ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Plasmoid pro Remember the milk
Student:	Karel Dvořák
Vedoucí:	Ing. Ondřej Guth, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Navrhněte a implementujte klienta pro službu Remember The Milk. Klienta realizujte jako miniaplikaci pro KDE Plasma 5 s funkčností omezenou na trvalé přihlášení, zobrazení úkolů, jejich jednoduché filtrování a zjednodušené přidání úkolu nového. Proveďte rešeršní rozbor existujících klientů této služby. Pokračujte analýzou, návrhem a implementací, která bude šiřitelná jako svobodný software (opensource). Hotové řešení vhodnými prostředky otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 12. listopadu 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Plasmoid pro Remember the milk

Karel Dvořák

Vedoucí práce: Ing. Ondřej Guth, Ph.D.

15. května 2017

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Ondřeji Guthovi, Ph.D. za ochotu, cenné rady a nasměrování v průběhu tvorby této práce. Dále bych chtěl poděkovat své rodině a přátelům za jejich podporu a pomoc.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. § 46 odst. 6 tohoto zákona 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 15. května 2017 .....

Odkaz na tuto práci

Dvořák, Karel. *Plasmoid pro Remember the milk*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce pojednává o návrhu a implementaci miniaplikace na plochu určené k organizaci úkolů využívající služby Remember the milk. Na základě požadavků a výsledků analýzy je navržena nová miniaplikace eliminující problémy předcházejících řešení. V práci je uvedena analýza, návrh, postup řešení a testování klienta. Výsledkem implementační části práce je miniaplikace na plochu linuxového prostředí KDE Plasma 5.

Klíčová slova miniaplikace, úkoly, Remember the milk, KDE, Plasma 5

Abstract

This thesis is dealing with analysis and implementation of a desktop widget designated for organizing task via the Remember the milk service. A new application eliminating problems of previous solutions is designed, based on the requirements and analysis. The thesis contains description of analysing, designing, implementing and testing the client. The output of the implementation part is the desktop widget for the Linux environment KDE Plasma 5.

Keywords widget, tasks, Remember the milk, KDE, Plasma 5

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Každý z nás se již někdy dostal do fáze, kdy se musel rozhodnout, jakým způsobem bude organizovat svůj pracovní či osobní život. Různé úkoly, připomínky, termíny, nápady, myšlenky a další podobná *data*, která musíme někde uchovávat a organizovat je. Nabízí se otázka *„Kam s nimi?“*, obdobně jako již pan Neruda ve svém díle kladl otázku *„Kam s ním?“*. Problematika udržování pořádku v povinnostech se ovšem přirozeně nachází na pomezí několika technologických oblastí. Řeší ji totiž lidstvo řadu staletí.

Mnohá řešení tohoto problému staví na neotřesitelných základech využití toho, co známe již několik staletí v různých formách. Myslím tím obyčejnou tužku a papír. Ať už se jedná o všestranný poznámkový blok, který neustále nosíme s sebou, nebo o jednoduché žluté přilepovací lístečky na nástěnku, jde o tradiční formu organizace úkolů. Tyto a další podobné varianty klasického přístupu typu: „*napíšu si to, abych na to nezapomněl*“ mají řadu výhod. Můžeme si zapsat skutečně cokoliv, kdykoliv a kdekoliv, bez žádného dalšího nutného vybavení a jiných závislostí (kromě téměř vždy všudypřítomné tužky a papíru).

Výše zmíněné ovšem nese také mnoho nedostatků, jelikož po několika dnech praktikování systému „*tohle napíšu sem, tamto tam*“ zjistíme, že veškeré naše úsilí plnění povinností skončí hledáním a probíráním stohů různých útržků a listů, na kterých se možná nachází to, co chceme splnit. Nevýhodu částečně eliminuje již zmíněný univerzální poznámkový blok či deník, do kterého se budeme snažit psát všechny naše úkoly a poznámky. Nicméně musíme vzít v potaz, že tento blok musíme neustále nosit při sobě. A to především pro eventualitu náhlé příležitosti si poznamenat důležitou věc. Pokud ho někde zapomeneme, přijdeme navíc o možnost plnění již vedených úkolů. A pokud ho náhodou nadobro ztratíme, již se nikdy nedozvíme, co jsme to vlastně měli na práci (nejspíše neztratit zápisník). Toto je jedna z největších nevýhod všech *analogových* řešení, tedy minimální až spíše nulová možnost zálohování.

S možností zálohování úzce souvisí již zmíněná centralizace poznámek na jednom místě. Ideálně bychom měli za cíl mít všechny úkoly dostupné na

jednom organizovaném místě (obdobně jako u papírového deníku). Ovšem s přidanou dostupností odkudkoli, kdykoli a v konzistentní, neustále aktualizované, podobě. Člověk se totiž přirozeně vyskytuje na různých místech, v různých situacích. Může v práci plnit úkoly zadané svým nadřízeným, dále je nutné provádět mnoho domácích prací, mimo domov plní různé závazky, chodí na schůzky a tak dále. Při každé zmíněné aktivitě používáme jiné prostředky, se kterými plníme zadané úkoly. Přirozeně to samé platí i pro jednotlivá prostředí. Dostáváme tedy určitou diverzifikaci formy, obsahu a způsobu organizace. Úkoly lze zaznamenávat takzvaně přímo v akci za chůze, nebo jen na specifickém místě. Nabízí se tak široká variabilita koncových řešení a jim příslušných zařízení.

Dále kdo nikdy na nic nezapomněl, nechť se přihlásí. Jakožto lidé máme schopnost ukládat a zpracovávat v paměti mozku neuvěřitelné množství různých informací a okolních vjemů. Nicméně všeobecně známý princip *škatulkování* a tvorby asociací mnohdy dokáže být nepředvídatelný. Ač se snažíte sebevíc, na některé myšlenky si zkrátka nemůžete vzpomenout. A to do té doby, dokud mysl něco trochu nenasměruje. Bylo by tedy velmi žádoucí, abychom měli možnost být upozorněni na nějaký termín či schůzku. Opět kdykoliv, kdekoliv.

Samozřejmě nejsme vždy schopni udělat naprosto každou věc, kterou jsme měli v plánu splnit. Hodilo by se tak některé úkoly upřednostňovat více před těmi ostatními, méně důležitými. Mnohdy stačí jen vizuální rozlišení, například pomocí barvy či čísla priority. Tím je okamžitě jasné, že určité věci mohu nechat na později, a naopak ty další musím dokončit co nejdříve.

Když už je té práce opravdu hodně, hodila by se také nějaká možnost rychlého hledání a filtrování podle různých parametrů. Výše jsem zmínil již mnoho kandidátů na ono filtrování, například datum, priorita, různé seznamy. Zúžením okruhu momentálně řešitelných úkolů na ty aktuálně podstatné dosáhneme většího povědomí o povinnostech, s čímž též naroste efektivita práce.

Všechno výše zmíněné přináší mnoho různých otazníků, jak toho docílit. Tyto požadavky reálně nelze splnit s pouhou pomocí tužky a papíru. Postupně tedy přecházíme ke stále větší a větší potřebě komplexnějšího řešení, které dokáže pokrýt požadavky a zároveň je schopné eliminovat dílčí nedostatky.

Abychom pokryli potřeby i toho nejnáročnějšího uživatele elektronického organizačního systému, musíme do něj zahrnout většinu požadavků vyjmenovaných dříve. Různé online dostupné webové nástroje naštěstí nabízejí ve velké míře řešení problému organizace úkolů. Mnohé z nich také nabízí klienty svých služeb na různá mobilní zařízení či stolní počítače. Ne všechny dostupné platformy jsou ovšem pokryty.

Cíl práce

Mým úkolem je vytvořit klienta služby *Remember the milk* na platformu *Linux* ve formě miniaplikace na plochu (*plasmoid*) desktopového prostředí *KDE Plasma 5*. Samotná aplikace bude umožňovat obsluhu základní funkčnosti zmíněné služby.

Jedná se především o operace prováděné při typickém používání nástroje na organizaci úkolů. Tedy zobrazování úkolů, jejich přidávání, mazání, označování jako splněné, jednoduché filtrování...

Na základě rešerše stávajících řešení, analýzy funkcionality webového klienta služby a požadavků bude vytvořen návrh řešení. Aplikace bude dle návrhu implementována. Součástí práce také bude testování aplikace.

Kapitola 1

Analýza

1.1 Současná řešení

1.1.1 Remember the milk v0.1

Klienta ve formě *plasmoidu* pro online úkolovací službu *Remember the milk* již dříve vytvořil v roce 2009 programátor Andrew Stromme [1]. Projekt byl funkční v prostředí *KDE 4*, ve kterém byl též obsažen. Jeho podobu lze vidět na obrázku 1.1.

Celkový vzhled vychází z principu použití na ploše ve formě *plasmoidu*. V kompaktním okně aplikace se seznamy úkolů (např. „Práce“ či „Osobní“) utvářejí do jednotlivých panelů. Ty fungují tak, že je zobrazen vždy právě jeden seznam úkolů. Pro změnu zobrazeného seznamu stačí kliknutím myši změnit aktuální panel na liště seznamů.

Samotné úkoly se zobrazují ve vertikálním seznamu, ve kterém jsou seskupovány dle termínu splnění. Stěžejní informací je název úkolu, který je doplněn barevným pruhem priority úkolu a případným datem termínu splnění.

Pod lištou seznamů je dostupné textové pole určené k vyhledávání. To umožňuje snadné filtrování mezi zobrazenými úkoly.

Přidávání úkolů funguje skrze textové pole umístěné na spodní hraně *plasmoidu*. Textový vstup podporuje funkci *Smart Add* [2], kterou nabízí služba *Remember the milk*. Tato funkce analyzuje jednotlivá klíčová slova a je schopna například zjistit, že chceme naplánovat úkol na zítra, pokud do názvu napíšeme: „napsat dopis tomorrow“.

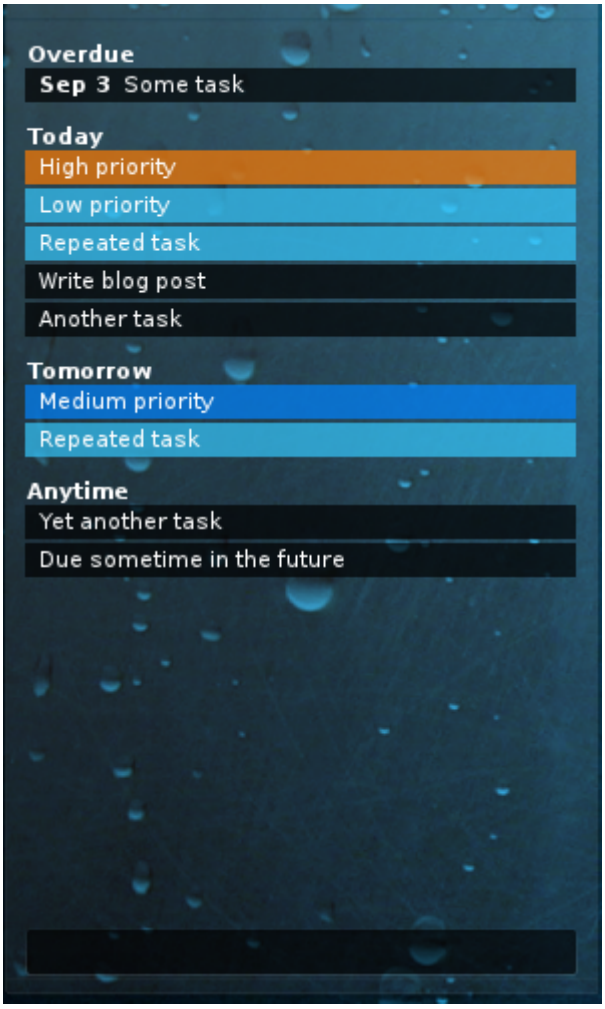
Nastavení aplikace se chová tak, jak je na platformě *KDE Plasma* standardní. Pomocí kliknutí pravého tlačítka myši kamkoliv na plochu *plasmoidu* se zobrazí kontextová nabídka, ze které lze otevřít dialogové okno s nastavením. Okno je systémové povahy a je rozšířeno o položky specifické pro konkrétní aplikaci. V tomto případě zde nalezneme především možnost pro přihlášení do *služby*. To probíhá skrze okno prohlížeče obsažené přímo v aplikaci.

Obrázek 1.1: Podoba plasmoidu pro *RTM* od Andrewa Stromma [3]

1.1.2 Remember the milk simple

Na předchozí projekt v roce 2011 dále navázal jiný programátor, Hans Chen. Ten dílo Andrewa upravil a pozměnil dle svých požadavků [4]. Jak můžeme vidět na obrázku 1.2, změny se týkaly především uživatelského rozhraní, které se mu zdálo nedostatečně čisté. Odebral několik ovládacích prvků jako například lištu seznamů či pole vyhledávání. Též rozšířil barevný pruh priority tak, aby podbarvoval celou položku úkolu. Celkově vzhled zjednodušil.

Také mu nepřišel příliš ergonomický postup označování úkolů jako splněné. Původně bylo nutné kliknout myší na daný úkol a až poté se zobrazilo pole pro zaškrtnutí. Následně se muselo vše potvrdit kliknutím na tlačítko aktualizování úkolu. Implementoval tak funkcionalitu, která mu umožňovala jednoduše kliknout prostředním tlačítkem myši na požadovaný úkol, který se poté označil



Obrázek 1.2: Upravený plasmoid pro *RTM* od Hanse Chena [5]

jako splněný. Změny tedy byly víceméně marginálního charakteru.

1.1.3 Souhrn

Vyzkoušel jsem verzi projektu, která je obsažena v balíčku *kdeplasma-addons* prostředí *KDE 4* pod jménem *Remember The Milk v0.1*, tedy aplikaci od Andrewa Stromma.

Co se funkcionalit týče, obsahuje výpis všech úkolů, které máme přiřazeny pod svým účtem. Je zde možnost výběru jednotlivých seznamů, či jednoduché filtrování dle zadaných klíčových slov. Také můžeme vytvořit zcela nový úkol.

Na systému s grafickým prostředím *KDE 4* je tato verze funkční. V novější verzi prostředí *KDE Plasma 5* je ovšem tato aplikace nefunkční. Při snaze o přihlášení totiž nastává problém se samotnou autentizací účtu služby se

serverem. Z tohoto důvodu ji dále nelze používat.

1.2 Funkcionalita služby Remember the milk

Služba *Remember the milk* je zaměřena na organizaci a plánování úkolů. Nabízí tedy funkcionalitu správce každodenních úkolů a připomínek. Můžeme zde evidovat naše osobní či pracovní povinnosti, které dále lze seskupovat do jednotlivých seznamů. Každý úkol je především definován svým názvem, prioritou, přidělením do seznamu a termínem splnění. Existuje také mnoho dalších atributů jako například místo, tag, poznámka a jiné [6].

1.2.1 Platformy

Provozovatel poskytuje službu na webu, je tedy dostupná skrze webový prohlížeč z kteréhokoliv zařízení. Existují také oficiálně podporované aplikace na různé desktopové a mobilní systémy. Z těchto systémů je většina pokryta, nicméně je k dispozici veřejné rozhraní, které umožňuje programátorům tvořit klienty pro jakékoli další platformy.

Na poli desktopových systémů můžeme využívat verzi aplikace pro následující operační systémy:

- Windows od firmy Microsoft,
- macOS od firmy Apple,
- systémy založené na jádře Linux.

Pokud chceme využívat služeb i v terénu, pomohou nám k tomuto účelu aplikace určené pro mobilní telefony a tablety. Existují totiž aplikace pro operační systémy:

- iOS od firmy Apple,
- Android od firmy Google,
- BlackBerry 10 of firmy BlackBerry.

Bohužel uživatelé, kteří vlastní nějaké zařízení běžící na mobilní verzi Windows Mobile (dříve pojmenované Windows Phone) mají smůlu, jelikož provozovatel na těchto zařízeních své služby neposkytuje. Uživatelé maximálně mohou používat webové rozhraní skrze internetový prohlížeč, což je na mobilních zařízeních přinejlepším nepohodlné a toto řešení tak ztrácí na reálné použitelnosti.

Dále se brzy očekává podpora pro systémy chytrých náramkových hodinek. V současné době probíhá testovací program pro tyto typy hodinek:

- Apple Watch of firmy Apple,
- Android Wear od různých výrobců.

Pokud uživatel používá desktopový klient Microsoft Outlook pro příjem elektronické pošty, může též využít doplňku, který synchronizuje úkoly ze služby *Remember the milk* přímo do Outlooku.

Podpora různých zařízení je tedy více než široká. A to i přesto, že některé platformy mezi nimi nenalezneme. Vývojáři též mají k dispozici veřejné *API* rozhraní, na kterém mohou postavit svou vlastní aplikaci využívající služeb *Remember the milk*. Právě díky tomu mohly vzniknout projekty třetích stran, které jsem již zmínil v sekci 1.1. Každý programátor si tak může sám vytvořit aplikaci fungující na své požadované platformě či operačním systému. Případně je taktéž možné vytvořit alternativního klienta oproti těm stávajícím.

1.2.2 Webový klient

Jak samotná služba na webu vypadá lze vidět na obrázku 1.3. Pro účely popisu je zobrazen seznam s názvem „Work“. Na horní hraně je zobrazena lišta obsahující vyhledávací pole a tlačítka placeného offline režimu, notifikací a nastavení. Vyhledávací pole standardně vyhledává slova obsažená v názvech úkolů, případně je možné dodefinovat pokročilejší kritéria jako neobsažená slova či stupně priority apod.

Dále je plocha rozdělena na tři sloupce. Jelikož je web responzivní, tak se podle šířky obrazovky tento počet eventuálně snižuje až na pouhý jeden.

Na levé straně je možné vybrat z mnoha seznamů úkolů. Ty je lze přidávat a mazat podle potřeby. Nalezneme zde klasické seznamy jako například osobní, práce a jiné, či tagy (označení) vhodné pro třídění. Uživatel tak může vytvořit dynamické chytré seznamy (smart lists), které pod sebe následně shlukují úkoly podle určitého kritéria, například všechny úkoly obsahující tag (označení) „Praha“. Takovéto seznamy se poté chovají jako seznamy standardní, jsou tedy obsaženy i ve výpisu skrze rozhraní *API*. Nicméně právě skrze *API* do nich nelze přímo přidávat úkoly.

Uprostřed se nachází sloupec obsahující samotné úkoly patřící do daného seznamu. Jsou uspořádány pod sebou do řádků. Každý z nich určuje název, prioritu a případný termín splnění. Mezi pruhem priority a názvem se nachází zaškrtávací pole, skrze které se po zaškrtnutí v pravém sloupci zobrazí detaily úkolu. Současně s označením více úkolů je lze hromadně upravovat.

Nad zobrazenými úkoly se nachází vstupní textového pole, díky kterému uživatel může jednoduše přidávat své nové úkoly. Je to intuitivní, poměrně jednoduché a efektivní. Při zadání názvu můžeme také přímo vložit určitá klíčová slova, která úkol dále blíže specifikují. Například „=2 minutes“ nastaví časovou náročnost na 2 minuty, „!1“ nastaví prioritu na hodnotu 1, přes „#Osobní“ nastavíme seznam, do kterého patří zadávaný úkol, v tomto případě na „Osobní“. Takto můžeme nastavit i mnoho dalších atributů úkolu [2].

Všechna tato nastavení jsou také dostupná skrze prvky uživatelského rozhraní – tlačítka pod vstupem.

Každému úkolu lze přiřadit jednotlivé termíny – do kterého dne má být splněn a kdy se na něm má začít pracovat. Existují zde priority, podle kterých je možné řídit upřednostnění některých úkolů před jinými. Konkrétně jsou priority tři číselné a pomyslná čtvrtá bez priority. Pomocí těchto parametrů společně s názvem lze úkoly řadit. Nejlepší možností je řazení dle termínu splnění. Nejprve jsou zobrazeny tyto úkoly, následují je poté úkoly bez data splnění. Zároveň jsou jako sekundární řadící kritéria priorita a jako poslední název úkolu. Tím je docíleno srozumitelného zobrazení všech úkolů.

Mezi další funkce patří možnost přiřazení lokace, uživatele, předpokládané časové náročnosti řešení úkolu nebo také opakování.

1.2.3 Desktopový klient pro Linux

Jak lze vidět na obrázku 1.4, desktopová verze klienta služby pro operační systém Linux nabízí funkce naprosto stejné jako jeho webová předloha. Jedná se totiž o webovou verzi, která je obsažena v rámci prostředí jakožto vložené okno. Veškerá funkcionalita je operována skrz webové rozhraní. Navíc je zde pouze panel *GTK*, který obsahuje možnosti práce s oknem samotným a odkazy do již zmíněné webové vrstvy [7].

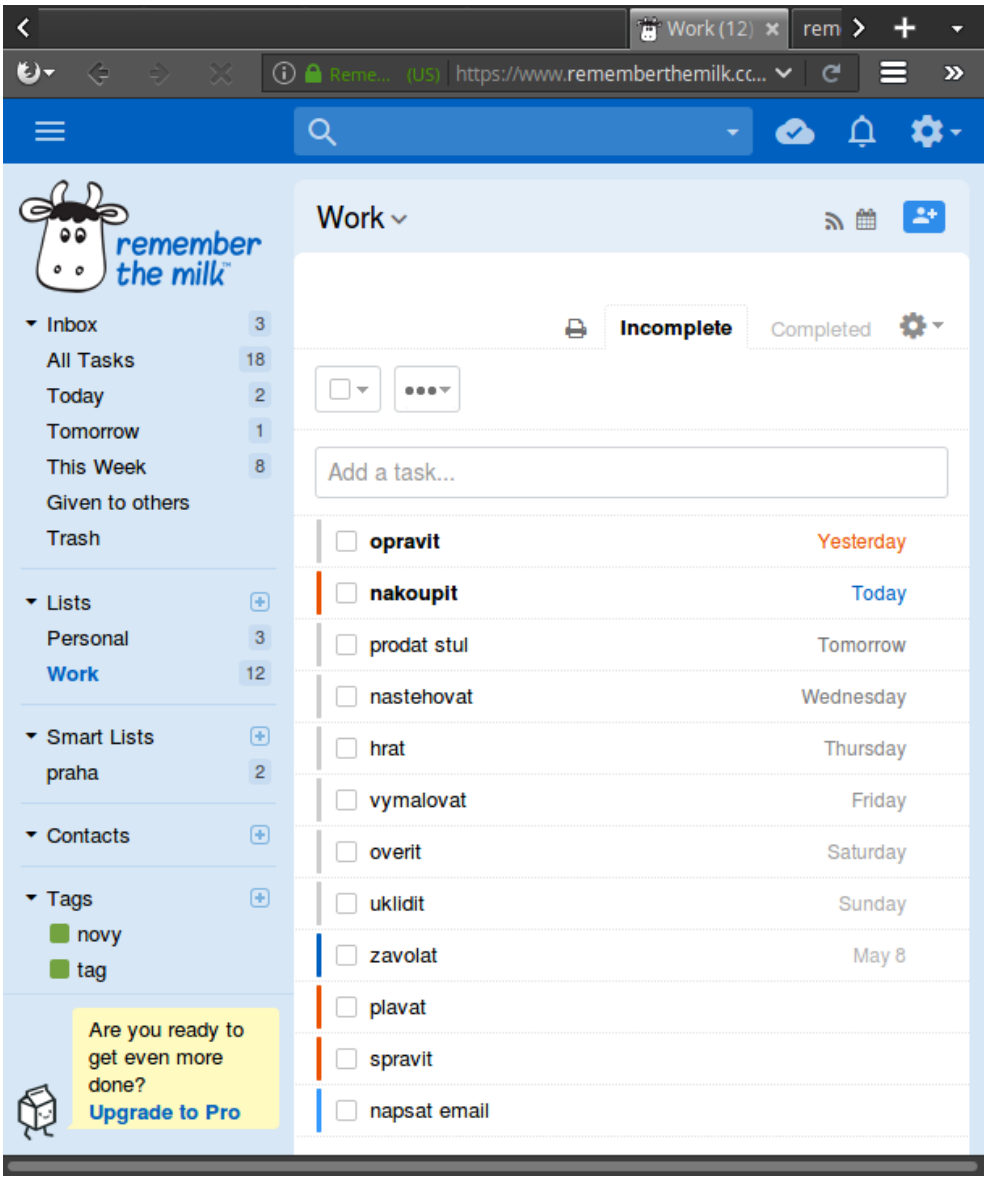
1.3 Definice požadavků

V úvodu této práce jsem již nastínil typické potřeby běžného uživatele na systém, který se zabývá problematikou organizace úkolů. V analýze služby samotné jsem se zaměřil právě na tyto potřeby a výstupem jsou mimo jiné i následující požadavky.

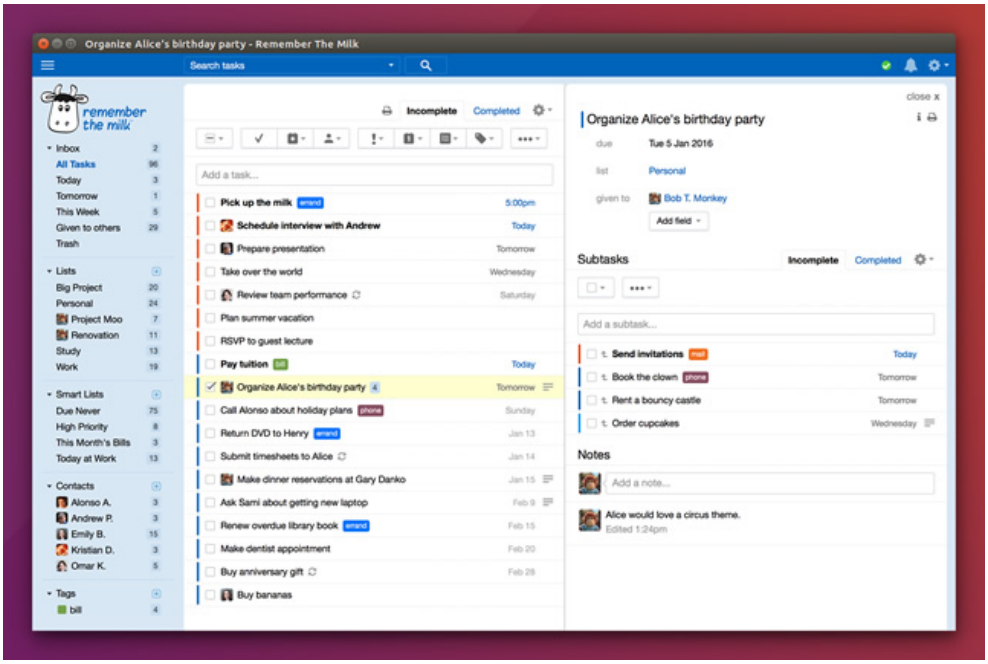
Pro upřesnění, aplikací se myslí klient služby *Remember the milk* pro prostředí *KDE Plasma 5* ve formě *plasmoidu*.

1.3.1 Funkční požadavky

- **FR1 Zobrazování úkolů**
  - Aplikace musí umět zobrazit výpis nesplněných úkolů jednotlivých seznamů.
  - Priorita: vysoká.
- **FR2 Přidávání nových úkolů**
  - Aplikace umožňuje vytvoření nového úkolu čtením vstupu z uživatelského rozhraní – pomocí textového pole.
  - Priorita: vysoká.



Obrázek 1.3: Náhled webové verze služby *Remember the milk*



Obrázek 1.4: Náhled Linuxového klienta *Remember the milk* [7]

- **FR3 Filtrování úkolů**
  - Aplikace dokáže filtrovat úkoly v aktuálně zobrazeném seznamu dle názvu úkolu na bázi fulltextového vyhledávání.
  - Priorita: vysoká.
- **FR4 Zobrazení termínu splnění**
  - Aplikace umí zobrazit termín splnění vybraného úkolu.
  - Priorita: střední.
- **FR5 Označení úkolu jako splněný**
  - Aplikace dokáže vybraný úkol označit jako splněný.
  - Priorita: vysoká.
- **FR6 Odstranění úkolu**
  - Aplikace podporuje odstranění vybraného úkolu.
  - Priorita: vysoká.
- **FR7 Zvýraznění priority úkolu**
  - Aplikace zobrazuje prioritu úkolu pomocí barevného pruhu vedle názvu.
  - Priorita: střední.

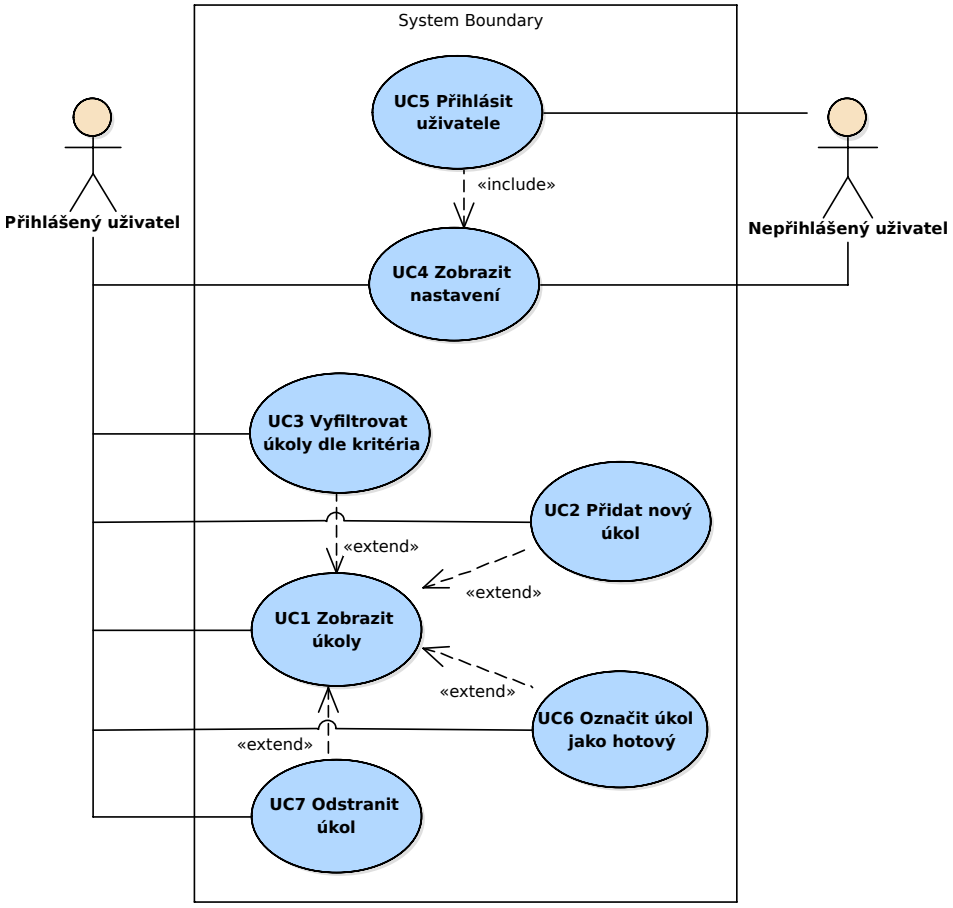
1.3.2 Nefunkční požadavky

- **NFR1 Spustitelnost v běhovém prostředí *KDE***
  - Aplikace lze spustit v linuxovém prostředí plochy *KDE Plasma 5*.
  - Priorita: vysoká.
- **NFR2 Napojení na službu *Remember the milk***
  - Aplikace využívá veřejné rozhraní online úkolovací služby *Remember the milk*.
  - Priorita: vysoká.
- **NFR3 Autorizace a autentizace účtu**
  - Aplikace umožňuje trvalé přihlášení k uživatelskému účtu služby uvedené v *NFR2*, včetně správné autentizace uživatele.
  - Priorita: vysoká.

1.4 Případy užití

Případy užití pokrývají dříve zmíněné požadavky. Níže nalezneme jejich podrobnější popis, případně na obrázku 1.5 můžeme vidět grafické zobrazení modelu případů užití a jeho vzájemné vazby.

- **UC1 Zobrazit úkoly**
  - Umožňuje uživateli zobrazit všechny aktuální seznamy *úkolů*.
- **UC2 Přidat nový úkol**
  - Umožňuje uživateli vytvořit nový *úkol* a přidat ho do aktuálně zobrazeného seznamu.
- **UC3 Vyfiltrovat úkoly dle kritéria**
  - Umožňuje uživateli zobrazení filtrovaných *úkolů* odpovídajících hledanému slovu v aktuálním seznamu.
- **UC4 Zobrazit nastavení**
  - Umožňuje uživateli zobrazit nastavení aplikace.
- **UC5 Přihlásit uživatele**
  - Umožňuje uživateli přihlášení ke službě *Remember the milk* pomocí přihlašovacích údajů.



Obrázek 1.5: Model případů užití

- **UC6 Označit úkol jako hotový**
  - Umožňuje uživateli změnit stav vybraného *úkolu* na splněný.
- **UC7 Odstranit úkol**
  - Umožňuje uživateli odstranění vybraného *úkolu* ze seznamu.

1.5 Scénáře případů užití

Všechny níže uvedené scénáře případů užití obsahují následující typy entit:

- **Uživatel**
  - Aktér, který obsluhuje *systém* v roli uživatele.

- **Systém**
  - Zastupuje klientskou aplikaci na ploše, která vykonává požadavky *uživatele*.
- **Server**
  - Zastupuje server služby *Remember the milk*, který komunikuje s *systémem*.

1.5.1 Scénář UC1 Zobrazit úkoly

Vstupní stav: *Uživatel* je přihlášen dle *UC5*.

1. *Systém* stáhne ze *serveru* jednotlivé seznamy.
2. *Systém* stáhne ze *serveru* odpovídající úkoly seznamů.
3. *Systém* zobrazí jednotlivé seznamy a úkoly ve svém grafickém rozhraní.
4. *Uživatel* může v případě potřeby kliknutím myší na lištu seznamů změnit aktuálně zobrazený seznam *úkolů*.

1.5.2 Scénář UC2 Přidat nový úkol

Vstupní stav: *Uživatel* je přihlášen dle *UC5* a má zobrazen seznam úkolů dle *UC1*.

1. *Uživatel* zadá do vstupního textového pole zobrazeného pod úkoly název nového úkolu s případnými dalšími atributy.
2. *Uživatel* potvrdí vložení úkolu kliknutím myši na tlačítko vložení nového úkolu nebo stisknutím klávesy „Enter“.
3. *Systém* vložený úkol zpracuje a odešle jej *serveru*.
4. *Server* potvrdí *systému* vložení odesláním úkolu zpět v odpovědi.
5. *Systém* zobrazí nově vložený úkol mezi odpovídajícím seznamem úkolů.

1.5.3 Scénář UC3 Vyfiltrovat úkoly dle kritéria

Vstupní stav: *Uživatel* je přihlášen dle *UC5* a má zobrazen seznam úkolů dle *UC1*.

1. *Uživatel* zadá do vstupního textového pole zobrazeného pod lištou seznamů klíčové slovo, podle kterého chce vyhledávat.

2. *Systém* okamžitě v průběhu zadávání vstupu prohledá aktuálně zobrazený seznam a vyfiltruje podmnožinu úkolů, které splňují kritéria aktuálního vyhledávaní.

3. *Systém* zobrazí vyfiltrovaný seznam úkolů na místě původního nefiltrovaného seznamu.

4. *Uživatel* kliknutím tlačítka myši na lištu seznamů může změnit aktuální seznam. *Systém* tak provede vyhledávání nad aktuálně zobrazeným seznamem.

5. *Uživatel* může zrušit vyhledávání kliknutím na tlačítko smazání textového vstupu uvnitř textového pole nebo postupným smazáním textu klávesou „Delete“ či „Backspace“. *Systém* tak zpět zobrazí nefiltrovaný seznam.

1.5.4 Scénář UC4 Zobrazit nastavení

1. *Uživatel* klikne pravým tlačítkem myši na plochu *systému*.

2. *Systém* zobrazí kontextovou nabídku aplikace.

3. *Uživatel* vybere položku nastavení kliknutím levého tlačítka myši.

4. *Systém* zobrazí okno nastavení.

5. *Uživatel* potvrdí nastavení kliknutím na tlačítka „Ok“ či „Apply“.

6. *Systém* uloží nastavení a uzavře dané okno.

1.5.5 Scénář UC5 Přihlášení uživatele

1. Dle *UC4* zobrazí uživatel nastavení.

2. *Uživatel* klikne myší na tlačítko přihlášení pojmenované „Login“.

3. *Systém* otevře webový odkaz na přihlášení do služby *Remember the milk* v externím webovém prohlížeči.

4. *Uživatel* vyplní své uživatelské jméno a heslo a přihlásí se na webu služby, tím udělí *systému* právo pro přihlášení k účtu.

5. *Uživatel* uzavře okno webového prohlížeče.

6. *Uživatel* klikne v *systému* na tlačítko potvrzení přihlášení.

7. *Systém* zažádá *server* o přidělení tokenu (bezpečnostního kódu).

8. *Server* odešle *systému* token.

9. *Systém* uloží token a zobrazí potvrzení přihlášení změnou ikony stavu přihlášení z červené na zelenou a přepsáním stavu z „Logged out“ na „Logged in“.

1.5.6 Scénář UC6 Označit úkol jako hotový

Vstupní stav: *Uživatel* je přihlášen dle *UC5* a má zobrazen seznam úkolů dle *UC1*.

1. *Uživatel* klikne myší na tlačítko splnění vedle požadovaného úkolu, který chce splnit.
2. *Systém* odešle serveru požadavek o splnění daného úkolu.
3. *Server* potvrdí operaci odpovědí se změněným stavem úkolu.
4. *Systém* provede označení úkolu jako splněný a daný úkol odebere ze seznamu aktuálních nesplněných úkolů.

1.5.7 Scénář UC7 Odstranit úkol

Vstupní stav: *Uživatel* je přihlášen dle *UC5* a má zobrazen seznam úkolů dle *UC1*.

1. *Uživatel* klikne myší na tlačítko odstranění vedle požadovaného úkolu, který chce odstranit.
2. *Systém* odešle serveru požadavek o smazání daného úkolu.
3. *Server* potvrdí operaci odpovědí se změněným stavem úkolu.
4. *Systém* provede odstranění úkolu a daný úkol odebere ze seznamu aktuálních úkolů.

1.5.8 Souhrn

V tabulce 1.1 lze vidět, které případy užití pokrývají jednotlivé požadavky. Vzhledem k tomu, že je každý požadavek pokryt, je přesně definováno jakým způsobem bude funkcionalita zajištěna. Na tomto základu je dále možné rozvíjet uživatelské rozhraní a logiku aplikace.

1.6 Doménový model

Při tvorbě doménového modelu 1.6 jsem vycházel z výše definovaných požadavků a případů užití. Potřebujeme zachytit následující klíčové entity:

- *Úkol*,

Tabulka 1.1: Pokrytí požadavků případy užití

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	NFR3
UC1	X			X			X	
UC2		X						
UC3			X					
UC4								X
UC5								X
UC6					X			
UC7						X		

- *Seznam.*

Ty si dále probereme níže.

1.6.1 Task (úkol)

Tato entita reprezentuje samotný úkol. Je nutné udržovat následující údaje o každém z nich. Jedná se o:

- název,
- identifikační číslo,
- prioritu,
- datum termínu splnění.

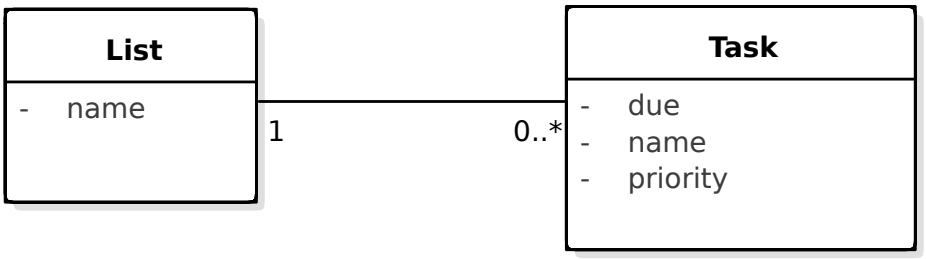
Tato entita definuje klíčové prvky, které budou tvořit obsah aplikace samotné. Vzhledem k účelu a formě aplikace je důležité, aby bylo dostupné relevantní a správné množství potřebných údajů. To je dáno náturou miniaplikace na ploše jakožto *plasmoidu*. Účelem tak není zobrazovat veškeré informace o úkolu, nýbrž jejich stěžejní podmnožinu.

1.6.2 List (seznam)

Entita představuje seznam úkolů. O seznamu je nutné znát:

- název,
- identifikační číslo.

Seznam sám o sobě je nutné především správně identifikovat. Jak programově tak uživatelsky. Pro tyto účely postačují jmenované atributy. Vzhledem k použití seznamu jakožto kontejneru jednotlivých úkolů je důležitý právě obsah úkolů samotných.



Obrázek 1.6: Doménový model

Kapitola 2

Návrh

2.1 Uživatelské rozhraní

2.1.1 Ideální uživatelské rozhraní

Existuje skutečně něco takového jako ideální uživatelské rozhraní? Každý uživatel má přeci jiné potřeby a nároky na používání aplikace. Někomu vyhovují nekonečné možnosti přizpůsobitelnosti a širokého nastavení každé možné části rozhraní. Naproti tomu existují ti, kteří vyžadují jednoduchost a kladou důraz na splnění funkcionality v co nejpřehlednějším možném uskupení. Oba dva přístupy mají své výhody a nevýhody.

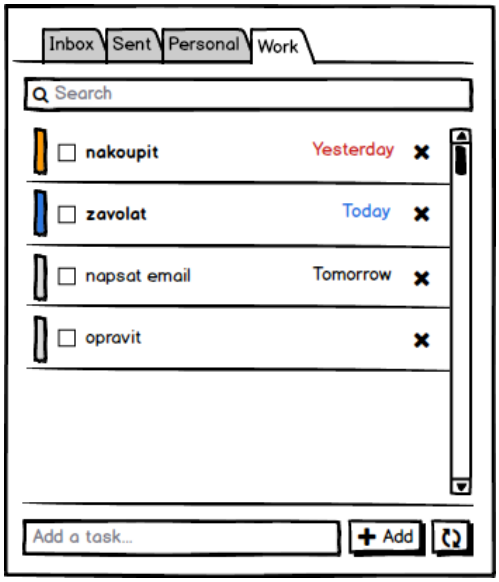
Dnes je možné snadno se setkat s postupným zjednodušováním na poli vývoje softwaru pro širokou veřejnost. Různí velcí hráči zjišťují, že drtivá většina specializovaných funkcí a vymožeností zůstává ve finálním měřítku nevyužita. A vzhledem k tomu, že tvorba těchto specialit stojí mnoho úsilí, práce a tedy peněz, nelze se divit tomu, že se široce uplatňují principy jednoduchosti a maximalizované praktičnosti.

Na druhou stranu tato generalizace často vede k omezení či nemožnosti využití pokročilejších funkcí. Ty mohou náročnějšímu uživateli dříve či později chybět, jelikož se omezila jeho možnost pokročilejšího využití aplikace.

Je tedy nutné dobře zvážit oba směry přístupu s cílem co nejlepšího zvolení správné množiny funkcí pro co největší skupinu uživatelů. Následně je klíčové tomu příslušně přizpůsobit uživatelské rozhraní. To by mělo snadným způsobem splňovat všechny požadavky z pohledu funkcionality a zároveň zachovat onu požadovanou intuitivnost a jednoduchost.

2.1.2 Uživatelské rozhraní plasmoidu

Dříve zmíněný *úkol* a *seznam* jsou centrálními body aplikace. Při zaměření na tradiční poznámky psané na papír je zřejmé, že přirozeně zapisujeme jednotlivé úkoly často s odrážkou na jednom řádku. Tuto myšlenku by měla aplikace



Obrázek 2.1: WF1: Zobrazené seznamy

reflektovat tím, že úkoly budou vypsány pod sebou. Na obrázku 2.1 blíže můžeme vidět, jak toto řešení bude vypadat.

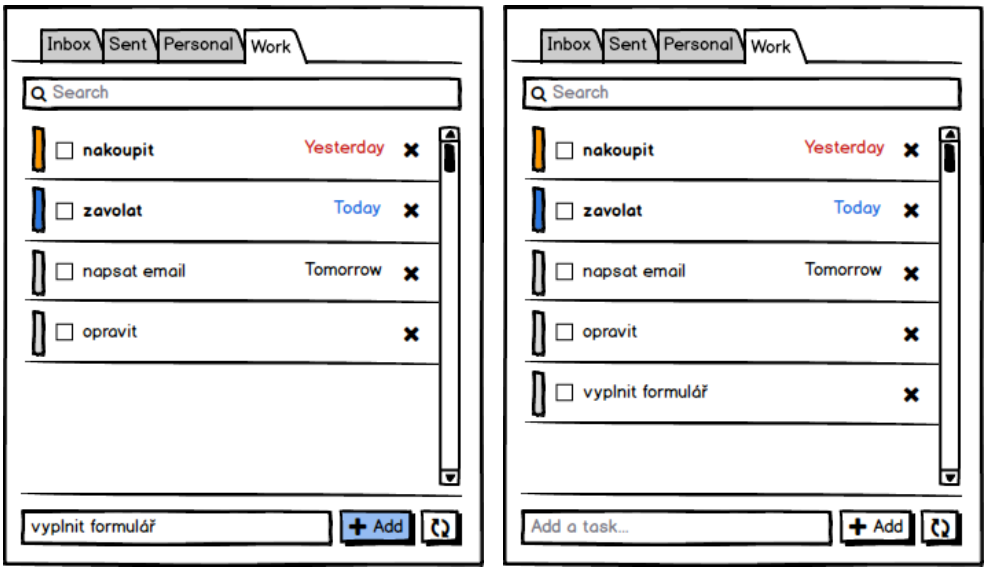
Skupinu úkolů sdružuje *seznam*, který je také třeba zobrazit. Každý *seznam* obsahuje jiné úkoly, a tak je nutné vystihnout onu kategorizaci také v grafickém rozhraní. Nabízí se řešení využít přepínatelné panely na horní hraně aplikace. Ty po jejich přepínání myší vždy zobrazí relevantní set úkolů, čímž se na stejně velké ploše aplikace umožní zobrazení různých úkolů. Tím je pokryto zobrazování seznamů a jím odpovídajících úkolů.

Dále je nutné nějakým způsobem snadno přidávat úkoly nové. K tomu je potřeba vstupního textového pole, skrze které uživatel bude nové úkoly vkládat. Potvrzení vložení by mělo být dostupné skrze tlačítko přidání, nejlépe co nejblíže danému poli. Pro usnadnění přidávání lze také přidat klávesovou zkratku, která po stisknutí klávesy „Enter“ uvnitř textového pole také provede vytvoření úkolu. Proces přidání úkolu lze vidět na obrázku 2.2.

Pro potřeby vyhledávání nastává otázka, jak tuto funkcionalitu zobrazit. Nejvíce uživatelsky přívětivým řešením je použití nového textového pole. Místo přidání potvrzovacího tlačítka, které by hledání uskutečnilo, jsem se rozhodl pro okamžité vyhledávání přímo při psaní vstupu. Jak lze vidět na obrázku 2.3, uživateli se tak zobrazují výsledky již během zadávání. Ty je možné zobrazit jako každý jiný *seznam*, nejlépe na stejném místě jako ten původní.

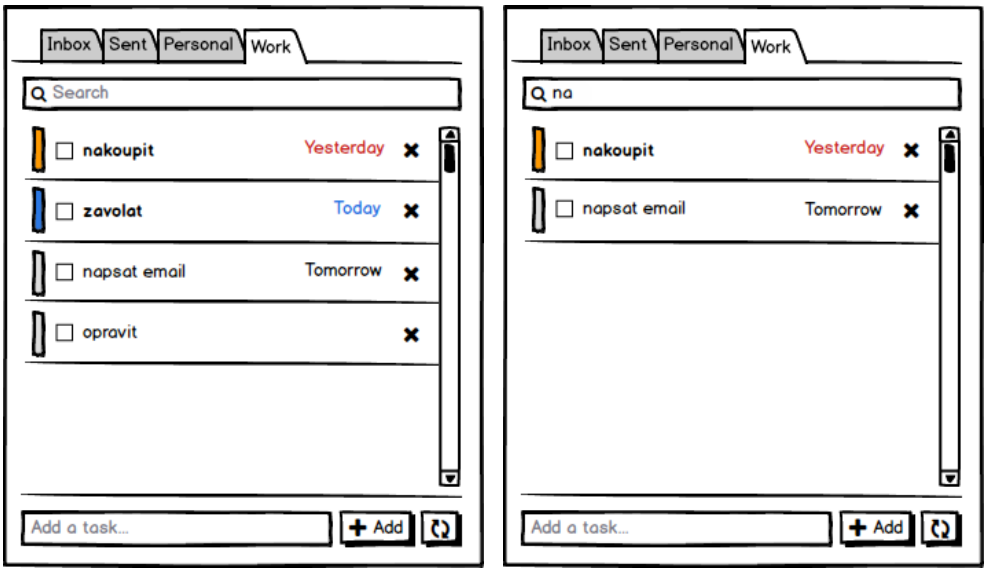
Dále aplikace vyžaduje obsluhu nastavení. Na obrázku 2.4 lze vidět, že nastavení půjde zobrazit kliknutím pravého tlačítka myši kdekoliv na ploše *plasmoidu*.

V nastavení jde především o možnost přihlášení se do služby. Pro přihlá-



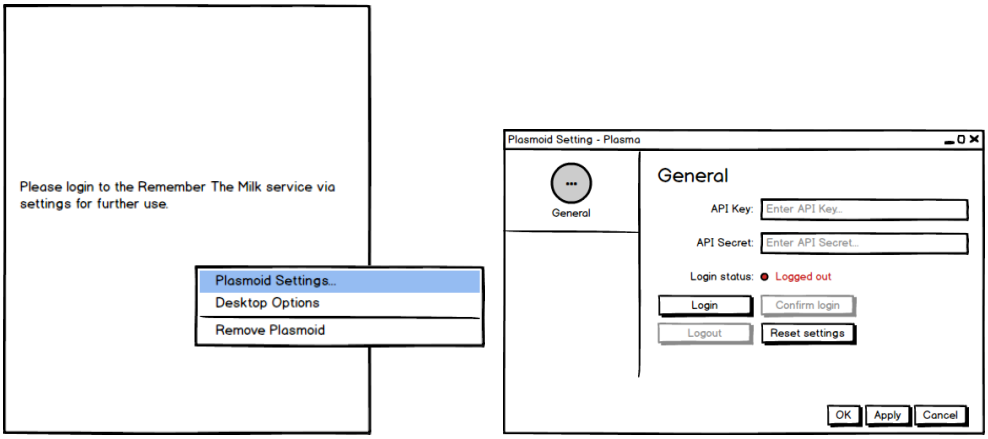
(a) Vyplnění nového úkolu (b) Zobrazení nového úkolu

Obrázek 2.2: WF2: Přidání nového úkolu



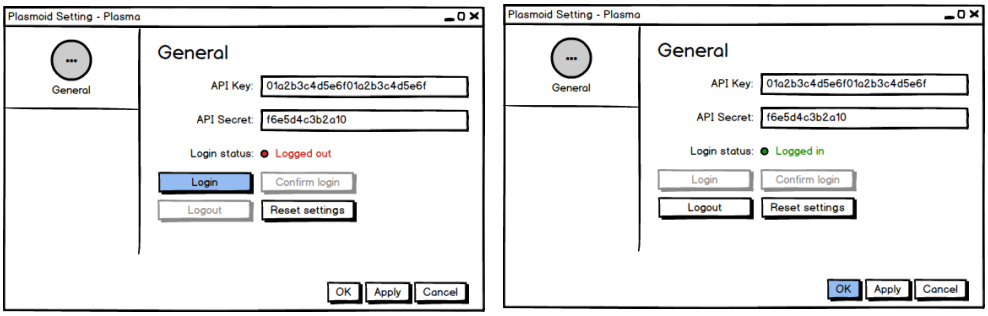
(a) Nefiltrovaný seznam úkolů (b) Vyfiltrované úkoly dle hledání

Obrázek 2.3: WF3: Vyfiltrování úkolů



(a) Otevření nabídky (b) Zobrazení okna nastavení

Obrázek 2.4: WF4: Zobrazení nastavení



(a) Spuštění procesu přihlašování (b) Úspěšně přihlášený uživatel

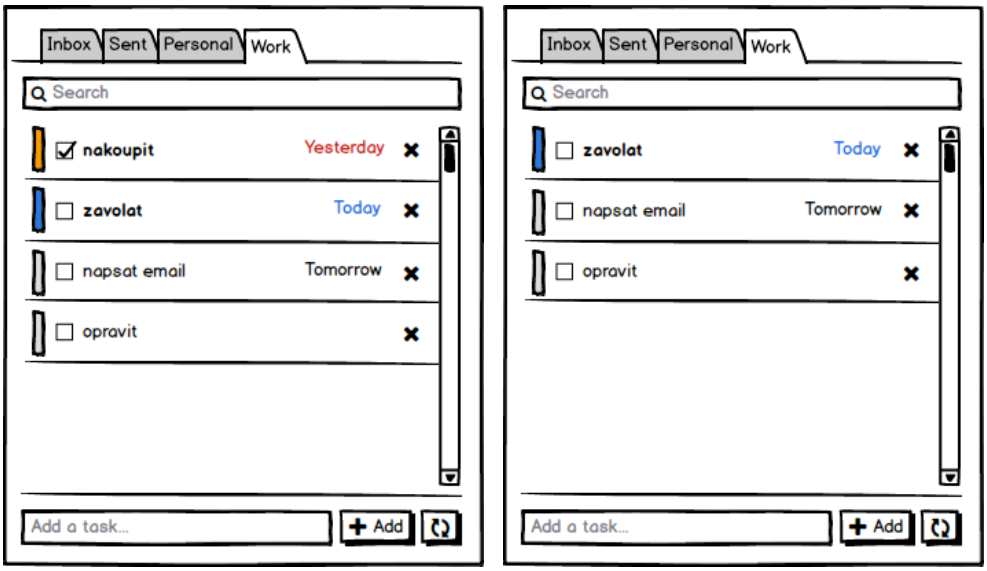
Obrázek 2.5: WF5: Přihlášení uživatele

šení jsou dostupná tlačítka „Login“ a „Confirm login“. Samotné přihlášení se odehrává skrze webovou stránku služby, kterou jsem se nakonec rozhodl otevírat v externím prohlížeči mimo aplikaci. Zaprvé se tím umožní uživateli přihlášení v jeho používaném prohlížeči, kterému může věřit, a zadruhé se nemusí zbytečně implementovat vnořené okno vestavěného prohlížeče do rozhraní aplikace. Stav přihlášení notifikuje textový popisek s barevnou ikonou viz obrázek 2.5.

Splnění úkolu probíhá dle obrázku 2.6. Po zaškrtnutí se splní a zmizí ze seznamu. Odstranění je obdobný proces, jak je vidět na obrázku 2.7, po odstranění také zmizí z aktuálního seznamu.

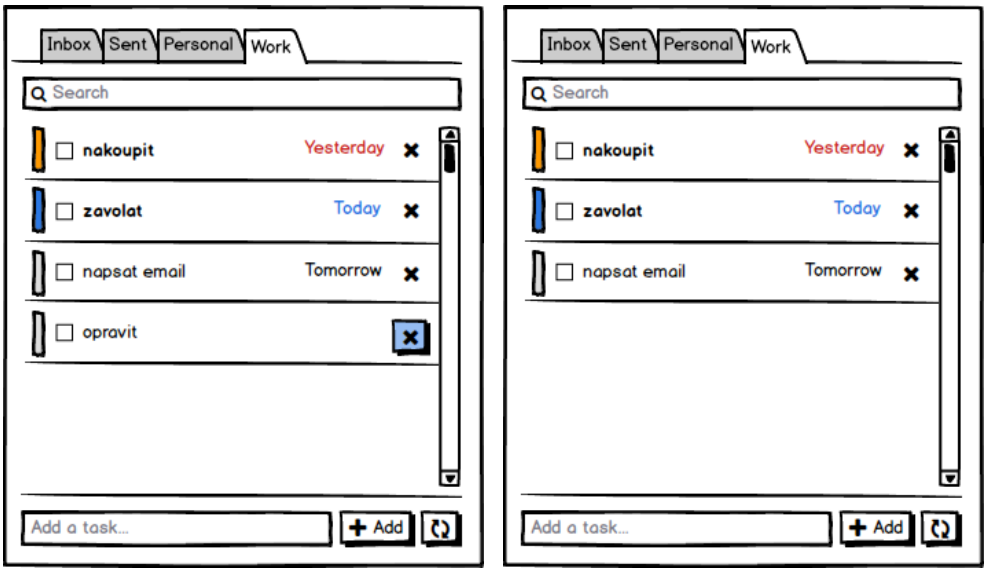
2.1.3 Souhrn

V tabulce 2.1 lze vidět, které návrhy rozhraní pokrývají jednotlivé případy užití. Jelikož jsou všechny případy užití pokryty, lze usuzovat, že rozhraní



(a) Vybrání úkolu jako splněný (b) Seznam po splnění úkolu

Obrázek 2.6: WF6: Splnění úkolu.



(a) Vybrání úkolu pro smazání (b) Seznam po smazání úkolu

Obrázek 2.7: WF7: Odstranění úkolu

Tabulka 2.1: Pokrytí případů užití wireframy

	UC1	UC2	UC3	UC4	UC5	UC6	UC7
WF1	X						
WF2		X					
WF3			X				
WF4				X			
WF5					X		
WF6						X	
WF7							X

umožňuje splnit požadovanou funkcionalitu.

2.2 Remember the milk API – rozhraní služby

Jak jsem již dříve zmínil, služba nabízí veřejné rozhraní sloužící pro přístup k ovládání účtu služby na serveru. Máme tedy prostředky dostupné pro tvorbu svého vlastního klienta. K dispozici je celá sada různých metod, které jsou seskupené do logických celků. Postupně projdu každý z nich [8].

2.2.1 Autorizace

Tato kategorie obsahuje metody potřebné pro správnou autorizaci uživatele. To probíhá pomocí získání tokenu, který má omezenou platnost. K potřebám přihlašování zde jsou metody na získání a ověřování tokenu.

2.2.2 Kontakty

Různé metody pro získání a manipulaci všech vedených kontaktů uživatele. Kontaktem by mělo být uživatelské jméno či email uživatele *služby*. Je umožněno získání seznamu kontaktů, jejich přidávání a odebírání.

2.2.3 Skupiny

Jednotlivé kontakty je možné seskupovat do různých logických celků. Rozhraní nabízí práci s těmito skupinami, vypsání jejich seznamu, přidání a odebrání kontaktu ze skupiny, případně přidávání a odebírání celých skupin.

2.2.4 Seznamy

Seznamy sdružují samotné úkoly. Máme možnost vypsat všechny seznamy. Dále je přístupné přidání a smazání seznamu, nastavení jeho jména, označení jako defaultní list a také archivace.

2.2.5 Lokace

Všechna uložená místa v rámci *služby* jsou evidována jako objekty lokace, které obsahují název místa a jeho adresu společně s zeměpisnou šířkou a délkou. Pomocí metody můžeme získat seznam všech těchto míst. Místa nelze skrze rozhraní přidávat ani upravovat.

2.2.6 Reflexe

Metody, které umožňují výpis všech dostupných volání. Také lze získat konkrétní informace o každé z nich.

2.2.7 Nastavení

Pomocí těchto metod jsme schopni získat uživatelské nastavení. To nelze měnit, jde pouze číst. Nastavení obsahuje následující informace:

- časová zóna,
- formát data (evropský nebo americký),
- formát času (12 nebo 24 hodin),
- defaultní seznam,
- jazyk.

2.2.8 Úkoly

Největší část rozhraní přirozeně obsluhuje úkoly samotné. Můžeme je přidávat, mazat, plnit, přesouvat mezi seznamy, měnit prioritu, přidat/mazat/upravovat poznámky, posouvat na jiný den, přidávat tagy, lokace a mnoho dalšího...

2.2.9 Testy

Testovací volání určená pro základní testování komunikace se službou. Najdeme zde typické *echo*, při kterém server odpoví posláním všech odeslaných parametrů zpátky. Také lze ověřit, zda je daný uživatel přihlášen nebo ne.

2.2.10 Čas

Metody určené ke správné manipulaci s daty. Můžeme konvertovat čas z jednoho časového pásma do druhého a parsovat různé formáty dat.

2.2.11 Časová linie

Zprostředkovává přístup k vytvoření nových časových linií (timelines). Ty umožňují určité transakce v rámci těchto linií vracet. Díky časovým liniím je tedy umožněn mechanizmus zpětných oprav.

2.2.12 Transakce

Pomocí těchto metod lze navrátit některé provedené změny do původního stavu, dříve než byla transakce provedena. Typická funkcionalita *vrátit zpět*. Je nutné specifikovat, v rámci které časové osy se má transakce navrátit.

2.2.13 Časové zóny

Můžeme získat seznam vedených časových pásem.

2.3 Remember the milk API – komunitní knihovny

Komunita služby je poměrně aktivní, existuje tudíž mnoho fungujících knihoven využívajících hlavního *API* služby pro různé programovací jazyky. Ve většině případů se jedná o podporu odesílání požadavků na server pomocí prostředků dostupných v daném programovacím jazyce a platformě. Programátor potřebuje specifikovat pouze název metody a její atributy a knihovna odešle požadavek na server a navrátí případnou odpověď. Tu je nutné dále zpracovat podle vlastních požadavků. Mezi důležité jazyky s dostupnými knihovnami momentálně patří [9]:

- C
- C++
- C#
- Java
- Javascript
- Objective-C
- PHP
- Python
- Ruby

2.4 Plasma API – rozhraní QML

Plasma shlukuje základní knihovny, nástroje a komponenty potřebné pro systém plochy vytvořené v této technologii. Je postavena na jazyce *C++* a *QML*. Samotné *QML* komponenty jsou [10]:

- org.kde.plasma.core,
  - obsahuje vazby na *C++* knihovnu *libplasma*,
- org.kde.plasma.components,
  - obsahuje grafické prvky jako tlačítka, panely, zaškrtávací políčka...
- org.kde.plasma.extras,
  - rozšiřující grafické prvky, které nejsou obsaženy v *components*,
- org.kde.plasma.plasmoid,
  - obsahuje prostředky pro manipulaci s plasmoidem.

Knihovna *libplasma* zajišťuje renderování SVG témat, přístup k datům a načítání struktury plochy.

Nejvíce důležitým prvkem bude právě *org.kde.plasma.components*, protože obsahuje většinu potřebných prvků pro tvorbu *plasmoidu*. Je také možné použít prvky přímo z *QtQuick*. Ty jsou ve většině případů základem prvků *plasmy*. Použití prvků přímo z *plasmy* má ovšem výhodu toho, že podporují změny vzhledu celého systému. Pokud tedy uživatel používá jiné téma plochy, *plasmoid* využívající těchto prvků se zobrazí se vzhledem nastaveného skinu.

2.5 Diagram tříd

Při tvorbě diagramu tříd, který lze vidět na obrázku 2.8, jsem se zaměřil na dříve zmíněná rozhraní *služby* a systému *KDE Plasma 5*, ve kterém bude aplikace fungovat. Dále jsem rámcově vycházel z obecnějšího doménového modelu (obr. 1.6), který nyní bylo třeba doplnit o platformově specifická rozšíření. Aplikace je nyní rozdělena do čtyř různých tříd, které zajišťují obsluhu *služby* a další nutnou režii. Jedná se o následující třídy:

- System,
- Settings,
- Task,
- List.

2.5.1 System

Třída *System* se stará o celkovou správu činností potřebných k tomu, aby bylo možné manipulovat se *službou* jako takovou. Především se jedná o získání seznamů s úkoly a úkolů samotných. Dále přidávání nových úkolů, splňování či mazání.

2.5.2 Settings

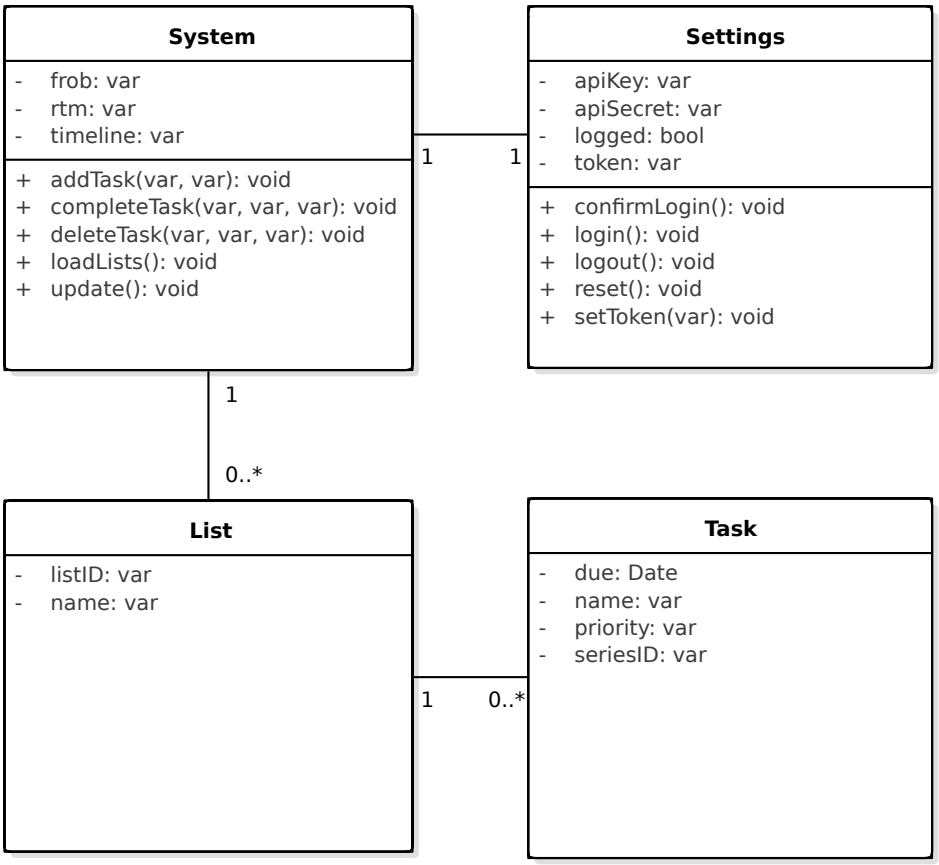
Tato třída sdružuje aktuální nastavení systému. Je třeba zajistit úspěšné přihlášení a případné odhlášení uživatele. Uchovává aktuální token, jednotlivé *API* klíče a stav přihlášení.

2.5.3 Task

Task je třída určená k sdružování dílčích atributů daného úkolu. Každý atribut je získán z volání třídy *System*, případně po vytvoření nového úkolu přímo v aplikaci.

2.5.4 List

Třída *List* reprezentuje daný seznam, který obsahuje jednotlivé úkoly. Je opět získán skrze volání třídy *System*. Při práci se seznamem nás zajímá především jeho název.



Obrázek 2.8: Diagram tříd

Kapitola 3

Realizace

Pro realizaci jsem se rozhodl použít jazyk *Javascript*, který lze vhodně napojit do miniaplikací využívajících *QML* v ekosystému *KDE Plasma 5*. V samotném *QML* lze přímo importovat javascriptové soubory a připojovat jejich funkce na prvky *Plasmy*. Pro tento jazyk existuje komunitní knihovna, která je schopna volat jednotlivé metody dostupné v *API* služby *Remember the milk*. Jedná se o knihovnu *rtm-js*, kterou vytvořil programátor Michael Day [11].

Co se grafického rozhraní aplikace týče, pracovat budu s prvky obsaženými v *org.kde.plasma*, a to především *org.kde.plasma.components*. Dále využiji framework *Qt* pro specifičtější prvky a funkce jako například externí otevírání odkazu apod.

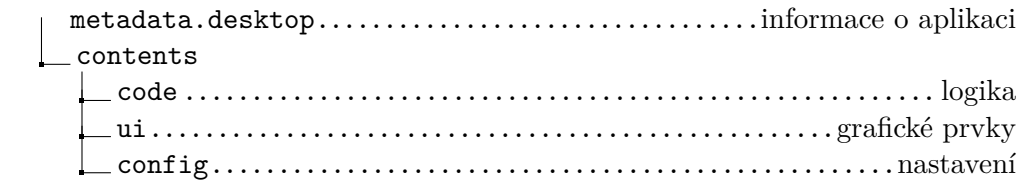
Jakožto vývojové prostředí využívám *Linux Mint* verze 18.1 obsahující grafické prostředí *KDE Plasma* ve verzi 5.8.2. Při implementaci aplikace využiji textového editoru *Sublime Text 2* a programu *Qt Creator*. Rychlé spouštění aplikace při vývoji lze docílit pomocí terminálové aplikace *plasmawindowed*. Pro samotnou instalaci *plasmoidu* slouží *plasmapkg2*.

3.1 Základní struktura plasmoidu

Aplikace typu *plasmoidu* se skládá z několika částí. Jednou z nich je soubor *metadata.desktop*, ve kterém jsou uvedené základní informace o aplikaci (název, popis, kontakt autora, verze, licence...) a také informace o specifikaci jazyka, hlavního souboru pro spuštění či typ služby [12].

Ostatní zdrojové soubory jsou umístěny ve složce *contents*. Obsah se dále typicky dělí na složky:

- ui,
- code,
- config.



Obrázek 3.1: Adresářová struktura *plasmoidu*

Pro bližší představu je struktura adresářů zobrazena na obrázku 3.1. Do složky *ui* se umisťují *QML* soubory definující grafické rozhraní a jeho provázání s logickou vrstvou. Nachází se zde hlavní soubor *main.qml*, který se načítá jako první při spuštění *plasmoidu*. Ostatní *QML* soubory obsahují typicky dílčí části definující jednotlivé obrazovky či prvky a volají se z *main.qml*. V mém případě se jednalo o vlastní prvky:

- TaskView.qml,
- PriorityStripe.qml,
- LoginWarning.qml.

Také je zde umístěno definované rozhraní pro nastavení aplikace, které se načítá systémem do systémového okna.

Uvnitř *code* se nachází většinou logická část programu, v mém případě tedy soubory *Javascriptu* a další pomocné knihovny. Tyto soubory se primárně importují do souboru *../ui/main.qml* a jejich metody se dají přímo v něm volat. To umožňuje dobré oddělení logické části programu od zbytku aplikace.

Adresář *config* obsahuje konfigurační soubory zejména formátu *XML*, které jsou využívány pomocí dialogu nastavení. Po zavření plasmoidu se nastavení serializuje právě do tohoto souboru nastavení. Následně po spuštění jsou data deserializována a opětovně použita.

3.2 Grafické rozhraní

Při vývoji jsem se rozhodl nejprve vybudovat prototypovou miniaplikaci se základními grafickými prvky, které nabízí *org.kde.plasma.components* a *QtQuick*. Jelikož jsem se s mnohými věcmi ohledně *KDE*, *Plasmy*, *QML* a *Qt* setkal poprvé, nebyla tato část pro mne nijak snadná. Nicméně jsem nakonec byl schopen z různých návodů, nepříliš podrobné dokumentace a především ostatních projektů leccos vyčíst a poznat tak souvislosti mezi stěžejními prvky, které ve výsledku tvoří samotný *plasmoid*.

PlasmaComponents obsahuje hlavní prvky potřebné pro tvorbu grafického rozhraní. Použil jsem prvku *TabBar*, který umožňuje zobrazení vícero stránek na celé ploše *plasmoidu* s možností přepínání. Dále pro přidávání nových úkolů

šlo využít prvky *TextField* a *Button*. Pro samotné zobrazení obsahu byl klíčový *TabGroup* obsahující posuvný seznam tvořený objekty *ScrollArea* a *ListView*. Samotné buňky listu jsou složené z *Buttonu*, *Labelu* a *Buttonu*.

3.3 Napojení na Remember the milk API

Následně bylo klíčové zprovoznění samotného napojení aplikace na rozhraní služby. Mnou použitá knihovna *rtm-js* je takzvaný *wrapper* (obalení), pomocí kterého lze volat jednotlivé metody rozhraní a získávat jeho odpovědi. Do *plasmoidu* jsem tuto knihovnu vložil a její funkčnost navázal na stěžejní části aplikace v *QML*.

Požadavky klienta jsou prováděny skrze *REST* volání. Odpovědi ze serveru chodí formátované jako *XML*, případně *JSON*. Pro potřeby aplikace jsem využil formátu *JSON*, protože má blíže k *Javascriptu*.

Během testování kompatibility na ostatních systémech jsem narazil na několik problémů. Hlavním z nich byla skutečnost, při které se nedařilo vytvořit novou instanci objektu knihovny *rtm-js* určené k odesílání a přijímání komunikace se serverem *služby*. Ve výpisu v konzolové řádce program hlásil, že objekt *RememberTheMilk* (při volání *new RememberTheMilk()*) není definovaný. Tento problém byl velmi zapeklitý, jelikož v mém běhovém prostředí všechno fungovalo správně. Podezříval jsem z problému fakt, že v *QML* se veškeré javascriptové soubory importují dovnitř jiných javascriptových souborů pomocí následujícího vzoru:

```
.include "folder/file.js" as Identifier
```

Identifier je libovolný název, pomocí kterého se přistupuje k veškerým funkcím daného souboru. Volání funkce *helloWorld* obsažené v souboru *greetings.js* tedy vypadá následovně:

```
.include "greetings.js" as Greetings

function someFunction () {
    Greetings.helloWorld();
}
```

Správným voláním konstruktoru by tak mělo být následující:

```
.include "rtm.js" as RTM

function init () {
    var rtm = new RTM.RememberTheMilk();
}
```

Nicméně knihovna samotná se uvnitř chová na principu *factory* (továrny). Nelze tak explicitně volat některou metodu, je nutné vytvářet novou instanci

objektu *RememberTheMilk*, který je umístěn ve stejnojmenném globálním objektu obsazeném při importování knihovny. Volání konstruktoru se specifikovaným identifikátorem tak nedávalo smysl a program s tímto způsobem vytváření nefungoval ani na mém funkčním běhovém prostředí.

Pokoušel jsem se také o importování pomocí funkce *Qt.include*, která se chová méně sofistikovaněji a zdrojový kód obyčejně kopíruje do cílového souboru. Nicméně problém stále přetrvával.

Po delším zkoumání jsem zjistil, že soubor samotný se importoval správně, jelikož se části tvorby nové *factory* prováděly správně, dokonce se objekt *RememberTheMilk* správně obsadil. Stále však toto propojení za žádnou cenu nechtělo fungovat.

Nakonec jsem porovnával rozdíly obou prostředí. Došel jsem ke zjištění, že se neshodovaly verze *Plasmy*. Funkční prostředí obsahovalo novější verzi 5.8.3, kdežto prostředí, ve kterém se projevil specifikovaný problém, bylo ve verzi 5.5.5. Provedl jsem tak upgrade systému pomocí terminálu. Ten povýšil *Plasmu* na verzi 5.8.5, která fungovala naprosto bez problémů.

3.4 Přihlášení

Takzvaným „hlavním kamenem úrazu“ v existujících projektech byla skutečnost nemožnosti přihlášení se ke službě v rozhraní *KDE Plasma 5*. Jedná se tedy o stěžejní část samotné implementace. Bez správného přihlášení k účtu uživatele nelze splnit a zajistit funkčnost aplikace. Přihlašovací proces je službou umožněn pomocí následující návaznosti kroků [13]:

1. získat takzvaný *frob* pomocí metody *rtm.auth.GetFrob*,
2. sestavit *URL* adresu pro autentizaci a otevřít ji uživateli,
3. uživatel se přihlásí a umožní aplikaci použít jeho účet,
4. získat *token* pomocí metody *rtm.auth.getToken*, který je použit pro veškerou další komunikaci vyžadující přihlášení.

3.4.1 Získání frobu

Frob je číselná sekvence v šestnáctkové soustavě, kterou potřebujeme pro pozdější komunikaci se serverem.

3.4.2 Sestavení URL adresy požadavku

Je třeba vytvořit celou adresu pro požadavek. Základem této adresy je v pořadí:

1. prefix adresy – *https://www.rememberthemilk.com/services/auth/*,

2. *API klíč*, což je číselná sekvence v šestnáctkové soustavě přidělená službou vývojáři aplikace,

3. oprávnění zda chceme číst, zapisovat, mazat,

4. *frob* získaný dříve,

5. podpis vytvořený podle následujícího postupu.

Získání podpisu:

1. seřazení parametrů v *URL* abecedně podle názvů klíčů,

2. spojení takto seřazené dvojice klíč hodnota za sebe,

3. toto celé se přidá za *API* tajemství, což je další sekvence v šestnáctkové soustavě (jedná se v principu o privátní klíč),

4. z výsledného řetězce se vypočítá MD5 hash, která je daným podpisem.

Pro výpočet MD5 je nutné knihovně *rtm-js* doplnit globální objekt *md5.js*. Využil jsem tedy další knihovnu a to *md5.js* od Josepha Myerse [14].

V této části jsem narazil na zásadní problém. Byl jsem schopen získat *frob*, komunikace skrze knihovnu a veškeré podpisy atd. tedy bylo funkční. Nicméně jsem po přesměrování uživatele na adresu autentizace získal požadovaný přihlašovací dialog, uživatel se přihlásil, potvrdil přístup a v ten moment server zobrazil dále nic blíže specifikující hlášku „něco se pokazilo“. Nebyl jsem tedy dále schopen pokračovat v přihlašovacím cyklu.

Analyzoval jsem tedy veškerou komunikaci od počátku až do tohoto bodu, a zaměřil jsem se především na správné odesílání adres. Po dlouhém ladění jsem nalezl chybu uvnitř *rtm-js* knihovny. Při každém požadavku na server se v adrese specifikovalo také formátování případné odpovědi, jednalo se o *JSON*. Nicméně při konstruování adresy se odpověď serveru bezprostředně neočekává, a tedy zde tento parametr nemá co dělat. Výsledkem bylo, že server se přes tuto nuanci nedokázal překlenout a vyhazoval dříve zmíněnou nespecifikovanou chybovou hlášku. Po upravení knihovny tak, aby při tomto požadavku neodesílala příslušný parametr jsem byl úspěšně schopen aplikaci přidělit práva.

3.4.3 Získání tokenu

Po přihlášení uživatelem v okně prohlížeče a udělení práv můžeme zavolat metodu pro získání tokenu. Jedná se o metodu *rtm.auth.getToken*. Pokud vše proběhlo v pořádku, získáváme v odpovědi kýžený *token*, který dále budeme využívat pro potvrzení přihlášení uživatele.

3.4.4 Uchování přihlášení

V této části implementace jsem se potýkal s následujícím problémem. Po získání přihlašovacího *tokenu* je nutné zaručit jeho uchování. Při běhu *plasmoidu* je uložen v paměti procesu, nicméně důležité je jeho uložení při ukončení aplikace a následné načtení při startu. K tomuto problému jsem se chystal přistoupit jednoduše a tedy uložit token do souboru. V tomto bodě jsem však narazil na limitace *Javascriptu*, protože s jeho pomocí nelze standardní cestou v *QML* zapisovat/číst soubor. Pro tyto účely je vhodnější využít napojení pomocí jazyka *C++*. Podařilo se mi pomocí *XMLHttpRequestu*, který v *QML* zvládá bohužel podmnožinu jeho celkových funkcí, asynchronně načíst obsah souboru z disku. Jenže funkcionalita uložení zde bohužel není umožněna.

Musel jsem se tedy vydat jinou cestou. *Plasmoid* samotný nabízí možnost udržovat určitá data typu jednoduchého nastavení. Pro tyto účely je nutné utvořit *kcfg* soubor, což je strukturované *XML* s daným formátem. V tomto souboru lze nadefinovat položky, které se budou sdílet jako aktivní nastavení *plasmoidu* při běhu, a zároveň se automaticky uchovají při jeho restartování. Tato funkcionalita je velmi užitečná a dá se poměrně snadno využít. Každá položka má svůj unikátní název a defaultní hodnotu.

Tento *kcfg* soubor je dále nutné napojit v *QML* na rozhraní rozšířeného nastavení. Plasma nabízí skrze kliknutí pravým tlačítkem myši na *plasmoid* zobrazení jeho nastavení. Toto nastavení lze dále rozšířit a přidat do něj různé prvky jako tlačítka či textboxy pro uživatelské nastavení.

Samotné propojení s *kcfg* se provede tak, že všem atributům z tohoto souboru, které chceme propojit s prvky rozhraní, nastavíme hodnoty těchto prvků jakožto *property alias*. Čili například pokud máme v *main.xml* atribut s názvem *name*, v nastavení vytvoříme *property alias* s názvem *cfg_name* a přiřadíme mu hodnotu prvku rozhraní, například *nameInput.text*, kde *nameInput* je *ID* prvku textbox. Cokoliv, co uživatel napíše do tohoto textboxu, se po zavření nastavení uloží jako položka dostupná přes *plasmoid.configuration.name* v kterékoli části plasmoidu [15].

Token tedy lze zachovat využitím této funkcionality.

3.5 Zobrazení úkolů

Než je možné úkoly zobrazit, je nejprve nutné získání seznamů. K tomuto účelu je určena metoda *API rtm.lists.getList*, pomocí které získáme soupis všech seznamů. Ty jsou rozlišeny unikátními číselnými identifikátory.

Pro získání úkolů daného seznamu slouží dále metoda *rtm.tasks.getList*. Musí se specifikovat *ID* seznamu, ze kterého chceme tyto úkoly vypsat. Také lze specifikovat určitý filtr, takže je možné například chtít jen neukončené úkoly. Vrátí se pak jednotlivé úkoly spolu se základními atributy. Z těchto atributů je pro účely klienta relevantní určitá podmnožina. Je potřeba uchovat název a identifikační čísla úkolu a série úkolu, prioritu a případný termín splnění.

Úkoly jsou obecně v různém pořadí, bylo je tedy potřeba seřadit dle požadovaných kritérií. Ve webové verzi služby je výhodné následující řazení (od nejvyšší priority po nejnižší):

1. termín splnění úkolu chronologicky,
2. priorita úkolu od nejvyšší po nejnižší,
3. název úkolu dle abecedy sestupně.

Pokud některý úkol nemá datum splnění, je zobrazen za všemi úkoly, které ho mají. Úkoly se stejnými hodnotami některé vrstvy kritérií se následně řadí podle nižší vrstvy. Tento typ řazení dokáže vhodně vystihnout a odlišit základní vlastnosti úkolů, a proto jsem jej implementoval v klientské verzi. Řazení probíhá vkládáním úkolu na správné místo v modelu, a to v průběhu zpracování strukturované odpovědi ze serveru. Není tak nutné řazení ex post dle některého řadícího algoritmu, což by zbytečně navýšilo náročnost zpracování.

Jelikož *API* posílá své odpovědi ve formátu *JSON*, lze tato data snadno extrahovat a propojit do modelu v *QML*. *QtQuick* nabízí prvek *ListModel*, do kterého lze získaná data uložit podobně jako do pole. Jednotlivé položky jsou javascriptové objekty, lze tedy pod daný klíč například *name* uložit název úkolu. Ten poté je z modelu dostupný jako *listModel.get(i).name*, kde *i* je index položky.

ListModel nabízí snadnou obsluhu dat a případné další použití v dynamicky vytvářených objektech. Je ho možné přiřadit prvku *ListView*, což je typický posunovací seznam položek, který můžeme znát například z *Androidu* či *iOS* na mobilních telefonech. Toto propojení zajistí, že podle obsahu v modelu se vykreslí jednotlivé položky dynamicky v *ListView*.

Každá položka *ListView* je delegována jednotlivým prvkem *PlasmaComponents.ListItem*. Uvnitř lze definovat další části. Pro účely zobrazení seznamu úkolů jsem využil následující prvky:

- PriorityStripe,
- PlasmaComponents.Button,
- PlasmaComponents.Label,
- PlasmaComponents.ToolButton.

PriorityStripe je ručně vytvořený prvek založený na objektu *Rectangle*, který svou barvou reprezentuje prioritu daného úkolu. Tento svislý tenký pruh na začátku řádku odpovídá stejnému prvku ve webové verzi klienta. Jednotlivé barvy taktéž korespondují svému webovému protějšku.

Button umožňuje interakci s úkolem. Kliknutím na něj lze daný úkol splnit. Tento prvek se nachází na místě checkboxu na webu, odpovídá tedy obdobnému grafickému rozhraní. Nicméně funkcionalita byla vylepšena pro účely snadného použití v *plasmoidu*.

Label byl použit jak na zobrazení názvu, tak i případného termínu úkolu. Právě termín určuje vzhled těchto dvou položek. Na základě data se vyhodnotí, zda zobrazit text úkolu tučně, což odpovídá tomu, že termín splnění je dnes či dřívějšího dne. Též se mění barvy termínu. Pro dnešní je zobrazen modře, pro starší slouží výrazná červená a v ostatních případech je defaultní černá. Samotné datum se zobrazuje následovně. V rozmezí termínu „včera“, „dnes“, „zítra“ se zobrazují právě tato pojmenování. Dále v rozmezí dalších následujících 7 dní se zobrazují pouze názvy dní, tedy například „středa“. Jakékoliv další datum se zobrazí standardně jako kombinace čísla dne a názvu měsíce, tedy například „5. května“. Toto veškeré chování je identické jako ve webové verzi klienta. To zaručuje zachování stejných zvyklostí pro uživatele.

Pro smazání úkolu jsem využil prvku *ToolButton*. Na pravé straně se zobrazí toto tlačítko s ikonou pro smazání, které se následně po stisknutí provede.

3.6 Přidání úkolu

K přidání úkolu slouží metoda *rtm.tasks.add*. Té je potřeba určit parametry úkolu, tedy jeho název, *ID* seznamu kam se má vložit a v rámci jaké *timeline* probíhá tato transakce. Na základě těchto informací následně server služby odpoví. V případě úspěšného přidání obsahuje tato odpověď i souhrn předcházejících vlastností.

Samotný název úkolu je zadán uživatelem v textovém poli. Zde může specifikovat pomocí klíčových znaků prioritu, datum do kdy má být splněno, či seznam. Pokud není specifikováno jinak, úkol se vloží do aktuálně zobrazeného seznamu. Po stisknutí tlačítka pro přidání nebo klávesy „Enter“ se úkol odešle serveru.

Problémem bylo přidávání úkolů do správného seznamu, pokud byl v názvu specifikovaný jiný seznam než aktuální. Server vložil úkol pod seznam, který uživatel definoval, zatímco v aplikaci bylo vložení míněné do aktuálně zobrazeného seznamu. Ošetřil jsem daný problém tak, že jsem informaci o přiřazeném seznamu získal přímo z odpovědi serveru namísto z lokálních zdrojů.

Důležitým faktorem je zde skutečnost, že přidaný text úkolu je zpracován analyzátorem. Ten rozpozná klíčové znaky 9 (!, #, ^, ~) před jednotlivými slovy a tím dokáže například určit do jakého seznamu chceme daný úkol vložit, či jakou prioritu mu nastavujeme. Z tohoto hlediska je důležité, že server vrací v odpovědi i vložený úkol, jelikož mohl být vložený i do jiného než aktuálně zobrazeného seznamu. Toto chování je identické jako ve webové verzi služby.

Po samotném vložení úkolu do seznamu na straně serveru je nutné jej správně vložit a následně zobrazit i na straně klienta. Protože server navrátil

zpět celý objekt úkolu, je možné využít stávajících metod pro vložení jako při prvotním stažení celého seznamu úkolů. Úkol má přiřazeno číslo seznamu, do kterého se má vložit. Je tedy nutné pomocí tohoto čísla zjistit pozici panelu, ve které se tento seznam nachází. K tomuto účelu jsem využil asociativního pole, které tuto informaci uchovává. Po vložení do příslušného modelu panelu se přidaný úkol zobrazí mezi ostatními.

Dále jsem se potýkal s problémem ohledně přidávání úkolů s diakritikou. Pokud ji obsahovaly, server jejich přidání odmítl s tím, že prý neseděl hash podpis přijatých parametrů. Po delším zkoumání jsem objevil problém výpočtu MD5 hashe v importované knihovně. Když obsahovala diakritiku, výpočet hashe neodpovídal. Pro porovnání jsem použil online nástroj `http://www.md5.cz`. Vyměnil jsem tak původní knihovnu [14] za novou knihovnu *JavaScript-MD5* od Sebastiana Tschana [16]. V té tento problém nenastával a bylo tak možné i přidávání s diakritikou.

3.7 Splnění úkolu

Pro splnění úkolu obsahuje rozhraní služby metodu *rtm.tasks.complete*. Té je nutné specifikovat *timeline*, ve které transakce probíhá, identifikační čísla seznamu, úkolu a série úkolu. V případě úspěšného splnění úkolu server odpoví vrácením úkolu s pozměněným obsahem. V tomto případě se jedná o změnu atributu splnění (completed), ve kterém je nyní uloženo datum splnění.

Označení úkolu jako splněný lze docílit pomocí kliknutí levého tlačítka myši na zaškrtávací pole vedle textu úkolu. Po této akci se na položce seznamu spustí metoda pro splnění úkolu.

Splněný úkol je potřeba ze zobrazeného seznamu úkolů smazat, protože jsou vždy zobrazeny jen nesplněné úkoly. Při testování jsem zjistil, že pokud uživatel splnil úkol z vyfiltrovaného seznamu, tak se v klientovi odstranil z modelu úkol s jiným indexem. Přitom na serveru vše proběhlo správně. Po delším zkoumání bylo zřejmé, že se k odstranění úkolu využívalo špatně indexu filtrovaných úkolů namísto indexu těch základních.

Upravil jsem tedy odstraňování tak, aby využívalo příslušného *ID* úkolu i při lokálním mazání. Protože úkol mohl být zobrazen skrze filtrovaný model, je nutné jej odstranit jak z filtrovaného tak původního seznamu. Kvůli této skutečnosti nelze při specifikaci úkolu předávat i index položky. Samotný úkol se odstraní z obou seznamů jejich projitím.

3.8 Odstranění úkolu

Odstranění úkolu probíhá analogicky ke splnění úkolu. Rozhraní služby nabízí metodu *rtm.tasks.delete*, která vyžaduje stejné parametry jako u splnění úkolu. Po odstranění ze serveru získáme odpověď, ve které je změněn stav smazání.

Smazání úkolu lze provést pomocí kliknutí na tlačítko smazání vedle úkolu. Tím se spustí metoda vymazání úkolu ze seznamu.

Po odstranění ze serveru je nutné opět odstranit úkol také ze zobrazeného seznamu v klientovi. Pro tyto účely bylo možné použít stejných metod jako u splnění úkolu.

3.9 Vyhledání úkolu

V rámci zobrazených úkolů bylo třeba mít možnost v nich snadno a rychle vyhledávat. Pro tuto funkcionalitu lze využít následující postupy.

Rozhraní služby *Remember the milk* při získávání úkolů ze serveru specifikaci filtru. Do tohoto filtru lze definovat klíčové slovo pro vyhledání, například část názvu úkolu, a služba následně pošle klientovi v odpovědi odpovídající zúžený seznam úkolů. Tato varianta je ovšem závislá na neustálém dotazování na server, což je pro okamžité vyhledávání nevhodné.

Rozhodl jsem se tedy jít cestou místního vyhledávání. Pro tyto účely je určen textbox nad aktuálně zobrazenými úkoly. Cokoliv se do něj napíše, se okamžitě začne vyhledávat v aktuálním seznamu. Pokud některý z úkolů obsahuje hledaný text, je dále předán do výsledků vyhledávání. Jelikož se jedná o klientskou funkcionalitu v rámci aplikace, je možné využít okamžité vyhledávání a ihned zobrazovat výsledky.

Po změně aktuálního seznamu se vyhledávání provede také na seznamu změněném. Pro zrušení vyhledávání stačí jednoduše smazat text vyhledávání, ať pomocí klávesnice či ikony uvnitř textboxu.

Dalším rozhodnutím bylo jak zobrazit nově vyhledané výsledky. Původně jsem měl v plánu vytvoření zcela nového seznamu jakožto položky panelu a do něj následně zobrazit výsledky vyhledávání. Druhá možnost byla zobrazení vyskakovacího okna a do něj vložit vyhledané úkoly. Obě tyto varianty mají výhodu v tom, že nemění aktuální seznam úkolů. Není tedy potřeba jej měnit při vyhledání. Tato řešení jsou ovšem méně intuitivní a neprovádějí proces hledání zcela tak jak je obecně známé. Jediným východiskem tedy bylo zobrazovat výsledky hledání přímo do aktuálního modelu *ListView* a vyřešit režii okolo. V té jsem musel určit způsob výměny aktuálních modelů na základě toho, zda se vyhledává či ne. Když je textové pole prázdné, je aktivní původní model, při započetí vyhledávání je aktivní filtrovaný model.

3.10 Aktualizace

3.10.1 Získání nového obsahu

V rámci správného fungování aplikace bylo nutné zajistit plynulou synchronizaci úkolů a seznamů se serverem *služby*. Ideálním řešením tohoto problému by byla možnost push notifikací vytvářených na serveru a odesílaných směrem

ke klientovi. Provozovatel *Remember the milk* ovšem bohužel tuto možnost ve svém veřejně dostupném rozhraní nenabízí.

K dispozici tak byla pouze možnost komunikace iniciované ze strany klienta. Pro potřeby získání aktualizací seznamů a úkolů šlo použít:

- *rtm.lists.getList*,
- *rtm.tasks.getList*.

První jmenovanou metodou jsem byl schopný získat list uživatelských seznamů. Ten jsem dále porovnal se stávajícími seznamy načtenými v aplikaci.

Když se jejich skladba oproti předchozí verzi změnila, provedl jsem nad všemi seznamy opětovné načtení všech jejich úkolů. Toto řešení jsem vybral, jelikož se dá předpokládat, že se většina seznamů buďto změní co se obsahu týče, a nebo bude nutné načíst seznamy zcela nové. Navíc je tato varianta změněného sestavení seznamů méně častá z pohledu běžného uživatele.

Naopak při stejném setu seznamů bylo žádoucí získávat informace o úkolech změněných v určitém časovém rozsahu, ideálně tak od data poslední synchronizace se serverem. Naštěstí druhá jmenovaná metoda právě podporuje specifikování data poslední synchronizace. Přijímá datum ve formátu dle normy ISO 8601. Při volání metody s definovaným datem vrátí server ve své odpovědi klientovi pouze ty úkoly, které byly změněny či přidány od doby uplynulé od daného data [17].

3.10.2 Zpracování nového obsahu

Při zpracování odpovědi aktualizovaných úkolů bylo nutné vytvořit novou podpůrnou infrastrukturu pro získání potřebných informací. Nově vzniklé úkoly se vyskytovaly ve stejné struktuře jako při získání všech úkolů. Taktéž se chovaly změněné úkoly, které také obsahovaly všechny svoje atributy i kromě těch změněných. Jinak se však zobrazovaly úkoly odstraněné. Jejich souhrn se obsáhl do položky „deleted“. Ta dále seskupovala jednotlivé „taskseries“ a úkoly samotné. O nich se v odpovědi zobrazovalo pouze jejich ID. Odstraněné úkoly tedy šlo po správně upraveném parsování snadno rozpoznat od ostatních a pomocí ID zařídit jejich nalezení a následné odstranění.

Stejný princip jako u odstraněných jsem očekával od úkolů splněných, tedy seskupení do položky „completed“. Rozhraní se však chovalo jinak. Vmísilo tyto objekty mezi ostatní přidané a změněné. Zpracování úkolů přidaných a změněných tak vyžadovalo jejich vzájemně jednoznačné rozlišení.

Splněné úkoly jsem rozlišil pomocí atributu „completed“, který oproti ostatním úkolům obsahoval časovou známku data splnění. Následně bylo nutné provést jejich odstranění ze stávajících struktur v aplikaci. Nad odpovídajícím *task modelem* seznamu jsem provedl odstranění úkolu stejně jako při iniciování této aktivity ze strany klienta. Tím se splněné úkoly řádně propojily do obsahu klienta.

Zbývaly namixované úkoly změněné a nově přidané. Původně jsem měl v plánu vyhledat daný úkol mezi těmi existujícími v aplikaci. Pokud by existoval, tak bych jeho obsah aktualizoval novým. Naopak při neexistenci bych úkol vložil jako nový. Tuto variantu jsem ovšem po delším zkoumání zavrhl. Případné změny existujících úkolů totiž mohou mít vliv na jejich vzájemné řazení. Provedl jsem tak místo toho jejich odstranění a opětovné vytvoření a vložení, které zajistí správné umístění i po aktualizaci jejich obsahu.

Poslední úkoly nově přidané se v aplikaci nenalézaly, a tak jsem je mohl přidat stejným způsobem jako při počátečním získávání úkolů.

3.10.3 Automatizace získávání nového obsahu

Uživatel může provést aktualizaci stisknutím tlačítka k tomu určeného. Veškeré seznamy a úkoly se tak obnoví dle výše popsaného postupu. Tuto manuální aktualizaci je třeba doplnit možností periodického obnovování.

K tomuto účelu jsem použil *Qt* prvku *Timer*. Ten zprostředkovává funkci provádění určité funkce s nastavitelným časováním. Pro účely aktualizací bylo výhodné nastavení opakování každých několik minut. Uživatel může z nastavení v sekci „Refresh“ změnit počet minut, ve kterých se bude aktualizace provádět. Pomocí posunovacího jezdce může zvolit hodnotu mezi jednou až šedesáti minutami.

3.11 Obecné problémy

Jedním z problémů při změně prostředí byly některé chybějící ikonky použité v rámci aplikace. Vybíral jsem je ze standardního tématu *KDE*, nicméně na jiném systému, který měl například spuštěné méně detailní schéma tato ikonka zcela chyběla. Musel jsem tak tomu přizpůsobit a zúžit možný výběr ikon tak, aby bylo pokryto co nejvíce sad ikon.

Jak jsem zmiňoval dříve, po dobu implementace jsem využíval programu *plasmawindowed* pro rychlé zobrazení *plasmoidu* bez nutnosti restartování *plasmashellu*. Aplikace takto fungovala bez problému. Při zkoušení aplikace přímo z plochy jsem ovšem zjistil, že se po normálním běhu aplikace a úspěšném přihlášení k účtu nezobrazují stažené úkoly ani seznamy. Spustil jsem tak *plasmashell* přímo z terminálu, abych mohl číst výpisy *plasmy* a *plasmoidu*. Po průzkumu výpisu jsem zjistil, že se aktuálně nepotvrzené nastavení nedostalo k dalším částem programu, které ho vyžadovaly. Komunikace tedy nemohla proběhnout. Tento problém se neprojevoval u otevírání pomocí *plasmawindowed*, jelikož po jeho několikátém otevření již informace o nastavení *plasmoid* měl.

Problém jsem tak vyřešil uložením nastavení v momentě potvrzení přihlášení uživatelem. Další komunikace pak probíhala bez problémů.

3.12 Zlepšování kvality

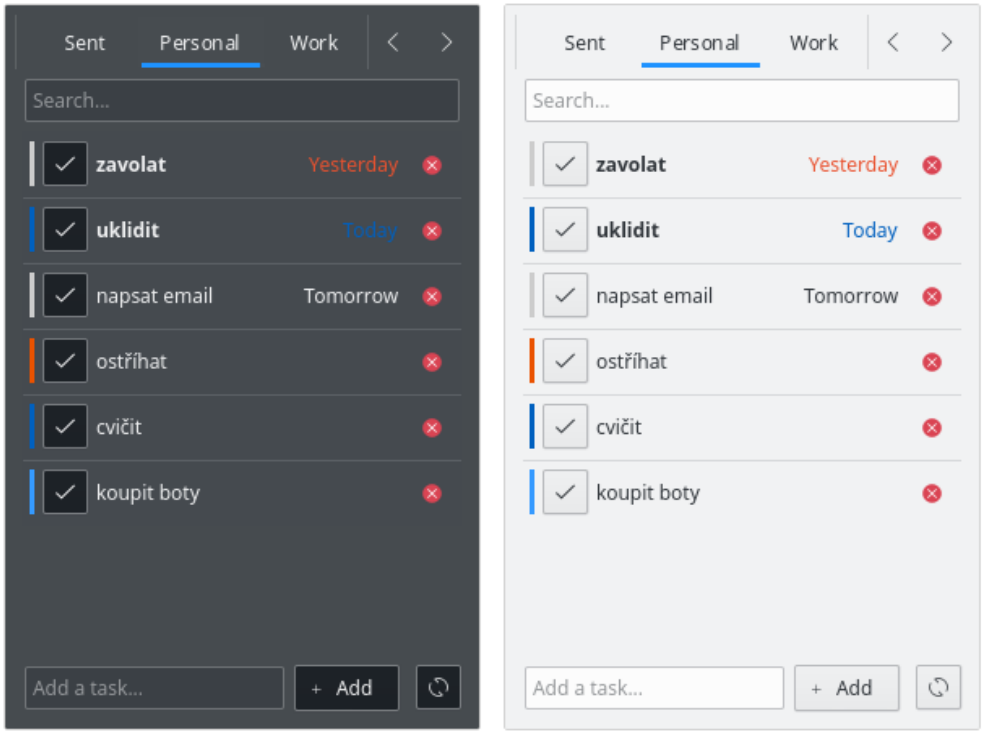
V rámci zlepšování kvality kódu jsem pro detekci takzvaných „mrtvých“ částí kódu využil funkcí vývojového prostředí (*IDE*). To také nabízelo možnosti zjednodušování a zkracování například logických výrazů. Dále jsem se snažil o co největší členění kódu do vlastních tříd. S tím souvisí provádění zpětného refactoringu ke zlepšení čitelnosti kódu.

3.13 Souhrn

Při tvorbě jsem čerpal především z různých návodů, za zmínku stojí web Davida Edmundsona a jeho návody „Plasmoid tutorial“ [18]. Ohledně základní komunikace se vzdáleným serverem mi pomohl tímto se zabývající článek od Juana M. De Hoyose [19].

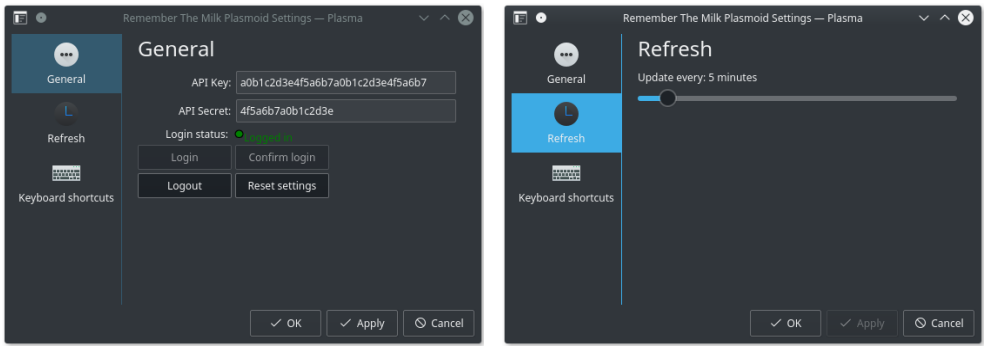
Bohužel nepříliš podrobná dokumentace nové *Plasmy* často znamenala, že jsem musel procházet materiály starších verzí a často zjišťoval, že některé skutečnosti jsou již zcela jinak. Velmi dobrou pomůckou tak byla neoficiální dokumentace *Plasmy* od Chrise Hollanda [15]. Jako nejvíce obsáhlou publikaci jsem používal *QmlBook* od Jürgena Bocklage-Ryannela a Johana Thelina, která obsahuje různé popisy práce s frameworkem *Qt*, *QML* a *Javascriptem* [20].

Výsledek průběhu realizace je aplikace, kterou lze vidět na obrázku 3.2. Reaguje na změny systémových témat, na obrázku 3.2 jsou zastoupeny příklady světlého a tmavého vzhledu. Nastavení aplikace lze dále vidět na obrázku 3.3.



(a) V systému s tmavým vzhledem (b) V systému s světlým vzhledem

Obrázek 3.2: Výsledný vzhled aplikace



(a) Obecné nastavení (b) Nastavení obnovování

Obrázek 3.3: Výsledný vzhled nastavení

Kapitola 4

Testování

4.1 Metody testování

Aplikaci bylo třeba v průběhu celé implementace řádně testovat. Prováděl jsem především testování funkční a systémová. Ta umožňovala objevování chyb vytvořených během implementace jednotlivých funkcí a aplikace jako celku. Dále jsem prováděl testování kompatibility a instalace programu v různých systémech obsahující *KDE Plasmu 5*. Závěry z těchto typů testování byly uvedeny v předchozí kapitole.

Po dokončení implementace jsem se skupinou testerů provedl testování použitelnosti. Jednotlivé výstupy testování rozeberu níže.

4.2 Testování použitelnosti

V rámci testování použitelnosti jsem provedl testování se skupinou testerů. Snažil jsem se pokrýt co nejširší spektrum uživatelů, kteří by mohli aplikaci používat. Kompletní pokrytí každého typu uživatele by bylo ovšem téměř nemožné. Pro testování jsem si připravil následující:

- testovací účet *služby*,
- testovací data,
- vstupní dotazník,
- popis testovacího scénáře,
- výstupní dotazník.

Jako testovací účet jsem použil nový běžný účet, který lze získat registrací na stránce *služby*. Do tohoto účtu jsem vytvořil testovací data. Jednalo se o nové seznamy a úkoly do nich přidané. Úkoly obsahovaly nejrůznější atributy k pokrytí co největšího spektra možností.

S jednotlivými testery jsem se odděleně sešel a zadal jim práci. Nejprve jsem testera požádal, aby vyplnil vstupní dotazník obsažený v příloze B. Snažil jsem se získat informace o zkušenostech v oblasti počítačů a informačních technologiích. Dále mne zajímala zkušenost s:

- operačním systémem Linux,
- desktopovým prostředím *KDE Plasma*,
- widgety na ploše některého z operačních systémů,
- nástroji pro organizaci úkolů,
- nástrojem *Remember the milk*.

Po vyplnění vstupního dotazníku nastal čas na provedení testovacího scénáře, který je obsažen v příloze C. Ten pokrýval jednotlivé scénáře použití. Ve zkratce se jednalo o:

1. přihlášení ke službě,
2. zobrazení úkolů v různých seznamech,
3. přidání nových úkolů s různými atributy do různých seznamů,
4. filtrování úkolů dle klíčových slov,
5. splňování úkolů,
6. odstraňování úkolů,
7. odhlášení z aplikace.

Během provádění testování jsem každému z testerů měřil čas, který vynaložil na splnění scénáře. Také jsem prováděl záznam plochy pro pozdější analýzu použitelnosti. Celý průběh jsem sledoval a vytvářel si poznámky ke každému úkolu. V případě technických nejasností mimo rámec účelu testování jsem s testery komunikoval.

Po provedení testovacího scénáře jsem opět požádal o vyplnění dotazníku. Výstupní dotazník je dostupný v příloze D. Snažil jsem se zjistit informace ohledně použitelnosti aplikace jako celku. Zajímalo mne, zda splnili veškeré zadané úkoly. Poté dotazník obsahoval otázky o dojmu z používání aplikace a vzhledu. Nakonec následovaly otázky ohledně kladů a záporů aplikace.

4.3 Testovací skupina

V testovací skupině jsem měl zástupce běžných uživatelů, pokročilých uživatelů i expertů. Z tohoto hlediska se jednalo o ideální zastoupení skupiny, jelikož každý uživatel měl ponětí minimálně o běžném používání technologií a zároveň jsem měl k dispozici i zkušenější uživatele.

Byli mezi nimi i zástupci uživatelů, kteří používají či někdy použili linuxový operační systém. Bohužel žádný z respondentů nikdy nepoužil desktopové prostředí *KDE Plasma*. Toto kritérium se mi bohužel nepodařilo pokrýt dostupným testerem.

Někteří účastníci někdy použili widget na ploše, jednalo se o widgety: „počasí, hodiny," či „kamerový systém, kontrola chlazení". Ostatní widgety nepoužívají. Dále pouze dva uživatelé aktivně používají nástroje pro organizace úkolů. Jeden z nich používá „Wunderlist", další využívá služby „OTRS Helpdesk" a „poznámky Android". Bohužel se mi pro testování nepodařilo získat uživatele služby *Remember the milk*.

4.4 Poznatky z průběhu testování

Z pozorování práce testerů jsem zjistil několik informací. První z nich byla určitá nejistota uživatele neznalého prostředí *KDE Plasma* při hledání nastavení *plasmoidu*. Nicméně po delší či kratší době nakonec pomocí kontextové nabídky nastavení našli. Jelikož se jedná o standardní obsluhu nastavení v tomto prostředí, dá se předpokládat, že uživatel znalý prostředí by s tímto neměl problém. Následné přihlášení probíhalo u každého bez problémů. Všichni tak postoupili dále k průzkumu zobrazeného obsahu. Najít zadané úkoly a zároveň procházet jednotlivé seznamy nikomu nedělalo potíže.

Při přidávání běžných úkolů jsem taktéž nezaznamenal zaváhání. Menším problémem ovšem bylo přiřazování atributů přidávaným úkolům. Tento nedostatek byl z velké části dán neznalostí možností dané služby. Nicméně po nalezení nápovědy zobrazované při najetí myši na textbox přidání byla testovací skupina schopna daným zkratkám porozumět a použít je při tvorbě úkolů.

Jeden z testerů objevil nedostatek při přidání termínu splnění před název úkolu. Pokud bylo datum definované s klíčovým znakem „^", server ho chybně vyhodnotil jako součást názvu úkolu. Bez použití klíčového znaku tento problém nenastával. V obou případech se zároveň termín dokázal přiřadit. Tato chyba je však součástí rozhraní a zpracování serverem služby.

Filtrování úkolů všichni zdárně uskutečnili a byli schopni ho použít. Při označování úkolů jako splněné jsem zaznamenal nejistotu ohledně checkboxu, který tuto aktivitu prováděl. Uživatelé zprvu očekávali pouhé označení úkolu, nikoliv jeho okamžité splnění. Následující interakce již byly v pořádku. Odstra-

ňování úkolů bylo bezproblémové, všichni byli schopni pomocí daného tlačítka všechna zadání provést.

Celý proces všichni zdárně dokončili požadovaným odhlášením účtu z aplikace.

4.5 Zhodnocení výstupního dotazníku

Všichni testeři všechny zadané úkoly dokázali splnit. Jejich dojmy z používání aplikace byly pozitivní. Na otázku ohledně dojmu uváděli:

- „Velmi dobrý."
- „Výborný."
- „Přehledná grafika, barevnost, intuitivní."
- „Výborný."

Vzhled aplikace se jim velmi líbil. Na otázku co se uživatelům na aplikaci líbí se v odpovědích objevily názory:

- „Seznamy jsou přehledné, úkoly jsou dobře uspořádány a rozlišeny."
- „Tmavé pozadí, font, zpracování."
- „Barevné odlišení priorit."
- „Intuitivní ovládání, jednoduchost – přehlednost."

Na otázku co by šlo na aplikaci zlepšit bylo zmíněno:

- „Mohlo by jít vyhledat úkoly z jakéhokoliv seznamu dohromady."
- „Přidat úvodní tutoriál."
- „Doladit grafické zpracování."
- „České menu."

První jmenované by šlo přidat do další verze programu, v původním požadavku se jednalo o filtrování v rámci aktuálních seznamů pro zúžení množiny zobrazených úkolů.

Vytvořit tutoriál je také dobrý nápad, nicméně v současné době podobný princip plní vysouvací nápovědy u prvků. Také jsou k dispozici návody na webu *Remember the milk* samotném. Důležitým faktorem je také to, že uživatel používající *službu* je s její funkcionalitou lépe obeznámen.

Zlepšování grafického zpracování je rozhodně vždy žádoucí. S dalšími verzemi je určitě prostor pro zlepšení či případnou možnost úprav vzhledu.

Ohledně lokalizace do českého jazyka, aplikace je připravena pro zpracování a přeložení. Veškeré texty zobrazené uživateli jsou zobrazovány pomocí metod „i18n()“. Je tedy vše připraveno pro překlady do dalších jazyků.

4.6 Celkové zhodnocení testování použitelnosti

Z poznatků uvedených v sekci 4.4 a po zhodnocení dotazníku v sekci 4.5 jsem byl schopen získat určitou představu o kladech a záporech řešení aplikace. Po zvážení každého poznatku jsem došel k následujícím změnám:

- přidat do textu zobrazovaného nepřihlášenému uživateli nápovědu jakým způsobem se dostane do nastavení,
- zaměnit checkbox za tlačítko s ikonou zaškrtnutí (pro splnění úkolu).

Celkové dojmy testerů mne velmi potěšily a jejich připomínky byly věcné. S pomocí testovací skupiny jsem byl schopen více široce zhodnotit použitelnost aplikace, která byla prokázána díky jejich úsilí.

Závěr

Cílem této práce bylo navrhnout a implementovat klienta ve formě *plasmoidu* pro desktopové prostředí *KDE Plasma 5*. Ten měl splňovat funkčnost trvalého přihlášení, zobrazování, filtrování a přidávání úkolů.

Po rešerši stávajících řešení jsem provedl příslušnou analýzu služby, požadavků a funkcionality. S využitím definovaných požadavků jsem sestavil jednotlivé případy užití a jim odpovídající scénáře. Stěžejní body aplikace jsem zahrnul v doménovém modelu.

Na základě analýzy jsem postoupil k návrhu architektury aplikace a jejího grafického rozhraní. Zhodnotil jsem uskutečnitelnost funkcionalit aplikace pokrytím jednotlivých případů užití návrhy obrazovek.

V realizační části jsem uvedl jakým způsobem se mi dařilo jednotlivé požadavky implementovat, a se kterými problémy jsem se v průběhu realizace musel potýkat.

Následovalo popsání testování použitelnosti uskutečněné testery a zhodnocení jeho výsledků.

Výstupem práce je funkční klient ve formě *plasmoidu* na plochu prostředí *KDE Plasma 5* splňující veškeré požadavky definované zadáním a požadavky uvedenými v analýze.

Aplikace je do budoucna možné rozšířit o další možnou potřebnou funkcionalitu. Je zde prostor pro další možnosti úprav daných úkolů, individuální úpravu vzhledu, notifikace, či překlady do různých jazyků.

Literatura

[1] Stromme, A.: RTM Plasmoid. *V: Chatonka - elektrotechnický blog [online]*, leden 2009, [cit. 2016-12-07]. Dostupné z: `http://blog.chatonka.com/2009/01/rtm-plasmoid/index.html`

[2] Remember the milk: What is Smart Add? *V: Remember the milk - stránka služby [online]*, [cit. 2017-05-01]. Dostupné z: `https://www.rememberthemilk.com/help/?ctx=basics.smartadd.whatis`

[3] Chen, H.: Remember the milk. [online], září 2011, [cit. 2017-05-01]. Dostupné z: `https://blog.hanschen.org/2011/09/05/the-journey-to-a-simple-todo-widget/`

[4] Chen, H.: The journey to a simple todo widget. *V: Hanschen - elektrotechnický blog [online]*, září 2011, [cit. 2016-12-07]. Dostupné z: `https://blog.hanschen.org/2011/09/05/the-journey-to-a-simple-todo-widget/`

[5] Chen, H.: Remember the milk simple. [online], září 2011, [cit. 2017-05-01]. Dostupné z: `https://blog.hanschen.org/2011/09/05/the-journey-to-a-simple-todo-widget/`

[6] Remember the milk: Tour. *V: Remember the milk - stránka služby [online]*, [cit. 2016-12-07]. Dostupné z: `https://www.rememberthemilk.com/tour/`

[7] Remember the milk: Remember The Milk for Linux. *V: Remember the milk - stránka služby [online]*, [cit. 2016-12-07]. Dostupné z: `https://www.rememberthemilk.com/services/linux/`

[8] Remember the milk: API. Build cool stuff that works with Remember The Milk. *V: Remember the milk - stránka služby [online]*, [cit. 2017-05-03]. Dostupné z: `https://www.rememberthemilk.com/services/api/methods.rtm`

[9] Remember the milk: Community API Kits. *V: Remember the milk - stránka služby [online]*, [cit. 2017-05-03]. Dostupné z: https://www.rememberthemilk.com/services/api/kits.rtm

[10] Dimitri van Heesch: Plasma. *V: The KDE developers - stránka služby [online]*, [cit. 2017-05-03]. Dostupné z: https://api.kde.org/frameworks/plasma-framework/html/index.html

[11] Michael Day: rtm-js. *V: Github - stránka služby [online]*, 2013, [cit. 2017-05-03]. Dostupné z: https://github.com/michaelday/rtm-js

[12] Heena: GettingStarted. *V: KDE TechBase - stránka služby [online]*, březen 2017, [cit. 2017-05-03]. Dostupné z: https://techbase.kde.org/Development/Tutorials/Plasma5/QML2/GettingStarted

[13] Remember the milk: Authentication. *V: Remember the milk - stránka služby [online]*, [cit. 2017-05-03]. Dostupné z: https://www.rememberthemilk.com/services/api/authentication.rtm

[14] Joseph Myers: MD5 Algorithm. *V: Myers daily - osobní stránka [online]*, [cit. 2017-05-03]. Dostupné z: http://www.myersdaily.org/joseph/javascript/md5-text.html

[15] Chris Holland: Plasma (Unofficial) API Docs. *V: Github - stránka služby [online]*, [cit. 2017-05-03]. Dostupné z: https://zren.github.io/projects/kde/docs/

[16] Sebastian Tschan: JavaScript MD5. *V: Github - stránka projektu [online]*, [cit. 2017-05-03]. Dostupné z: https://github.com/blueimp/JavaScript-MD5

[17] Remember the milk: rtm.tasks.getList. *V: Remember the milk - stránka služby [online]*, [cit. 2017-05-03]. Dostupné z: https://www.rememberthemilk.com/services/api/methods/rtm.tasks.getList.rtm

[18] Edmundson, D.: Plasmoid Tutorial 1. [online], leden 2015, [cit. 2017-05-01]. Dostupné z: http://blog.davidedmundson.co.uk/node/89

[19] de Hoyos, J. M.: How to write a small plasmoid in QML for KDE desktop. [online], květen 2014, [cit. 2017-05-01]. Dostupné z: http://juanmanueldehoyos.com/how-to-write-a-small-plasmoid-fo/

[20] Jürgen Bocklage-Ryannel, J. T.: *Qt5 Cadaques*. Třetí vydání, 2015. Dostupné z: https://qmlbook.github.io/

Příloha A

Seznam použitých zkratek

API Application programming interface

JSON Javascript object notation

KDE K desktop environment

GTK GIMP toolkit

GIMP GNU image manipulation program

GNU GNU's not unix

MD5 Message-digest algorithm

QML Qt modeling language

Qt Q toolkit

REST Representational state transfer

RTM Remember the milk

UI User interface

Příloha B

Vstupní dotazník

1. V oblasti počítačů a informačních technologií jsem (vyberte jedno):

   Začátečník / běžný uživatel / pokročilý uživatel / expert.

2. Použil/a jste někdy linuxového operačního systému?

   Ano / ne.

3. Použil/a jste někdy desktopového prostředí *KDE Plasma 5*?

   Ano / ne.

4. Používáte nějakou miniaplikaci (widget) na plochu kteréhokoliv operačního systému? Pokud ano, specifikujte který.

   Ano, používám ..... / ne.

5. Využíváte některého z úkolovacích nástrojů? Pokud ano, specifikujte který.

   Ano, používám ..... / ne.

6. Znáte nástroj pro správu úkolů *Remember the milk*?

   Ano / ne.

Příloha C

Testovací scénáře

Tato aplikace slouží jako klient online služby *Remember the milk* určené k organizaci úkolů. Postupně proveďte všechny níže uvedené činnosti.

Přihlaste se ke službě

V rámci aplikace se pokuste přihlásit pomocí testovacích přihlašovacích údajů.

Zobrazte úkoly

Ujistěte se, že následující úkoly patří do vybraných seznamů:

1. úkol „zavolat na úřad“ v seznamu „Personal“,
2. úkol „prodat akcie“ v seznamu „Work“,
3. seznam „Sent“ je prázdný.

Přidejte nový úkol

Postupně vytvořte a přidejte následující typy úkolů. Ujistěte se, že se přidání provedlo dle kritérií. Přidejte úkoly:

1. běžný úkol s libovolným názvem do kteréhokoliv seznamu,
2. úkol s názvem „napsat dopis“ a nejvyšší prioritou do kteréhokoliv seznamu,
3. úkol s názvem „odepsat na email“ do seznamu „Work“,
4. úkol s názvem „nakoupit“ s datem termínu splnění „6.7.2017“ do seznamu „Personal“,
5. úkol s názvem „uklidit“ s přiřazeným místem „doma“.

Vyfiltrujte úkoly

Postupně vyfiltrujte úkoly dle následujících kritérií:

1. v seznamu „Work“ vyfiltrujte úkoly začínající předponou „za“,
2. zrušte vyhledávání popsané v předchozím bodě,
3. pomocí vyhledávání se pokuste najít úkol nazvaný „tajemství“.

Označte úkoly jako splněné

Následující úkoly postupně označte jako splněné:

1. úkol „opravit auto“,
2. úkol „prodat akcie“.

Smažte úkoly

Proveďte smazání těchto úkolů:

1. úkol „odstranit úkoly“,
2. úkol „koupit CD“.

Odhlaste se ze služby

V rámci aplikace se pokuste odhlásit.

Příloha D

Výstupní dotazník

1. Podařilo se Vám splnit veškeré zadané úkoly?

   Ano / ne, nesplnil jsem .....

2. Jaký máte dojem z pracování s aplikací?

   .....

   .....

   .....

   .....

3. Líbí se Vám vzhled aplikace?

   Ano / ne.

4. Co se Vám na aplikaci líbí?

   .....

   .....

   .....

   .....

5. Co by šlo na aplikaci zlepšit?

   .....

   .....

   .....

   .....

Příloha E

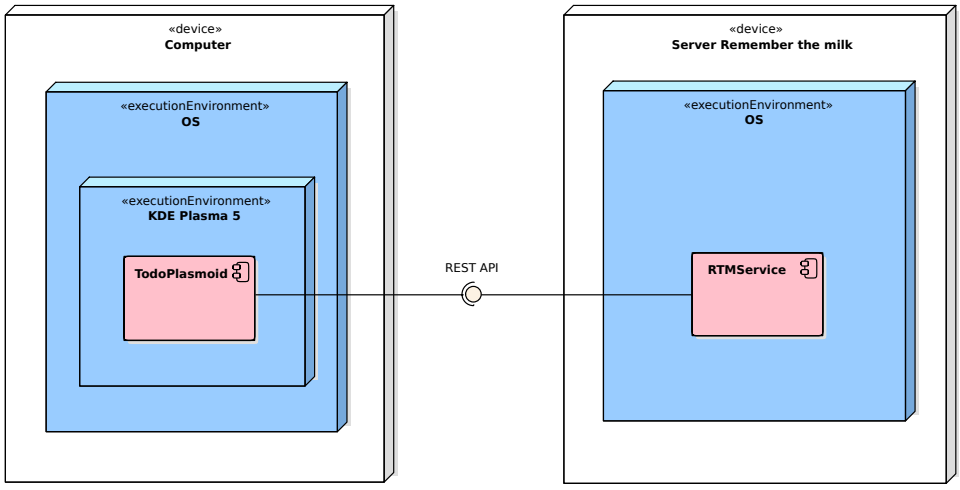
Instalační příručka

E.1 Diagram nasazení

Aplikace je určena k běhu na operačních systémech obsahujících prostředí *KDE Plasma*. Toto prostředí je dostupné na systémech jako například *Linux Mint*, *Kubuntu* či openSUSE. Minimální požadovaná verze je *Plasma 5.8.x*. V diagramu nasazení na obrázku E.1 je možné vidět celkové propojení se serverem služby *Remember the milk*.

E.2 Zajištění prostředí

Je možné, že požadovanou verzi prostředí stávající instalace některých systémů neobsahují. Je ji tak třeba doinstalovat. Pro systémy založené na sys-



Obrázek E.1: Diagram nasazení

tému *Ubuntu* lze použít následující postup pomocí terminálu (čtenář provádí změny svého systému na vlastní riziko). Pro přidání repozitáře obsahujícího požadovanou verzi *plasmy*:

```
$ sudo add-apt-repository ppa:kubuntu-ppa/backports
```

Následně pro provedení aktualizace a upgradu potřebných balíčků:

```
$ sudo apt-get update && sudo apt-get dist-upgrade
```

V případě instalace *plasmy* do systému, který ji neobsahuje, je nutné ještě provést:

```
$ sudo apt-get install kubuntu-desktop
```

Tím by mělo být zaručeno nainstalování aktuální verze *plasmy*.

E.3 Zajištění dependencí

Dále je nutné, aby byly nainstalovány následující požadované dependence:

- Qt verze > 5.2
  - QtQuick > 2.0
  - QtLayouts > 1.1
- org.kde.plasma.components/extras/core/plasmoid > 2.0

Pro instalaci *Qt* doporučuji následovat instrukce dostupné na stránce frameworku: `https://www.qt.io/download/`.

S instalací Qt verze alespoň 5.2 by mělo být zaručeno správných verzí *QtQuick* a *QtLayouts*. Dále komponenty *plasmy* jsou též instalovány, pokud uživatel nainstaluje minimální potřebnou verzi *plasmy*.

E.4 Nasazení aplikace

E.4.1 Instalace

Aplikace je dostupná jako balíček typu *plasma package* s příponou *.plasmoid*. Pro jeho nasazení do systému je nutné provést jeho instalaci. Tu lze zahájit kliknutím myši na plochu prostředí, vybrání položky *Add widgets*. V seznamu *plasmoidů* je na konci položka *Get new widgets*. Po jejím zvolení vyberte *Install widget from local file*. Ve vyskakovacím okně vyberte soubor *todo-plasmoid.plasmoid* z adresáře práce. Tím je aplikace nainstalována.

Také je práce dostupná jako adresář obsahující zdrojový kód. Pro instalaci ze zdrojového kódu proveďte v terminálů následující. Přejděte v terminálu z kořenového adresáře práce do složky *src*:

```
$ cd ./src/impl/
```

Pokračujte instalací pomocí programu *plasmapkg2* s definováním aktuálního adresáře:

```
$ plasmapkg2 -i .
```

Tím je aplikace nainstalována. Pro odinstalaci proveďte analogicky:

```
$ plasmapkg2 -r .
```

Pokud chcete zobrazit *plasmoid* v okně pomocí terminálu, můžete tak učinit pomocí *plasmawindowed* a specifikací názvu nainstalovaného *plasmoidu*:

```
$ plasmawindowed todo-plasmoid
```

E.4.2 Přidání *plasmoidu* na plochu

Po úspěšném nainstalování se aplikace zobrazí v nabídce mezi aktuálními *plasmoidy*. Kliknutím pravého tlačíka myši na plochu je položka *Add widgets*. Po kliknutí na ni se zobrazí panel instalovaných *plasmoidů*. Přetažením požadovaného *plasmoidu* na plochu se na ní vytvoří nová instance. Tu je možné dále používat.

Příloha F

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF