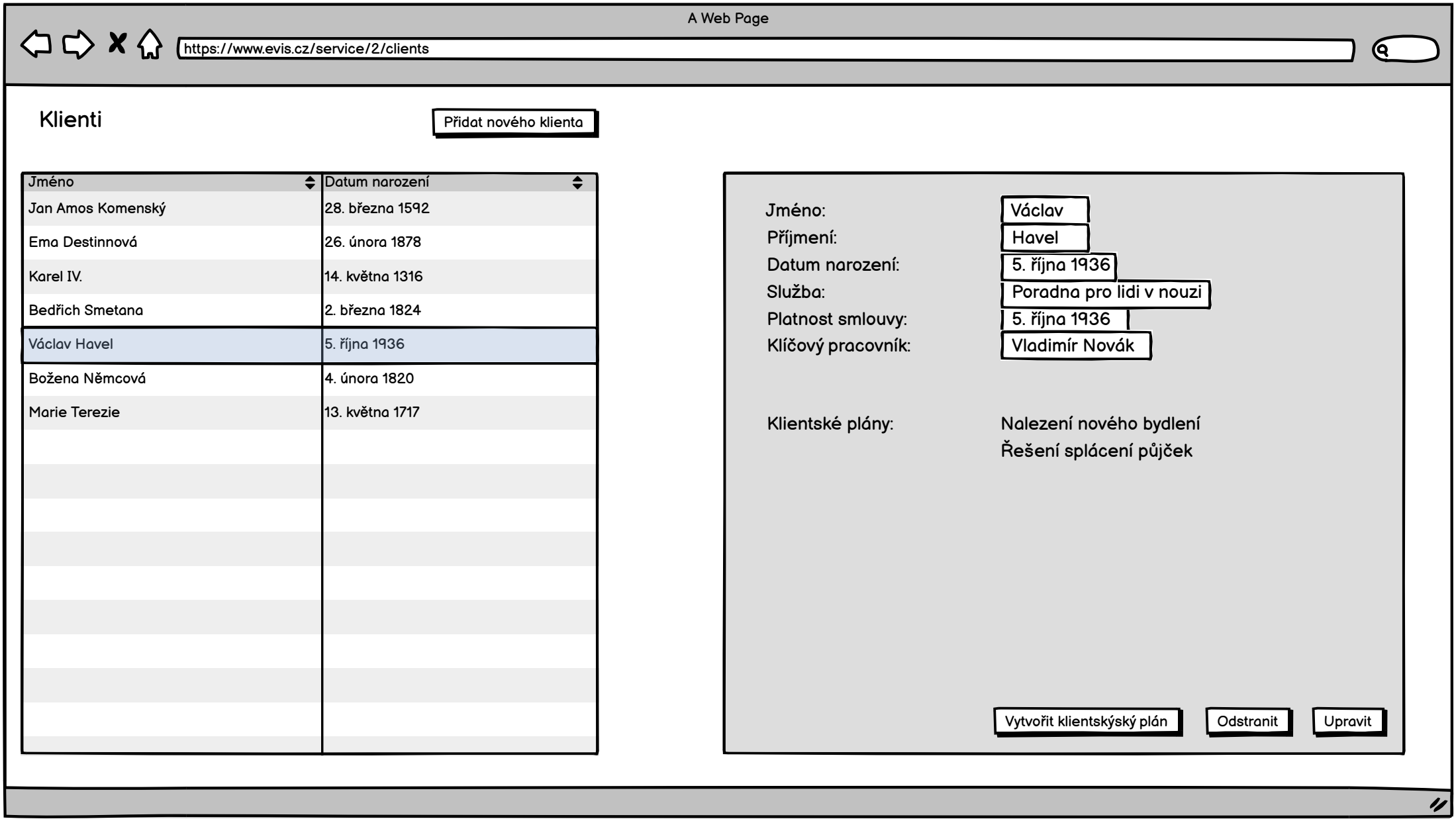
A Web Page

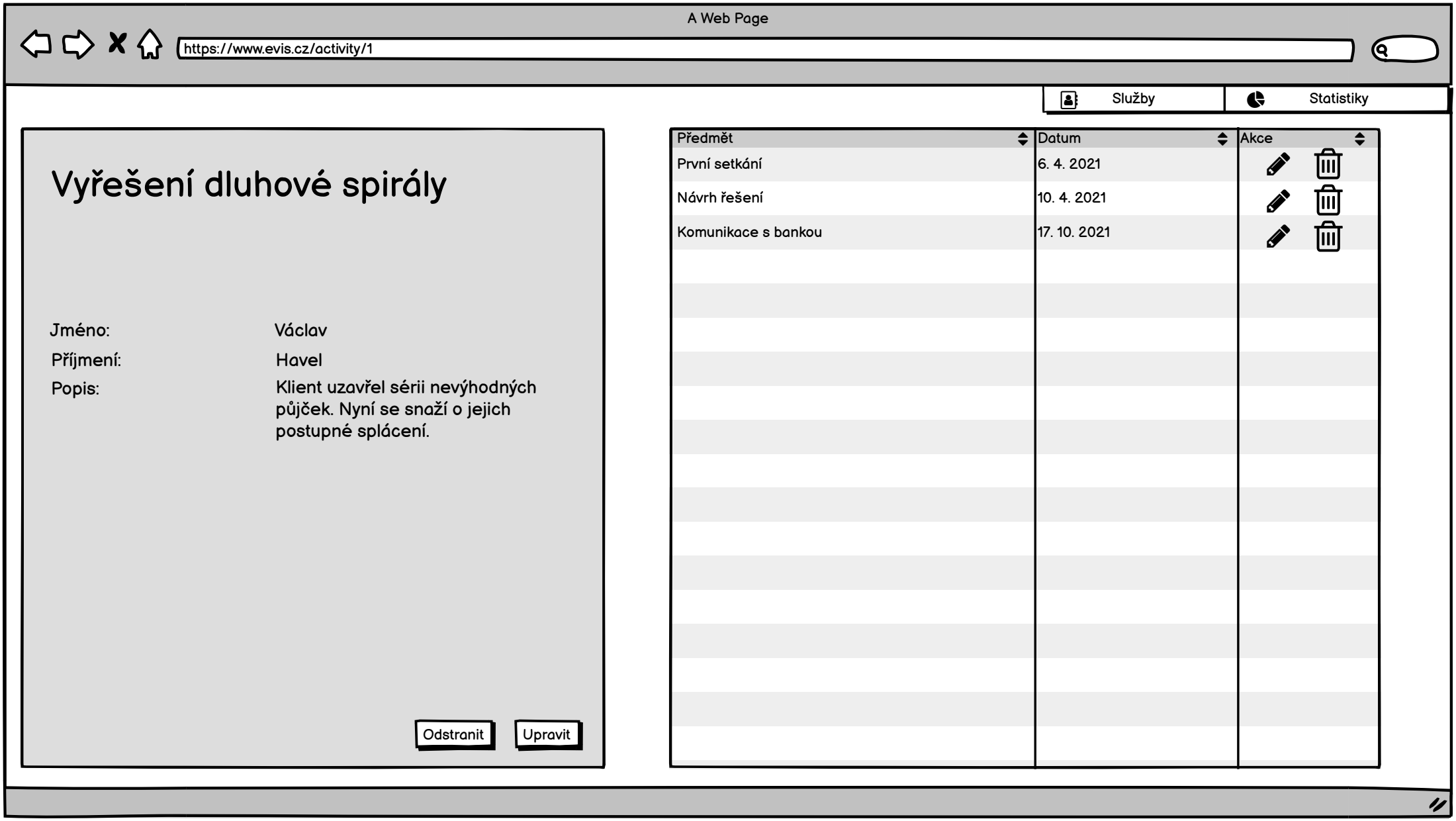
https://www.evis.cz/employees

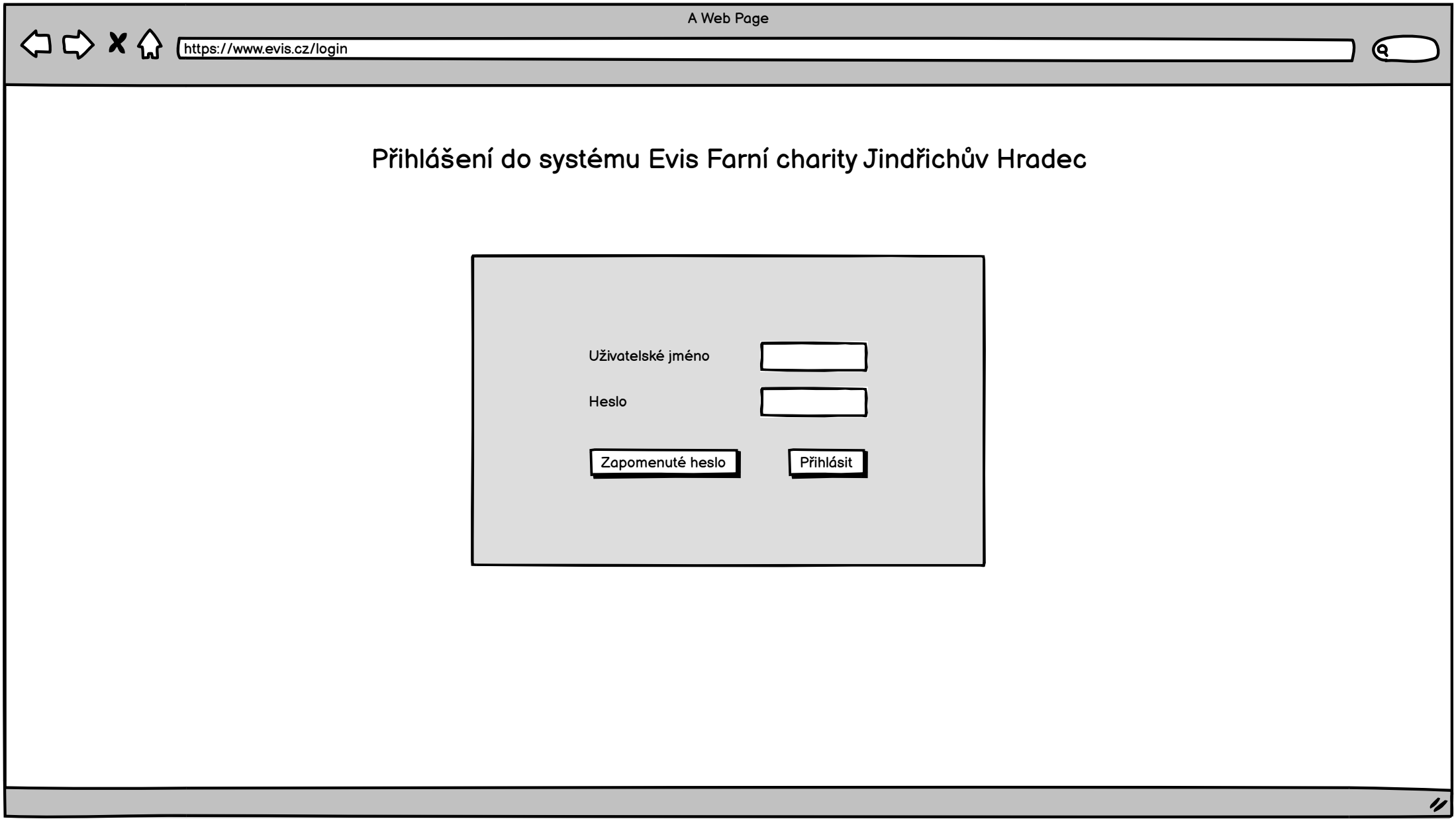
Můj profil

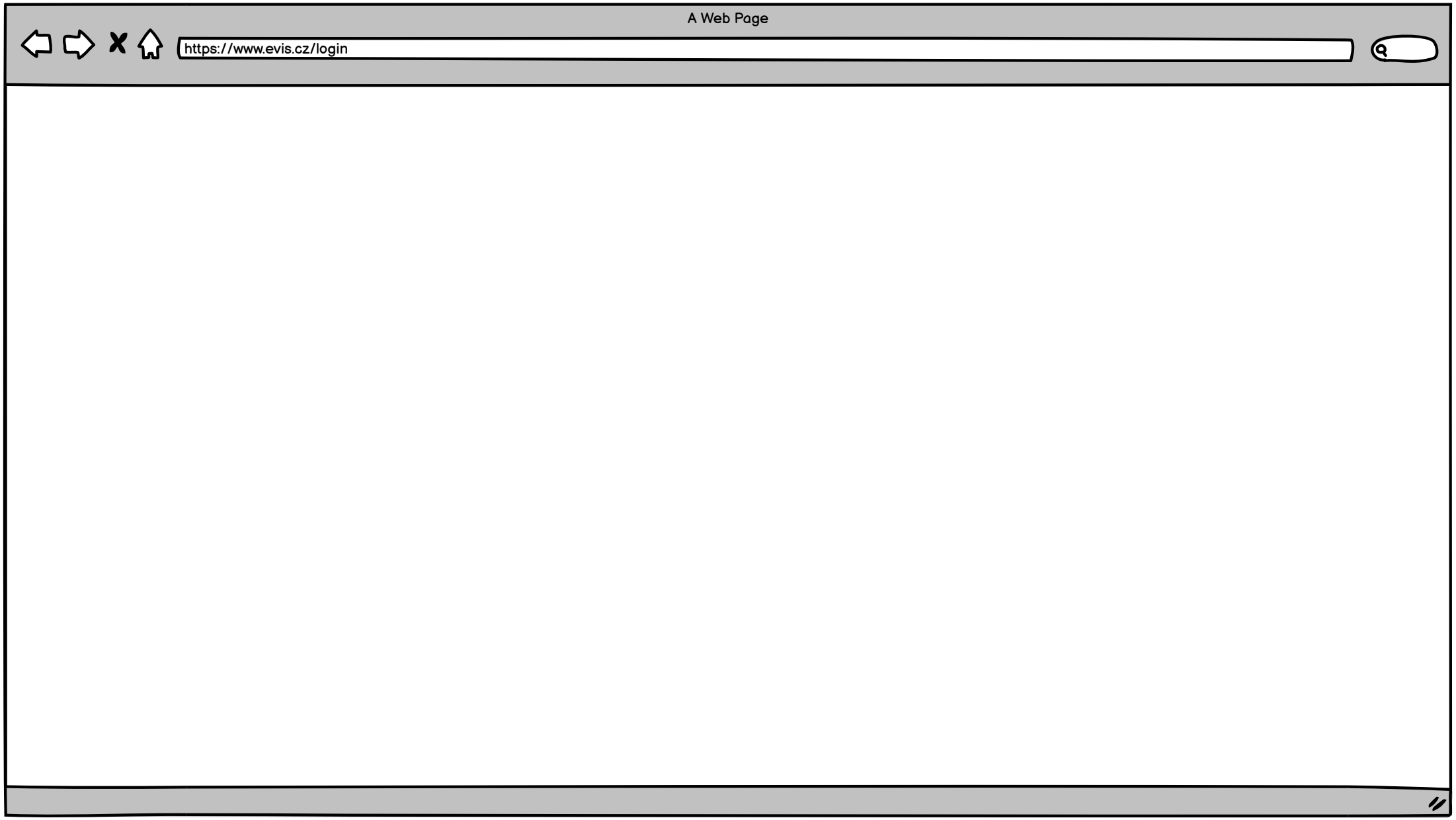
Jméno:	Václav
Příjmení:	Havel
Datum narození:	5. října 1936
Platnost lékařské prohlídky:	4. listopadu 2021
Typ smlouvy:	Hlavní pracovní poměr
Platnost pracovní smlouvy:	-
Platnost školení BOZP a PO:	4. listopadu 2021
Platnost školení řidičů referend:	4. listopadu 2021
Zdravotní pojišťovna:	
Rodné číslo:	
Místo narození:	
Bydliště, telefon, e-mail,...	
Služby:	Sociálně aktivizační služba pro rodiny s dětmi sv. Rity Výdejna Samaritán Noclehárna sv. Antonína
Role:	.....
Vzdělání:	.....
Další vzdělávání (kurzy,...):	.....

Změnit heslo | Upravit


A Web Page
https://www.evis.cz/service/2/clients
Klienti
Přidat nového klienta
Jméno
Datum narození
Jan Amos Komenský
28. března 1592
Ema Destinnová
26. února 1878
Karel IV.
14. května 1316
Bedřich Smetana
2. března 1824
Václav Havel
5. října 1936
Božena Němcová
4. února 1820
Marie Terezie
13. května 1717
Jméno:
Václav
Příjmení:
Havel
Datum narození:
5. října 1936
Služba:
Poradna pro lidi v nouzi
Platnost smlouvy:
5. října 1936
Klíčový pracovník:
Vladimír Novák
Klientské plány:
Nalezení nového bydlení
Řešení splácení půjček
Vytvořit klientskýský plán
Odstranit
Upravit


A Web Page
https://www.evis.cz/activity/1
Služby
Statistiky
Vyřešení dluhové spirály
Jméno:
Václav
Příjmení:
Havel
Popis:
Klient uzavřel sérii nevýhodných půjček. Nyní se snaží o jejich postupné splácení.
Odstranit
Upravit
Předmět
Datum
Akce
První setkání
6. 4. 2021
Návrh řešení
10. 4. 2021
Komunikace s bankou
17. 10. 2021

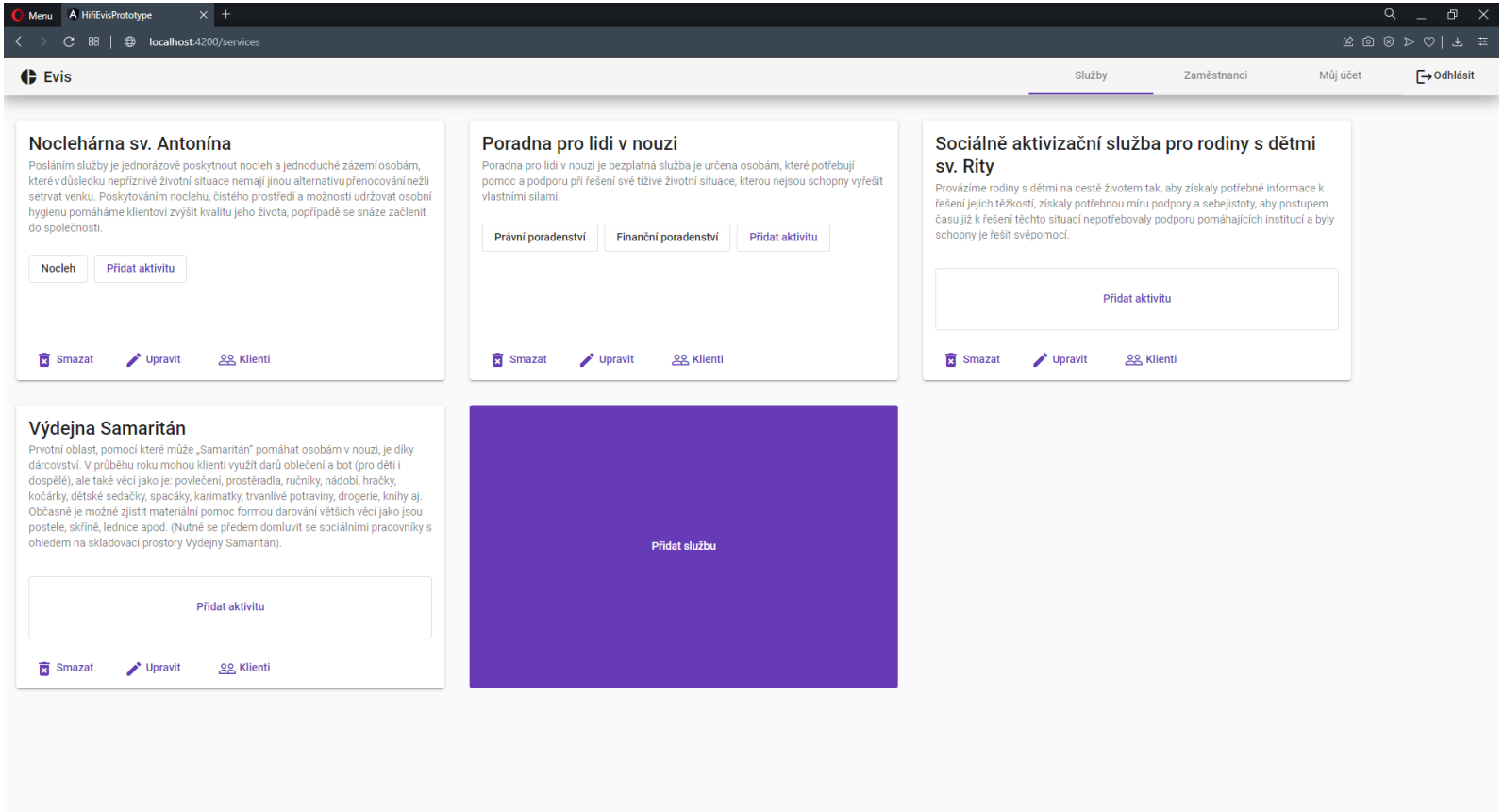

A Web Page
https://www.evis.cz/login
Přihlášení do systému Evis Farní charity Jindřichův Hradec
Uživatelské jméno
Heslo
Zapomenuté heslo
Přihlásit


A Web Page
https://www.evis.cz/login

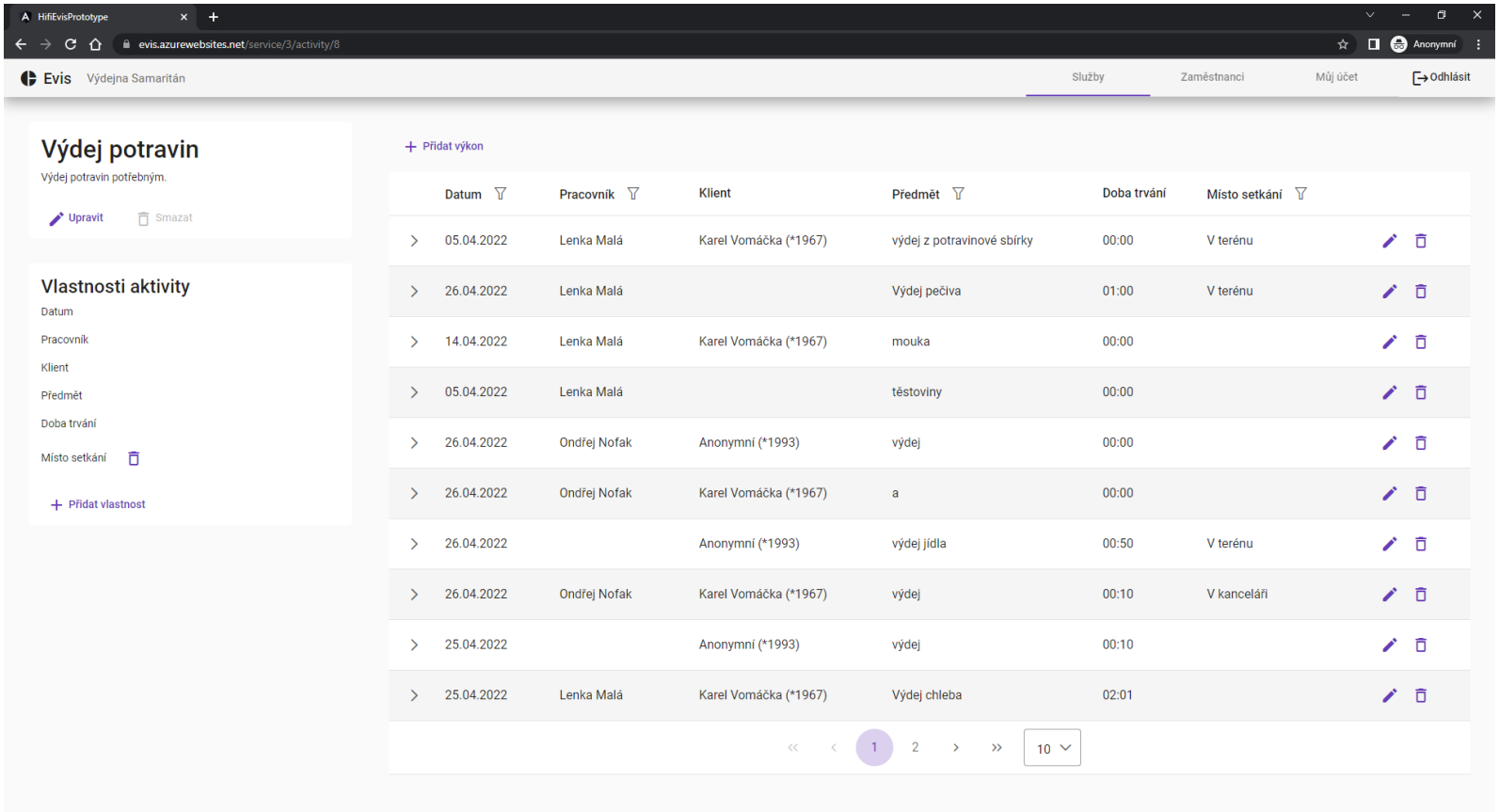
Příloha B

Snímky obrazovky prostředí Evis

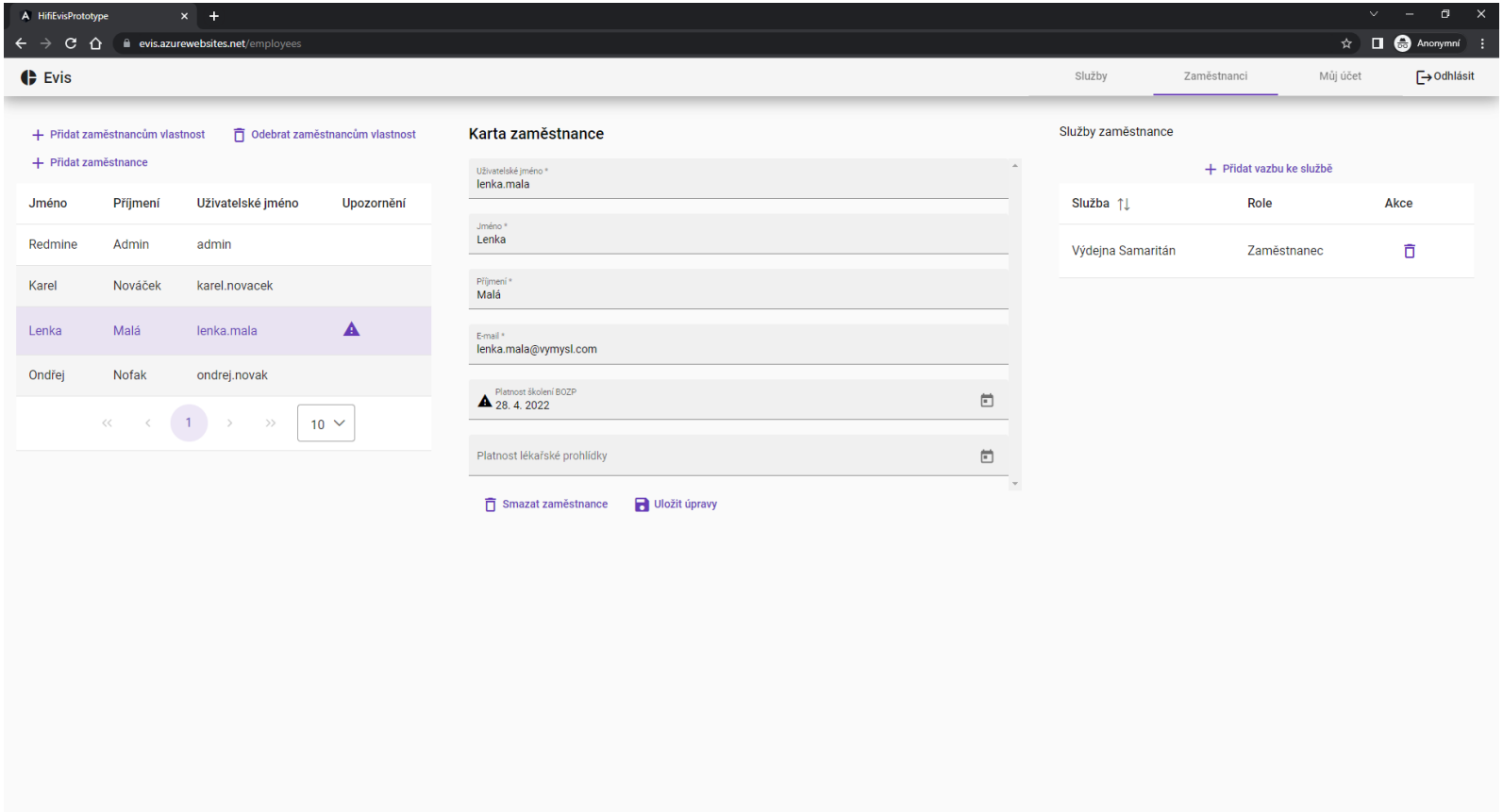
Obrázek B.1: Přihlašovací obrazovka



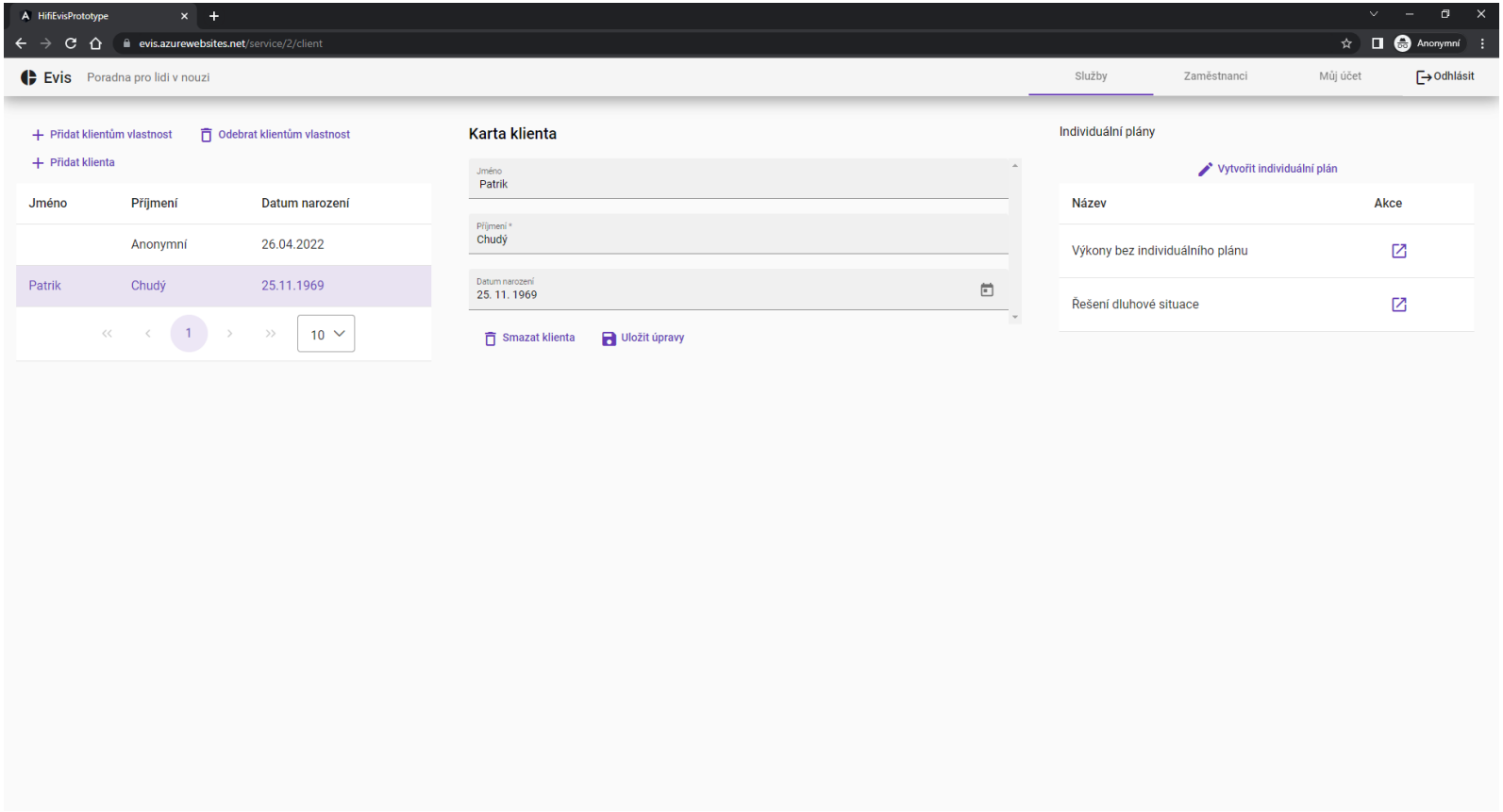
Obrázek B.2: Domovská stránka



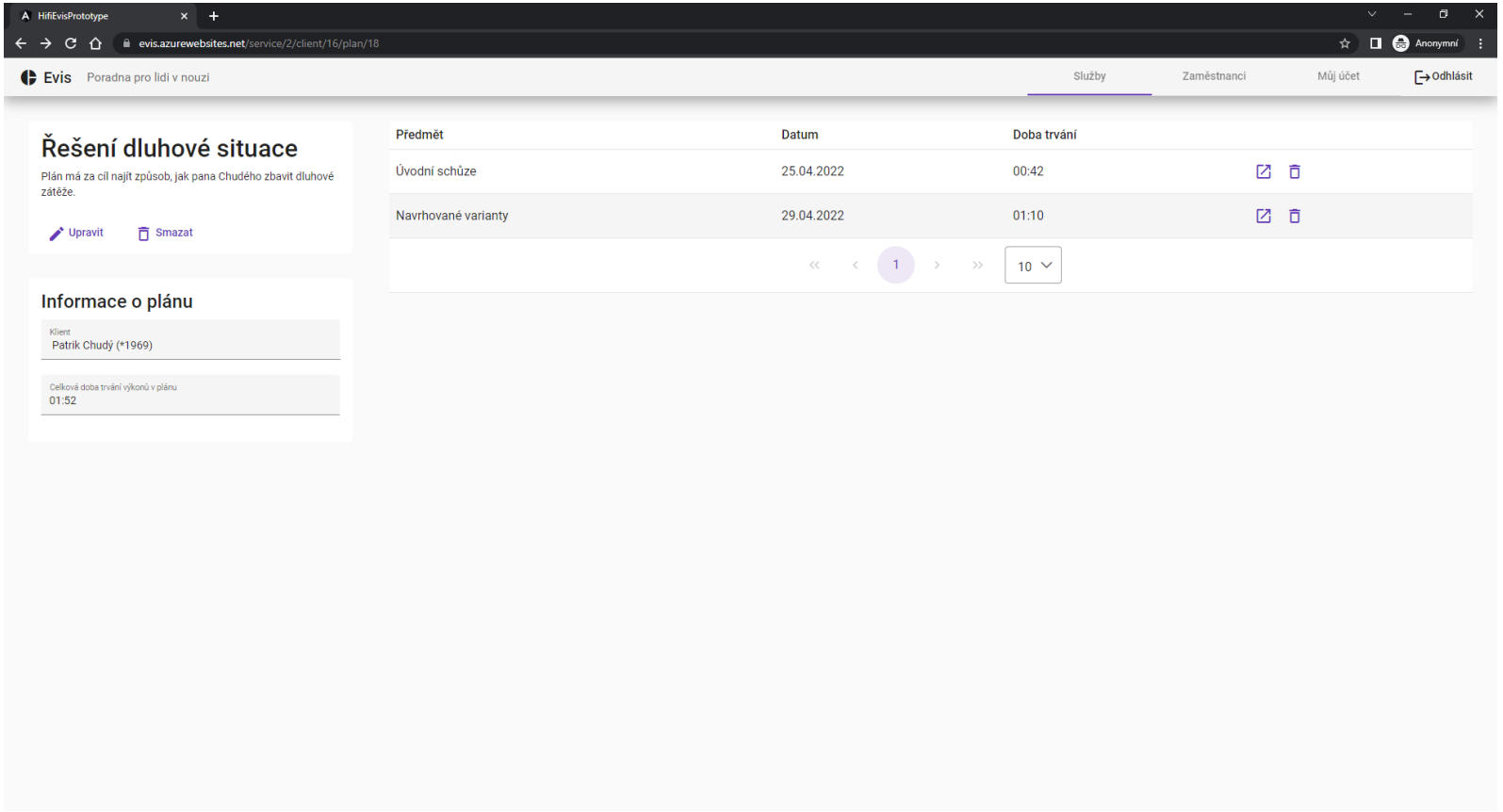
Obrázek B.3: Detail aktivity



Obrázek B.4: Zaměstnanci



Obrázek B.5: Klienti



Obrázek B.6: Individuální plán

Příloha C

Autentické poznámky ze schůzek s vedoucím práce Ing. Jiřím Hunkou

28.09.2021

Vymyslet anglicky nazev

Vymyslet, proc delat tohle a nepouzit neco existujiciho

Udelat podrobnou analyzu existujici

Pripravit si na NUR navrh projektu, abychom to mohli delat v ramci semestralky

Zkonzultovat s reditelkou ty veci od statu

Systemy vlastni nebo uz existujici nebo uz pouzivany

07.10.2021

Příprava

- zjistili jsme, že Redmine je s úpravami a pluginy velmi pravděpodobně použitelný pro daný problém
- potřebujeme zjistit, jakým způsobem dále pokračovat vzhledem k diplomkám a reálnému

06.12.2021

stáhnout si apku redminu na android
zeptat se na fóru

- trackery musí mít někde možnost, jak přes api zjistit všechny fieldy, prostě musí, najdu to
- kdyby náhodou ne, tak se zeptat na fóru, stáhnout si existující apku, která s tím pracuje nebo použít všechno custom fieldy

k trackerům určitého projektu lze přistupovat přes
http://localhost/redmine/projects/2.xml?include=trackers
a custom fieldy vidím zase v
http://localhost/redmine/custom_fields.xml?include=trackers

takže dokážu mít výběr aktivit pro jednotlivé služby a u aktivit můžu mít také atributy v podobě custom fieldů, ostatní fieldy asi budou vždy stejné a ty custom budou rozšiřovat

neumím teda ještě CRUD operace s trackery

03.02.2022

Příprava

1. zadání
    a. jak řešit navázání na Davida po formální stránce
2. u Redminu jsem si přidal některé endpointy, některé si ještě přidám, testuju si API přes SoapUI, už jsem si i zkoušel přistupovat k API přes Angular, který ještě rozšířím

Hovor

1. přečíst si Davidovu práci
2. pokračovat v propojování
3. začít něco psát
4. kontaktovat charitu a taky se domluvit

21.02.2022

Příprava

- ukázat pokrok v sw řešení i textu, zmínit schůzku s Kájou
- u textu se zeptat
    - jak se správně cituje přes latex jiná práce
    - jak odkazovat na Davida z pohledu titulu a jak být jinak při oslovování formální
        - on mě netituloval, tak snad je to bez titulu taky v pořádku
    - chci dodržet podobnou strukturu jako má David
    - ujistit se, co všechno patří do analýzy z pohledu odkazů na současné řešení
- u kódu se zeptat
    - umí Redmine něco jako nástěnku
        - zatím mě napadá jen udělat nový projekt Nástěnka nebo použít Novinky, ale to je taky přes separátní projekt

nástěnka

- přidat rozšíření

Davida představit a odkazovat

15.03.2022

1. probrat diplomku
    a. řešit, jak hodně můžu psát to stejné, co psal David
    b. funkční požadavky možná napsat ve stejném výčtu jako David a zvýraznit změny
2. popsat stav kódu
3. zeptat se na oponenta
    a. je nějaká výhoda mít stejného, jako měl David? (Matoušek)

hi-fi prototyp můžu víc rozvést

- vysvětlit svůj přínos

komunikace se zadavatelem
- vypsat na začátku možné způsoby (s odkazem na souhrn způsobů) a jejich výhody a nevýhody a pak vybrat jedno

u slovníku vyjasnit referenci na Davida

růžovou na slovníku změnit

napsat na bpr, jak se odkazuje na jiné diplomky

zamyslet se na rozdělením analýzy a návrhu - předěl pro čtenáře
v analýze popsat aktuální stav, odkazovat na Davida a zmiňovat svůj přínos v těch částech

Matouška se v určitý moment zeptat, jestli by neměl zájem na tuhle práci a zmínit, že to je na Redminu

někam dát teorii k testování, může být až u samotného testování

22.03.2022
- reference na davida se stránkami v textu

01.04.2022

Zeptat se
- Jak to vidi s úpravami v aktuální verzi textu
- Kam zaradit vyber knihoven (analyza vs návrh vs implementace/realizace)

Hovor

funkční a nefunkční požadavky udělat podle SI1, udělat více podle definice (každý bod má ještě nějaký popis)

pak přidat use case se scénářema, kde si můžu hrát s rolema

diagram architektury component systému, příklad v SI2, udělat podle notace, obrázek dobrý, ale aby to odpovídalo nějaké notaci

u task graphu vyznačit startovní stav

rozebrat víc návrh, jak jsem postupoval, kolik jsem měl verzí, jestli je lo-fi, hi-fi, nějaké zdroje

výběr knihoven do implementace

Příloha D

Autentické poznámky ze schůzek s ředitelkou Farní charity Jindřichův Hradec Mgr. Karolínou Píchovou, DiS.

25.09.2021

karta klienta
- bydliště, jméno,...
- uzavření smlouvy
  - upomínky k tomu

plán klienta
- najít bydlení
- zlepšení si školní prospěch

karta zaměstnanec
- osobní údaje
- lékařská prohlídka,...
- zase připomenutí třeba lékařské prohlídky, konec smlouvy a podobně

práva podle sekcí, uživatelů a podobně (noclehárna, nízkoprahové,...)
- zaměstnanec je v 0-N skupinách/sekcích

Otazky
Klient ve vice sekcich, zamestnanci vidi na celou kartu nebo jen na jeji cast, ktera se vztahuje k dane sekci?

Jaké custom typy jsou potřeba? Stačí jen enum?

11.10.2021

Informovali jsme Káju o situaci.

Kája souhlasí, že je lepší z jejího pohledu použít něco existujícího a upravit to.

08.12.2021

eQuip
- moc složitý
- “archivní” klienti
  - ideálně nastavit, aby to dávalo do archivu samo
- 4 typy klientů
  - nový klient
    - formulář na nového klienta je hrozně velký
  - existující klient
  - archivnní klient (vypršela smlouva - nějaký čas od posledního výkonu)
  - anonymní klienti
- mít nějaký události vzhledem k nějakému uživatelskému cíli

16.02.2022

Příprava

1. Zjistit, co je potreba pridat za obrazovky
    a. Je potreba nejak vic resit klienty a zamestnance, ale zalezi jak
2. Zjistit, jak dobre je potreba filtrovani issues a strankovani
    a. Jestli filtrovat jednim stringem pres vsechny atributy nebo jinak
    b. Strankovani nebo kontinualni nacitani
3. Zjistit, jestli by stačilo vytvářet vlastní datové typy jen v Redmine a v EVISu je pak jen přidávat
    a. Zatim mam problem s vytvarenim, muzu ukazat
4. Pobavit se trochu o rolích a právech zaměstnanců v systému
    a. Kdo muze mazat treba sluzbu,...
5. pořešit rozdělení uživatelů na zaměstnance a klienty (kolik je čeho kategorií, čím se liší,...)
    a. U klientu jsme resili nejake ty anonymni a tak
6. Povinnost atributu
    a. Maji byt nektere atributy povinne?
7. Co jsou kategorie v pohledu struktury sluzby aktivity

Hovor

1. ds
2. filtrování stačí přes jeden filtr, plus přidat filtr na datum výkonu
    a. počítat s řazením
    b. kontinuální načítání lepší
3. snažit se vyřešit i vytváření
4. role
    a. zaměstnanci nevidí do všeho,
5. anonymní a známí klienti budou jiná role, řešíme u nich totiž jiné povinné atributy, u anonyma toho chceme méně
6. u atributů výkonů i klientů řešit (ne)povinnost

klient

- jméno, příjmeni, datum narození, bydliště, uzavření smlouvy, počet dětí, pohlaví
- různé role podle služeb
- zájemce o službu vs klient, pro filtrování
- má vazbu na klíčového pracovníka
- od kdy je uzavřena smlouva

obrazovka klient

- vlevo karta klienta s informacemi
- vpravo tabulka všech výkonů daného klienta v určité službě (protože klient se váže k jedné službě)
- vytvořit klienta
- ještě zde mám seznam individuálních plánu

obrazovka zaměstnanec

- jméno, přijmení,..
- lékařská prohlídka, BOZP, ….
    - ideálně připomínky, šlo by přes mail nebo přímo v EVISu

obrazovka nástěnka

- zprávy o systému, kam může každý psát postřehy, problémy,....

obrazovka individuální plán

- plán zastřešuje skupinu výkonů se stejným cílem
- lze ho vytvořit při vykazování nebo lze výkon připojit k existujícímu
- na obrazovce s plány lze vidět plán a jeho údaje a poté výkon k plánu přiřazené
- na obrazovku plán se dostanu z karty klienta

03.03.2022

Příprava

1. ukázat filtrování a řazení
2. přemýšlet o tom, kolik tam bude sloupečků a jakým stylem to udělat
    a. chci filtrovat datum, text, pracovníka, klienta a číslo
    b. jestli nějaké sloupečky třeba schovat
        i. určitě chci schovat třeba nějaký popis
3. povedlo se mi vymyslet, jak vytvářet uživatelské typy
4. možná si vypsat, jaké datové typy potřebujeme
    a. text, datum, číslo, výčtový typ

Hovor

Filtrování klientů předělat na vyhledávání

filtrování datumu přes období

přidat typ souborů

klient má na kartě klíčového pracovníka

atributy všech výkonů: datum, předmět, popis, zaměstnanec, klient, doba trvání výkonu, místo výkonu (V terénu/V kanceláři), v přítomnosti klienta (ano/ne)

- pro většinu asi ještě navázání na individuální plán
- v poradně je třeba typ poradenství

možná by se hodil výtisk jednoho nebo více výkonů

- většinou dva tři výkony, není kritické, kdyžtak řešit přes ctrl+p

popis výkonu dát jako rozevírací část řádku a všechny najednou otevřít/zavřít nebo rozevírat po kliku po jednom

14.03.2022

Příprava

1. Ukázat pokrok
    a. stránkování, filtrování, řazení
    b. vytváření a upravování výkonů (povinné atributy s hvězdičkou,...)
    c. mazání výkonů - s možností vrátit
    d. vytváření datových typů - nových nebo přidání existujících
    e. mazání služeb je řešeno archivováním
    f. odhlášení
2. ukázat wireframy
3. pobavit se o uživatelích(klientech a zaměstnancích)
    a.

Hovor

ano/ne možná předělat na výčet
časový formát - výběr podle počtu hodin a minut

klienty a zaměstnance místo mazání archivovat

upozornění 14 dnů dopředu

zaměstanci mají kurzy
- kurz má jméno a dobu trvání
- ideálně aby si to vyplňovali

obrazovky

- zaměstnanci
- klienti
- klientský plán (výkony, info o plánu)
- Můj účet (správa vlastního účtu a změna hesla)

<u>Postřehy od Davida</u>

- na dobu trvání použít integer
- kontrola čísla při vyplňování formuláře dovoluje i třeba: 4+4, ++,...
- klient možná nemusí být vždy povinný - praní prádla
- David dává naději, že prý e-mail půjde

- breakthrough
    - výkon je issue, plán je taky issue, pozor klient je taky issue
    - výkon má jako rodiče plán, plán má jako rodiče klienta
    - výkon si pamatuje klienta skrze sloupeček s id (prostě number hodnota)

31.03.2022

ukázat nové věci

- sekci s uživateli
    - vytváření, editace, filtrování, přiřazování služeb a práv
    - upozornění na datumy
- sekce můj účet
    - podobné jako u zaměstnanců
    - zatím lze upravovat jen u admina, musím opravit
- datový typ doba trvání
    - ukázat aktuální implementaci
    - probrat, jestli je potřeba a nestačí jen integrovaná hodnota
- klienti
    - jen nástřel
    - ukázat možná wireframy, ale asi už se ukazovaly minule
- vysvětlit strukturu klient - klientský plán - výkon
    - pořešit jméno u klienta, teď to mám jen v rámci jednoho políčka
- možná se zeptat na statistiky, jestli bude dostatečné
- ukázat restart hesla

vyhodit hlášku po úpravě účtu
doba trvání stačí ta od redminu a dát ji vždy k výkonu

ke klientovi patří klíčový pracovník

do příště mít ideálně finální verzi i s daty pro prezentaci

07.04.2022

- ukázat výsledné řešení klientů a klientských plánů
    - vytvoření klienta vytvoří výchozí klientský plán
    - včetně vytváření výkonů s vazbou na plán a klienta
    - vytváření služeb dělá i klientskou sekci
    -

klientský plán je individuální plán

návrhy na zlepšení

- při vykazování výkonu mít možnost vytvořit nový klientský plán
- v inviduálním plánu mít možnost minimálně zobrazit atributy výkonu, ideálně i upravit a vytvářet další
- úroveň splnění individuálního plánu

domluvili jsme se na

- dodělat systém ideálně s navrhovanými funkcemi
- připravit testování

Příloha E

Scénář k uživatelskému testování

Scénář pro vedení

1. Otevřete si stránku https://evis.azurewebsites.net/
2. Přihlašte se s uživatelským jménem “admin” a heslem “password”.
3. Vytvořte službu Noclehárna sv. Antonína s libovolným popisem.
4. Ve službě Poradna pro lidi v nouzi vytvořte aktivitu Finanční poradenství.
5. Nové aktivitě přidejte vlastnost Místo setkání s možnostmi “V kanceláři” a “V terénu”.

1. Přesuňte se na kartu se zaměstnanci.
2. Zaměstnancům přidejte vlastnost Platnost lékařské prohlídky typu datum.
3. Přidejte zaměstnance Ondřeje Nováka s e-mailovou adresou ondrej.novak@neexistuje.org a uživatelským jménem ondrej.novak. Platnost školení BOZP nastavte na zítra a zdravotní prohlídku na 30.5.2023.
4. Vytvořenému zaměstnanci přidejte roli Zaměstnanec ve službách Poradna pro lidi v nouzi a Výdejna Samaritán.
5. Zjistěte, u jakých zaměstnanců je upozornění a dále jaká vlastnost daného zaměstnance upozornění způsobuje.
6. Odhlaste se

Scénář pro zaměstnance

1. Přihlašte se s jménem “ondrej.novak” a heslem “tajneheslo”.
2. V rámci služby Poradna pro lidi v nouzi vytvořte klienta Patrik Chudý s libovolnými dalšími parametry.
3. Vytvořenému klientovi vytvořte individuální plán Řešení dluhové situace

1. V aktivitě Finanční poradenství ve službě Poradna pro lidi v nouzi vykažte dva výkony (oba přidejte pod klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace):
    a. první výkon má předmět Úvodní schůzka a trval 1 hodinu a 15 minut
    b. na druhém výkonu se probíraly různé možnosti řešení dluhové situace a tato schůzka trvala 40 minut
2. Přes klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace zjistěte celkový čas, jaký zaměstnanci charity strávili prací na tomto individuálním plánu.

1. V rámci služby Výdejna Samaritán a aktivity Výdej potravin zjistěte popis výkonu s předmětem Výdej chleba.
2. U stejné aktivity zkuste smazat libovolný výkon.
3. Šlo by výkon nějakým způsobem po smazání ještě vrátit? Zkuste to na smazání a vrácení dalšího výkonu.

1. V rámci správy svého uživatelského účtu upravte datum platnosti školení BOZP na 1.11.2023.

Příloha F

Poznámky k testování

Poznámky jsem si psal přímo k testovacím scénářům a jsou označeny tučným písmem. Nejsou formální a byly psané přímo při testování.

Poznámky Bc. Petr Větrovský

Scénář pro vedení

1. Otevřete si stránku https://evis.azurewebsites.net/
2. Přihlašte se s uživatelským jménem “admin” a heslem “password”.
3. Vytvořte službu Noclehárna sv. Antonína s libovolným popisem.
4. Ve službě Poradna pro lidi v nouzi vytvořte aktivitu Finanční poradenství.
5. Nové aktivitě přidejte vlastnost Místo setkání s možnostmi “V kanceláři” a “V terénu”.
    a. **“what, to co je za obrázek” (myšleno jako pochvala) - když není u aktivity žádný výkon, tak tam je pěkná grafika, aby stránka nebyla prázdná**
    b. **vytvořil ok, trochu se rozkoukával**

1. Přesuňte se na kartu se zaměstnanci.
2. Zaměstnancům přidejte vlastnost Platnost lékařské prohlídky.
    a. **u vlastnosti možná nemít vlastnost “jméno”, ale spíš použít “název”**
3. Přidejte zaměstnance Ondřeje Nováka s e-mailovou adresou ondrej.novak@neexistuje.org a uživatelským jménem ondrej.novak. Platnost školení BOZP nastavte na zítra a zdravotní prohlídku na 30.5.2023.
    a. **ok**
4. Vytvořenému zaměstnanci přidejte roli Zaměstnanec ve službách Poradna pro lidi v nouzi a Výdejna Samaritán.
    a. **chvíli hledal role a nechápal vazbu ke službě - to si vysvětluji spíš neznalostí domény, protože zaměstnanci vědí, že role se váže ke každé službě zvlášť**
5. Zjistěte, u jakých zaměstnanců je upozornění a dále jaká vlastnost daného zaměstnance upozornění způsobuje.
    a. **našel to**
6. Odhlaste se

Scénář pro zaměstnance

1. Přihlašte se s jménem “ondrej.novak” a heslem “tajneheslo”.
2. V rámci služby Poradna pro lidi v nouzi vytvořte klienta Patrik Chudý s libovolnými dalšími parametry.
3. Vytvořenému klientovi vytvořte individuální plán Řešení dluhové situace
    a. **Hledal to všude možně, ale neotevřel si konkrétního klienta na stránce s klienty. Poté už mu to dávalo smysl, ale intuitivní to pro něj nebylo.**

1. V aktivitě Finanční poradenství ve službě Poradna pro lidi v nouzi vykažte dva výkony (oba přidejte pod klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace):
    a. první výkon má předmět Úvodní schůzka a trval 1 hodinu a 15 minut
    b. na druhém výkonu se probíraly různé možnosti řešení dluhové situace a tato schůzka trvala 40 minut
        i. **zkoušel zadat čas v různých podobách a u některých případů zadávací pole přijalo i věci, které by nemělo, nic ale neznamenalo**

fatální chybu a v nejhoším případě si systém vstup typu 2+12 přeložil jako 0

2. Přes klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace zjistěte celkový čas, jaký zaměstnanci charity strávili prací na tomto individuálním plánu.

1. V rámci služby Výdejna Samaritán a aktivity Výdej potravin zjistěte popis výkonu s předmětem Výdej chleba.
    a. **našel výkon a skrze jeho rozbalení i hledaný popis**
2. U stejné aktivity zkuste smazat libovolný výkon.
3. Šlo by výkon nějakým způsobem po smazání ještě vrátit? Zkuste to na smazání a vrácení dalšího výkonu.
    a. **nepozdává se mu, že notifikace na vrácení smazaného výkonu je vlevo dole a ostatní notifikace vpravo nahoře**

1. V rámci správy svého uživatelského účtu upravte datum platnosti školení BOZP na 1.11.2023.
    a. **upravil bez problémů**

Poznámky Mgr. Karolína Píchová, DiS.

Scénář pro vedení

1. Otevřete si stránku https://evis.azurewebsites.net/
2. Přihlašte se s uživatelským jménem “admin” a heslem “password”.
    a. **bez problému**
3. Vytvořte službu Noclehárna sv. Antonína s libovolným popisem.
    a. **pomalé reakce systému - kladu za vinu spíše hostingu, později vše bylo lepší)**
4. Ve službě Poradna pro lidi v nouzi vytvořte aktivitu Finanční poradenství.
5. Nové aktivitě přidejte vlastnost Místo setkání s možnostmi “V kanceláři” a “V terénu”.
    a. **nejde upravit vlastnost, chtělo by to přidat vedle koše ještě ikonku tužky na úpravy vlastnosti**

1. Přesuňte se na kartu se zaměstnanci.
2. Zaměstnancům přidejte vlastnost Platnost lékařské prohlídky.
    a. znovu nejde upravovat
3. Přidejte zaměstnance Ondřeje Nováka s e-mailovou adresou ondrej.novak@neexistuje.org a uživatelským jménem ondrej.novak. Platnost školení BOZP nastavte na zítra a zdravotní prohlídku na 30.5.2023.
    a. **kvitována práce s výběrem datumů**
4. Vytvořenému zaměstnanci přidejte roli Zaměstnanec ve službách Poradna pro lidi v nouzi a Výdejna Samaritán.
    a. **bez problému**
5. Zjistěte, u jakých zaměstnanců je upozornění a dále jaká vlastnost daného zaměstnance upozornění způsobuje.
    a. **orientuje se, podílela se na návrhu**
6. Odhlaste se

Scénář pro zaměstnance

1. Přihlašte se s jménem “ondrej.novak” a heslem “tajneheslo”.
2. V rámci služby Poradna pro lidi v nouzi vytvořte klienta Patrik Chudý s libovolnými dalšími parametry.
3. Vytvořenému klientovi vytvořte individuální plán Řešení dluhové situace
    a. **za všechny 3 body v pořádku**

1. V aktivitě Finanční poradenství ve službě Poradna pro lidi v nouzi vykažte dva výkony (oba přidejte pod klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace):
    a. první výkon má předmět Úvodní schůzka a trval 1 hodinu a 15 minut
    b. na druhém výkonu se probíraly různé možnosti řešení dluhové situace a tato schůzka trvala 40 minut
        i. **v dialogu pro přidání výkonu našla v pořádku klienta i plán**
2. Přes klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace zjistěte celkový čas, jaký zaměstnanci charity strávili prací na tomto individuálním plánu.
    a. **v pořádku našla přes kartu klienta a individuální plán**

1. V rámci služby Výdejna Samaritán a aktivity Výdej potravin zjistěte popis výkonu s předmětem Výdej chleba.
2. U stejné aktivity zkuste smazat libovolný výkon.
3. Šlo by výkon nějakým způsobem po smazání ještě vrátit? Zkuste to na smazání a vrácení dalšího výkonu.
    a. **mazání a vrací dobré**

1. V rámci správy svého uživatelského účtu upravte datum platnosti školení BOZP na 1.11.2023.
    a. **neudělá notifikaci**

Moje poznámky navíc

- přidat možná proklik z Evisu na Redmine pro statistiky
- u statistik ještě rozmyslet, jak by je šlo přenést do Evisu, prostředí Redminu působí moc složitě

Poznámky Mgr. Petr Švepeš

- má menší rozlišení obrazovky, zajímavá věc

Scénář pro vedení

1. Otevřete si stránku https://evis.azurewebsites.net/
2. Přihlašte se s uživatelským jménem “admin” a heslem “password”.
3. Vytvořte službu Noclehárna sv. Antonína s libovolným popisem.
4. Ve službě Poradna pro lidi v nouzi vytvořte aktivitu Finanční poradenství.
    a. **chtěl rovnou dávat vlastnost při vytváření služby**
5. Nové aktivitě přidejte vlastnost Místo setkání s možnostmi “V kanceláři” a “V terénu”.
    a. **chvilku trvalo to najít**
    b. **drobné problémy, ale celkově docela dobře našel a zorientoval se**

    c. **práce s výběrem výčtového typu a jejich vyplnění nebyla tak intuitivní, když jsem mu ale vysvětlil, jak by to udělal správně, tak hned pochopil**

1. Přesuňte se na kartu se zaměstnanci.
2. Zaměstnancům přidejte vlastnost Platnost lékařské prohlídky typu datum.
    a. **našel hned a správně vyplnil**
3. Přidejte zaměstnance Ondřeje Nováka s e-mailovou adresou ondrej.novak@neexistuje.org a uživatelským jménem ondrej.novak. Platnost školení BOZP nastavte na zítra a zdravotní prohlídku na 30.5.2023.
    a. **nefunguje na menším rozlišení, zmizí tlačítko na přidání zaměstnance**
    b. **u datumů se orientuje v pořádku**
4. Vytvořenému zaměstnanci přidejte roli Zaměstnanec ve službách Poradna pro lidi v nouzi a Výdejna Samaritán.
    a. **chtěl původně přidat obě služby najednou (jak je to přímo napsáno v zadání), pak přidal postupně**
5. Zjistěte, u jakých zaměstnanců je upozornění a dále jaká vlastnost daného zaměstnance upozornění způsobuje.
    a. **všiml si a našel, i když předtím o této funkci vůbec nevěděl**
6. Odhlaste se

Scénář pro zaměstnance

1. Přihlašte se s jménem “ondrej.novak” a heslem “tajneheslo”.
    a. **všiml si správně, že tam jsou jen ty dvě služby, které předtím přidal a chápe tedy provázanost mezi úkoly**
2. V rámci služby Poradna pro lidi v nouzi vytvořte klienta Patrik Chudý s libovolnými dalšími parametry.
    a. **vytvořil v pořádku**
3. Vytvořenému klientovi vytvořte individuální plán Řešení dluhové situace
    a. **v pohodě našel docela rychle**

1. V aktivitě Finanční poradenství ve službě Poradna pro lidi v nouzi vykažte dva výkony (oba přidejte pod klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace):
    a. první výkon má předmět Úvodní schůzka a trval 1 hodinu a 15 minut
    b. na druhém výkonu se probíraly různé možnosti řešení dluhové situace a tato schůzka trvala 40 minut
        i. **šel na to intuitivně přes klienty a individuální plán, zatím lze ale vytvářet výkony jen na stránce s aktivitou**
        ii. **ukazuje se tak, že funkce vytváření výkonů u individuálního plánu by byla užitečná**
2. Přes klienta Patrika Chudého a jeho individuální plán Řešení dluhové situace zjistěte celkový čas, jaký zaměstnanci charity strávili prací na tomto individuálním plánu.
    a. **našel v poho, trochu hledal**

1. V rámci služby Výdejna Samaritán a aktivity Výdej potravin zjistěte popis výkonu s předmětem Výdej chleba.
    a. **první našel přes úpravu, apk jsem se doptal, jak by popis našel jiným způsobem a našel i možnost rozevření řádku s výkonem**
2. U stejné aktivity zkuste smazat libovolný výkon.
    a. **špatně četl, ale poté našel v pohodě**
3. Šlo by výkon nějakým způsobem po smazání ještě vrátit? Zkuste to na smazání a vrácení dalšího výkonu.
    a. **všiml si notifikace na první dobrou a k tomuto úkolu se dostal až v době, kdy ho měl vlastně hotový, vrácení smazaného výkonu (/smazaných výkonů) mu tak přišlo intuitivní**

1. V rámci správy svého uživatelského účtu upravte datum platnosti školení BOZP na 1.11.2023.
    a. **našel kartu se svým účtem a v pořádku upravil**

Příloha **G**

Seznam použitých zkratek

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

Příloha **H**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace s instrukcemi pro spuštění
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF