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

Bakalářská práce

Porovnání knihoven pro tvorbu uživatelských rozhraní

Plzeň 2020 Lukáš Moučka

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš MOUČKA**
Osobní číslo:	**A17B0424P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační systémy**
Téma práce:	**Porovnání knihoven pro tvorbu uživatelských rozhraní**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s problematikou uživatelských rozhraní (GUI) a použitelnosti (UX).
2. Seznamte se základní funkcionalitou existujících a používaných knihoven pro tvorbu GUI.
3. Detailně porovnejte vybrané knihovny s ohledem na náročnost jejich použití i s ohledem na tvorbu přístupných a použitelných GUI.
4. Implementujte ukázkovou aplikaci s využitím vybraných knihoven.
5. Využijte ukázkovou aplikaci k porovnání vlastností knihoven.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Richard Lipka, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **7. října 2019**
Termín odevzdání bakalářské práce: **7. května 2020**

Západočeská univerzita v Plzni
L.S.
Fakulta aplikovaných věd

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 15. října 2019

Prohlášení

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

V Plzni dne 7. května 2020

Lukáš Moučka

Abstract

The bachelor's thesis deals with issues of user interfaces (UI) and usability (UX). The theoretical part of the thesis is based on user interfaces analysis and UI types analysis. The usability is processed in the same way. This chapter describes other aspects of UX (psychology, design and copywriting). Then selected GUI libraries and widgets are analyzed in detail. In the second part of thesis is created an application (file manager) in two different libraries (Qt and NCurses). Next chapter describes library properties based on the applications. Both libraries are also evaluated from the perspective of the programmer, UI/UX designer and economist. At the end, the libraries are compered in measurable data and key features.

Abstrakt

Práce se zaměřuje na problematiku uživatelských rozhraní (UI) a použitelnosti (UX). V teoretické části se zabývá rozborem uživatelských rozhraní a jejich typů. Stejným způsobem je zpracována použitelnost a vliv psychologie, designu a copywritingu na ni. Dále jsou vybrané knihovny pro tvorbu GUI a jejich komponenty detailně analyzovány. V praktické části je vytvořena aplikace (souborový manažer) ve dvou odlišných knihovnách (Qt a NCurses). Na základě vzniklých aplikací jsou porovnány jejich technické vlastnosti. Obě knihovny jsou také zhodnoceny z pohledu programátora, UI/UX designera a ekonoma. Nakonec jsou porovnány v měřitelných hodnotách a jejich klíčových vlastnostech.

Poděkování

Tímto bych rád poděkoval vedoucímu práce, panu *Ing. Richardu Lipkovi, Ph.D.* za odborné a důkladné vedení, poskytnuté cenné rady, strávený čas a pomoc při řešení mé bakalářské práce.

Obsah

1 Úvod

Uživatelské rozhraní neboli UI (*User Interface*) je tu s námi již od dob, kdy počítače první generace (např. ENIAC a IBM NORC) byly schopny zabrat celou tělocvičnu.

Jejich rozhraní bylo složité z technického i uživatelského hlediska. Programování probíhalo pomocí sady přepínačů, potenciometrů, propojovacích drátů a za pomocí děrných štítků [52]. Výstup byl zajištěn světelnými signály, tiskem a děrnými štítky. V této době byly frazémy jako „uživatelský prožitek“ nebo „použitelnost uživatelského rozhraní“ ještě neznámé – dnes tuto problematiku zahrnujeme pod zkratku UX (*User Experience*). I na tomto dobovém sálovém počítači si lze představit mnohá úskalí, která vznikala při používání jeho uživatelského rozhraní. Rozhraní bylo neintuitivní a většinou mu rozuměl jen konstruktér a dobře proškolený personál. Pokud již došlo k zadání vstupu do počítače, výstup byl dostupný na druhé straně místnosti ve formě tištěného dokumentu. Validace ani verifikace vstupu nebyla v žádné formě možná, a pokud výsledek neodpovídal předpokladu, bylo nutné celý proces zopakovat znovu.

Vývoj hardwaru šel rychle dopředu a brzy přišly první monitory a ovládací zařízení jako je myš a klávesnice – původní mechanická myš byla vynalezena v roce 1972 ve vývojovém centru *Xerox PARC*. Mohlo by se zdát, že dnešní UI nemá s tím historickým rozhraním nic společného – opak je však pravdou. Většina přepínačů a hardwarových tlačítek všech typů se přesunula na naše monitory. Výstup již nemusí být realizován za pomocí tisku, ale je vykreslen do několika okamžiků na obrazovce počítače. Z pohledu UX již neřešíme problémy jako propojování modulů počítače nebo chození pro výstup do jiné místnosti, ale problémy s rozložením UI řešíme i dnes. Máme řadu graficky vyspělých UI, ale to ještě neznamená, že jsou pro uživatele použitelné a je pomocí nich schopen efektivně ovládat danou aplikaci.

Tato práce by měla poskytnout přehled používaných knihoven pro tvorbu UI a zároveň se dotknout problematiky UX. Z přehledu knihoven budou vybrány ty nejvíce odlišné. Ve vybraných knihovnách bude naprogramována testovací aplikace, která bude hodnocena z několika perspektiv. Z programátorského hlediska bude hodnocena kvalita dokumentace, dostupné komponenty, atd. Z pohledu UX bude hodnoceno chování komponent při zadání neobvyklých vstupů a z ekonomického hlediska bude hodnoceno finanční zatížení vývoje v dané knihovně v porovnání s jinými.

2 Uživatelské rozhraní UI

V obecné rovině lze chápat UI jako množinu jednotlivých komponent, kterými lze ovládat počítačové programy, různá embedded zařízení nebo stroje. S vývojem počítačové techniky paralelně probíhal a stále probíhá vývoj UI. Z tohoto pohledu vznikly tři typy aplikací (chronologicky řazeno): aplikace bez uživatelského rozhraní, aplikace s textovým uživatelským rozhraním a aplikace s grafickým uživatelským rozhraním [52].

2.1 Aplikace bez uživatelského rozhraní

Tato kategorie je tvořena aplikacemi, které nevyžadují přímou interakci s uživatelem. Jednou se spustí a autonomně běží na pozadí operačního systému, kde vykonávají činnost podle vzniklých situací.

Nejpočetnější skupinu tvoří *firmware*, do kterého lze zařadit řídící software různých hardwarových zařízení, který zajišťuje přístup k počítači na nejnižší úrovni. Dále sem patří *služby operačního systému* a *utility* běžící na nízké úrovni (flashování BIOSu (*Basic Input-Output System*), rozdělení disku na oddíly, ...). K ovlivnění běhu aplikace slouží vstupní parametry zadané při startu programu [52].

2.2 Aplikace s textovým uživatelským rozhraním TUI

TUI (*Textual User Interface*) jsou prvním vzniklým typem uživatelského rozhraní a na rozdíl od předchozího typu aplikací podporují interakci s uživatelem. Někdy je tento typ rozhraní označován souhrnně jako konzolové aplikace CLI (*Command-Line Interface*). Jedná se o příkazové interprety a příkazové řádky. Dále sem můžeme zařadit veškeré aplikace, které používají textové řetězce a příkazy k interakci s uživatelem.

Typickým příkladem je operační systém *MS-DOS* nebo *linuxová konzole* [52]. V době vzniku a největší slávy TUI také vznikaly *textové hry* na všechny tehdy dostupné operační systémy. V rámci příkazové řádky byl vypisován příběh a aktuální uživatelův postup. Uživatel mohl ovlivňovat hru jen pomocí předefinovaných textových řetězců. Existuje dokonce archiv českých a slovenských her dostupný na `http://www.textovky.cz`. Textové hry

i dnes mají pár skalních fanoušků a stále probíhá vývoj nových. Stejným způsobem fungují ostatní konzolové aplikace. Jejich fungování a seznam příkazů je popsán v manuálových stránkách. Výstup programu na základě zadaných příkazů je opět vypisován do konzole.

I v rámci konzole lze vytvářet grafický obsah. Za pomocí skládání ASCII[1] znaků je možné vytvářet obrazce. Proces skládání ASCII znaků do obrazců je označován jako *ASCII art* nebo jako *ANSI art* a i dnes má stále své využití v mnoha konzolových aplikacích [18].

2.3 Aplikace s grafickým uživatelským rozhraním GUI

Grafické uživatelské rozhraní neboli GUI (*Graphical User Interface*) není přímým nástupcem TUI. Za spojovací článek mezi TUI a GUI jsou považována *semigrafická uživatelská rozhraní*. Ovládat aplikaci jen skrze příkazovou řádku není moc uživatelsky přívětivé, a tak se postupem času začala řada aplikací přeměňovat v aplikace, které obsahovaly menu a různé kontextové nabídky. Grafického efektu bylo dosaženo pomocí kombinace ASCII znaků [18]. Mezi stále používané programy využívající tuto formu rozhraní patří linuxový *GNU Midnight Commander*, *Memtest86+* nebo BIOS, který je však dnes nahrazovaný UEFI (*Unified Extensible Firmware Interface*).

Nároky uživatelů na rozhraní aplikací se stále stupňovaly, a tak časem nestačilo ani semigrafické uživatelské rozhraní. Zvyšování výkonu osobních počítačů vedlo k myšlence vytvořit plně grafické uživatelské rozhraní. První takové rozhraní vytvořila firma *Xerox* ve vývojovém centru PARC (*Palo Alto Research Center*) pro svůj první počítač v rámci projektu *Xerox Star* [52]. Toto historicky první grafické rozhraní položilo základy dnešních moderních GUI. Bylo tak úspěšné, že jej okopírovala společnost *Microsoft* (*Windows 3.11*, *Windows 95*) i *Apple*. Z ekonomického hlediska však už tak úspěšné nebylo a nikdy nedošlo k jeho rozšíření.

Následně začaly vznikat nové knihovny a celé frameworky pro tvorbu GUI, kterým se primárně věnuje tato bakalářská práce.

[1] Kódová tabulka obsahující znaky anglické abecedy a další znaky používané v informatice.

3 Uživatelský prožitek UX

Uživatelský prožitek neboli UX je složitě přeložitelný pojem do českého jazyka, a tak se můžeme setkat i s překladem *použitelnost uživatelského rozhraní*. Neexistuje ani přesná definice UX, neboť každý z UX designerů do tohoto procesu zahrnuje jiné aktivity (informační struktura, copywriting, ...).

UX designer a autor knihy *UX pro začátečníky*, *Joel Marsh* zastává názor, že uživatelský prožitek je jen vrcholkem ledovce, ale ve skutečnosti jde o proces tvorby prožitku. „UX design (někdy nazývaný UXD) zahrnuje postupy velmi podobné vědeckým postupům: děláte výzkum, abyste pochopili uživatele, rozvíjíte nápady, jak vyřešit potřeby uživatelů – a potřeby firmy – a tato řešení aplikujete a měříte v reálném světě, abyste zjistili, zda fungují.“ [58] Tento autor také rozdělil proces UX do pěti hlavních kategorií a v rámci popisu základních komponent GUI budou zmíněny případy užití UX.

3.1 Psychologie

Správné pochopení myšlení uživatele rozhoduje o úspěchu rozhraní. UX designer musí během své práce často ignorovat své vlastní pocity a myšlenky [58]. V ideálním případě jsou dostupná tvrdá data např. z *A/B testování* nebo z *analýzy chování uživatelů*. A/B testování je také označováno jako *split testování* nebo *bucket testování*. Je to metoda, které porovnává dvě verze aplikace nebo webové stránky a určuje, která z nich je výkonnější nebo lepší pro uživatele [2]. Uživateli je náhodně zobrazena jedna z verzí a na základě statistické analýzy je vyhodnocena nejlépe fungující varianta. U webových aplikací lze využít např. službu *Smartsupp*, ve které můžeme sledovat chování uživatelů, souhrnné reporty a heat mapy[1]. V opačném případě musí designer vycházet ze svých vlastních zkušeností a subjektivních myšlenek.

Designer pracuje s myšlenkami a pocity uživatele. Zjišťuje motivaci uživatele GUI vůbec používat [58]. Měří se čas a počet kroků vedoucích k dokončení dané operace. Pomocí rozhraní uživatel získává návyky, které je schopen reprodukovat i v jiných částech GUI. Sleduje se očekávání uživatele při interakci s nějakou komponentou [58]. Častou chybou designerů je předpoklad,

[1]Heat mapa neboli teplotní mapa zobrazuje pohyb uživatele v rámci aplikace včetně interakcí (např. kliknutí myši).

že si uživatel s daným úkonem bude umět poradit a není mu poskytnuta žádná forma nápovědy. Za další chybu lze považovat, že si designer rozhraní přizpůsobí svým potřebám, ne potřebám uživatele. Společnost *Google* propaguje u návrhu uživatelského rozhraní myšlenku, resp. doporučený způsob návrhu „Mobile First“[2]. Z pohledu UX designera by tato myšlenka měla být přetransformována na „User First“.

3.2 Použitelnost

Pokud *psychologii* chápeme jako podvědomou složku mysli, tak *použitelnost* je čistě vědomá záležitost. V některých případech může být vyžadováno, aby GUI bylo nejednoznačné a matoucí (počítačové hry) [58]. Ve všech ostatních případech je snaha tyto aspekty eliminovat a GUI navrhnout v jeho nejjednodušší formě. Použitelnost je obecně měřitelná, a proto je možné vyloučit subjektivní myšlenky UX designera.

K dosažení nejvyšší míry použitelnosti je potřeba předcházet uživatelským chybám. Uživatele se snažíme zapojovat co nejméně a nepožadujeme od něj nadbytečné úkony [58]. Návrh GUI musí reflektovat podvědomí uživatele a jeho očekávání – komponenty budou umístěny na obvyklých místech, proto např. nevložíme ikonku menu do patičky webu. GUI je navrhováno primárně pro uživatele, funguje v souladu s jeho očekáváním a poskytuje mu potřebné informace [58].

3.3 Design

Z pohledu UX designera není design posuzován z uměleckého hlediska, ale jde primárně o funkčnost GUI [58]. V tomto bodě nastává nejčastěji rozkol mezi UX designerem a klasickým UI designerem. UI designer nahlíží na uživatelské rozhraní jako na umělecké dílo a UX designer jako na funkční celek. V dnešní době jsou tyto dvě profese spojovány do jedné a každý správný designer uživatelského rozhraní by měl mít povědomí o UX.

I z pohledu UX musí do jisté míry GUI vypadat dobře, aby uživatele neodradilo od jeho dalšího používání. Design by měl mít jednotnou podobu. Řeší se, zdali design vede uživatele na správná místa. Zabýváme se otázkami, jestli barvy, tvary a typografie pomáhají lidem najít danou věc a zdali vylepšují použitelnost jednotlivých detailů. Pomocí odlišného designu stejné komponenty je uživatel schopen odlišit jakou akci komponenta vykoná. Např. odlišný design tlačítka určuje, jestli je možné na něj kliknout [58].

[2]Nejdříve návrh UI pro mobilní zařízení a až poté pro zařízení s většími displeji.

3.4 Copywriting

Psaní textů pro UX se liší od klasického copywritingu. V rámci klasického copywritingu píšeme texty propagující společnost nebo značku za účelem zvýšení konverzního poměru. Copywriting pro UX má za cíl dosáhnout výsledku nejjednodušší a nejpřímější cestou [58]. Dříve byl copywriting v rámci UX považován za samostatnou disciplínu, ale poslední dobou se stává nedílnou součástí.

Texty v uživatelském rozhraní musí uživateli jednoznačně říkat, co má uskutečnit za akci. Musí ho motivovat k učinění další akce a informovat o aktuálním stavu. Text psaný největší velikostí fontu je ten nejdůležitější. Každý text musí být jednoduchý, přímý a stručný [58].

3.5 Analýza chování uživatelů

Analýza je pro UX velmi důležitá, ale mezi designery stále podceňovaná [58]. Na počátku je vytvořen prototyp aplikace nebo využit podobný typ aplikace, který je následně analyzován. O tyto analytické výstupy by se měl opírat každý designer. Analýza představuje cyklický proces, proto i aplikace nasazená v produkčním prostředí podléhá neustálému testování. To poskytuje zpětnou vazbu, na základě které dochází k úpravám již hotového GUI.

Analýza dále poskytuje data o chování uživatelů a vyvrací subjektivní názory designerů [58]. Pomocí shromážděných dat jsou designeři schopni řešit vzniklé problémy nebo pracovat na vylepšeních. V rámci analýzy se měří efektivita práce uživatelů s rozhraním. Shromážděná data slouží i pro marketing, např. při snaze o zvýšení konverzního poměru na e-shopu.

4 Knihovny pro tvorbu UI

Existuje mnoho knihoven pro tvorbu UI a největší rozdíly lze nalézt u *podporovaných operačních systémů*, *komunity kolem knihovny*, *licenčních podmínek* a *podporovaných programovacích jazyků*.

Detailně popsané knihovny byly vybrány na základě jejich popularity mezi programátory v rámci diskuzních fór na `https://stackoverflow.com` nebo na `https://www.quora.com`. Dalším kritériem pro výběr byly statistiky zveřejněné na slant.co, analyticsindiamag.com nebo na dev.to. Výběr knihoven pro tvorbu uživatelského prostředí také do jisté míry reflektují vazby populárních *Python frameworků*, protože programovací jazyk Python je v době psaní bakalářské práce na vzestupu podle *TIOBE indexu*.[1]

4.1 Qt

Qt je určeno pro vývoj aplikací napříč operačními systémy pro stolní počítače, embedded a mobilní zařízení. Jedná se o knihovnu napsanou v programovacím jazyce *C++*. Používá preprocesor MOC[2], a tím rozšiřuje programovací jazyk o další funkce. Preprocesor musí před každou kompilací analyzovat zdrojové soubory napsané v Qt (rozšířené C++) a vygenerovat standardní zdrojové soubory kompatibilní s C++. Po dokončení práce preprocesoru je možné vygenerované zdrojové soubory zkompilovat standardními známými kompilovacími nástroji pro C++ (*Clang*, *GCC*, *ICC*, *MinGW* a *MSVC*) [40].

Aktuální verze knihovny v době psaní bakalářské práce je 5.13 a této verzi je věnován i následující text.

4.1.1 Podporované operační systémy

Mezi podporované operační systémy patří *Windows*, *Linux*, *OS X*, *VxWorks*, *QNX*, *Android*, *iOS*, *BlackBerry*, *Sailfish OS* a další [40].

4.1.2 Komunita

Za dobu své existence od roku 1990 se kolem knihovny, resp. společnosti jménem *The Qt Company* vytvořila poměrně velká komunita, která je slo-

[1] Vyjadřuje popularitu programovacích jazyků na základě vstupů do 25 internetových vyhledávačů.

[2] Meta-object Compiler – doc.qt.io/qt-5/moc.html

žena z mnoha společností a jednotlivců. Každý může do knihovny přispět napsáním kódu nebo dokumentace. Podílet se může udržováním wiki stránek dostupných na `https://wiki.qt.io/`. Tuto komunitu zaštiťuje aliance *The Qt Project*, která je v současné době větší než samotná společnost. Společnost je ale stále hlavním iniciátorem ve vývoji knihovny [40].

Existuje oficiální fórum. Obsahuje vlákna o vývoji knihovny, vzdělání a certifikaci a samotná vlákna uživatelů. Fórum je dostupné na `https://forum.qt.io`.

4.1.3 Licence

Qt je dostupné pod mnoha licencemi. Společnost prodává komerční licence, ale je možné knihovnu získat bezplatně pod licencemi GPL (*General Public License*) a LGPL (*Library General Public License*) [40]. Od verze 5.7 jsou dostupné tyto licence:

- komerční licence
- LGPL3 open source licence
- GPL2 nebo GPLv3 open source licence [43].

4.1.4 Podporované jazyky

Popis uživatelského prostředí probíhá pomocí deklarativního jazyku QML (*Qt Modeling Language*) založeného na *JavaScriptu*. Spolu se znalostí HTML (*Hypertext Markup Language*) a CSS (*Cascading Style Sheets*) je možné kódovat celé aplikace. Deklarativní jazyk spolu s frameworkem *Qt Quick* slouží pro produkci výsledného GUI.

Pro vazby na GUI je možné využít těchto programovacích jazyků: *Python*, *Ring*, *Go*, *Rust*, *PHP* a *Java*. Jediná podpora pro Python je oficiálně udržovaná pod názvem projektu *Qt for Python* [16].

4.2 JavaFX (OpenJFX)

JavaFX vznikla jako náhrada za knihovnu *Swing*. Jedná se o standardní knihovnu pro tvorbu grafického uživatelského prostředí pro *Java SE*. Od verze Javy 8 je součástí JRE (*Java Runtime Environment*) / JDK (*Java Development Kit*). Součástí JRE/JDK zůstala knihovna do verze Javy 10. Java 11 přišla se zásadní změnou, kdy se v roce 2018 JavaFX stává součástí open

source knihovny *OpenJDK* v rámci projektu *OpenJFX*. Knihovna je určena pro vývoj aplikací určených pro stolní počítače a embedded zařízení [31].

Dále se budeme zabývat jen open source knihovnou OpenJDK ve verzi 13.

4.2.1 Podporované operační systémy

Knihovna je multiplatformní díky tomu, že je napsaná v programovacím jazyce *Java*. Java je interpretovaný jazyk, a proto nedochází k vytváření *strojového kódu* ale *bytecodu*. Tento formát je nezávislý na operačním systému a architektuře daného zařízení, protože je interpretovaný pomocí JVM[3].

Podporované operační systémy lze rozdělit na dvě skupiny. První skupinou jsou podporované společností *Oracle*. Jedná se o *Windows*, *Mac OS X*, *Solaris*, *Oracle Enterprise Linux* [28]. U této skupiny lze očekávat plnou podporu a rychlé řešení reportovaných chyb.

Druhá skupina již není podporovaná Oraclem. Pro operační systémy v této skupině lze normálně vyvíjet aplikace, ale není žádná garance bezchybného sestavení [28]. Velkou roli při řešení reportovaných problému má komunita kolem knihovny, která vyvíjí úsilí, aby se počet podporovaných platforem rozšiřoval. V současné době jsou podporovány tyto operační systémy: *IBM AIX*, *Microsoft Windows Server*, *Mac OS X* a *Suse SLES* [28].

4.2.2 Komunita

JavaFX je využívána pro pedagogické účely v rámci středních a vysokých škol [59]. V komerčním prostředí má jen marginální využití a nebo je průběžně nahrazována jinými knihovnami.

Kontinuálně je udržována wiki *OpenJDKWiki* dostupná na `https://wiki.openjdk.java.net`, ale žádné diskuze zde nejsou. Komunita však přispívá do samotné knihovny. O dění kolem knihovny informují blogy `https://jonathangiles.net` a `http://fxexperience.com`, jejichž společným autorem je *Jonathan Giles*, který byl dříve technickým vedoucím v *JavaFX týmu* ve společnostech *Sun Microsystems* a *Oracle Corp.* Autor *Pedro Duque Vieira* přispívá do komunity vývojem open source knihoven, které rozšiřují použitelnost samotné knihovny. Mezi jeho nejznámější projekty patří: *JXScene*, *JMetro* a *FXRibbon*.

[3]Java Virtual Machine – `https://docs.oracle.com/javase/8/docs/technotes/guides/vm/index.html`

4.2.3 Licence

Jedná se o svobodný software, který je pod licencí GPLv2 [26].

4.2.4 Podporované jazyky

Jediným podporovaným programovacím jazykem je *Java*. Pomocí deklarativního jazyku FXML založeného na značkovacím jazyce XML (*Extensible Markup Language*) je možné definovat GUI. Poskytuje potřebné struktury pro jeho vytváření a díky tomu je možné oddělit aplikační logiku od samotného rozhraní. FXML kód není nutné psát, protože existuje nástroj *Java Scene Builder*, který kód generuje – uživatel stylem drag&drop umístí komponenty na pracovní plátno, upraví jejich styl a vlastnosti.

4.3 wxWidgets

Knihovna byla založena již v roce 1992. Je napsaná v *C++*, ale je možné použít celou řadu programovacích jazyků pro psaní samotného kódu. Používá nativní ovládací prvky a nástroje daného operačního systému. Výsledné GUI má stejnou podobu jako nativní aplikace operačního systému [17].

Aktuální verze knihovny v době psaní bakalářské práce je 3.1.2 [17].

4.3.1 Podporované operační systémy

Knihovna je multiplatformní. Při psaní kódu stačí využít správnou knihovnu a kompilátor pro využívanou platformu [17].

Podporované platformy:

- wxGTK – doporučeno pro *Linux* a *unixové operační systémy*, které používají *GTK+* ve verzi 2.6 a vyšší
- wxMSW – Windows v 64bitové i 32bitové verzi ve variantách *Windows XP*, *Vista*, *7*, *8* a *10*
- wxOSX/Cocoa – aplikace založené na frameworku *Cocoa* na *Mac OS X*
- wxQt – využití pro knihovnu *Qt* (vyžaduje verzi 5 nebo vyšší)
- wxX11 – *Linux* a *Unix* využívající grafické prostředí *X11*
- wxMotif – *Linux* a *Unix* využívající set widgetů *OpenMotif* nebo *Lesstif* [17].

4.3.2 Komunita

Existuje komunitní wiki dostupná na `https://wiki.wxwidgets.org`. Obsahuje obecné informace o knihovně, nástroje, pomoc a podporu, příručky a tutoriály, dokumentaci a informace o vývoji. Wiki odkazuje na oficiální fórum dostupné na `https://forums.wxwidgets.org`. Fórum má velký počet členů, kteří ho udržují aktivní.

4.3.3 Licence

Aktuální licence nese název *wxWindows* – stále se čeká na schválení názvu *wxWidgets*. Jedná se o licenci LGPL. Důvod vlastní licence je ten, že uživatel smí určit podmínky distribuce svého softwaru. Hlavním důvodem této výjimky je vývoj proprietárního softwaru [38].

4.3.4 Podporované jazyky

Hlavním podporovaným jazykem je *C++*, ve kterém je napsána i samotná knihovna. Dále je možné využívat programovací jazyk *Python*, *Perl* a *C*.

4.4 GTK

GTK neboli *GIMP Toolkit* je multiplatformní knihovnou pro tvorbu GUI. Obsahuje kompletní sadu widgetů a je vhodná pro menší projekty i kompletní sady aplikací [10]. Knihovna je využívána pro vývoj desktopového linuxového prostředí *GNOME*. Je v ní napsána i řada známých a hojně využívaných aplikací. Mezi nejznámější patří grafický bitmapový editor *Gimp* nebo vektorový grafický editor *Inkscape* [33].

4.4.1 Podporované operační systémy

Primárně je určena pro okenní systémy založené na *X11* a *Waylandu* běžící pod *Linuxem*. Knihovna funguje i pod operačním systémem *Windows* a *Mac OS X* [29].

4.4.2 Komunita

Kolem knihovny neexistuje žádné oficiální diskuzní fórum. Podporu zajišťuje několik společností, které poskytují své zkušenosti a dovednosti. V rámci edukace nabízí různá školení, konzultace a vývoj aplikací. V rámci jejich spektra služeb je nabízena i hardwarová integrace [44].

4.4.3 Licence

Knihovna je svobodným softwarem jako součást projektu GNU (*GNU's Not Unix*). Licenční podmínky GNU GPL umožňují vývojářům použití této knihovny pro vývoj proprietárního softwaru bez jakýchkoli poplatků.

4.4.4 Podporované jazyky

Pro vazby na GUI je možné použití mnoha programovacích jazyků. Od verze 3 je podporovaným jazykem *C++*, *D*, *FreeBasic*, *Haskell*, *Java*, *JavaScript*, *Lua*, *Pascal*, *Pascal*, *Python*, *Ruby*, *Rust* a *Valda*. Mezi částečně podporované patří *Ada*, *Fortran* a *Go* [3].

4.5 Tcl/Tk

Tcl/Tk je widget toolkit pro vytváření okenních aplikací. Knihovna je napsána v programovacím jazyce *C* a je určena pro vývoj desktopových aplikací. Tcl/Tk nám umožňuje umisťovat jednotlivé prvky komponenty do obdélníkové oblasti, kde definujeme jejich rozvržení. Geometrie komponentů je dynamická, takže rozvržení okenní aplikace může reagovat na změny v rozvržení nebo změnu zobrazení [27].

Aktuální verze Tcl/Tk v době psaní bakalářské práce je 8.6.9 [15].

4.5.1 Podporované operační systémy

Knihovna poskytuje rozhraní pro operační systém *MacOS* a všechny verze *Microsoft Windows* počínaje verzí *95*. Na ostatních platformách, kde je knihovna dostupná vypadá výsledné GUI podobně jako výsledek knihovny *Motif* – nepoužívá však tuto knihovnu [27].

4.5.2 Komunita

Komunita kolem knihovny je poměrně aktivní. Existuje udržovaná wiki dostupná na `https://wiki.tcl-lang.org/`. Neexistuje žádné oficiální fórum, ale existuje desktopová aplikace *TkChat* [30], pomocí které je možné komunikovat. Pro komunikaci a sdílení znalostí bylo možné využít oficiální kanál na *Slacku*, ale ten již není aktivní.

4.5.3 Licence

Knihovna je open source projektem. Licence je typu BSD (*Berkeley Software Distribution*) – je mnohem méně omezující než licence GPL. Podle definice licencí od *Fedora Project* je kompatibilní i s licencemi GPL ve verzi 2 a 3 [50].

4.5.4 Podporované jazyky

Nativním jazykem knihovny je *Tcl*. Dále je možné vyvíjet pomocí *Ruby*, *Perlu* a *Pythonu* [27].

4.6 Ultimate++

Jedná se o multiplatformní knihovnu určenou pro rychlý vývoj aplikací se zaměřením na produktivitu programátorů. Rychlého vývoje je dosaženo inteligentním používáním *C++*. Kromě knihovny pro vývoj GUI obsahuje také knihovnu pro SQL (*Structured Query Language*). Ultimate++ je napsáno v C++. Knihovna integruje i vývojové prostředí *TheIDE*, které obsahuje technologii *BLITZ-build*. Tato technologie výrazně zrychluje sestavení programu. IDE (*Integrated Development Environment*) může spolupracovat s těmito kompilovacími nástroji: *GCC*, *Clang*, *MinGW* a *Visual C++* [25].

Na vývoji se podílí i několik českých vývojářů. Mezi hlavní vývojáře patří *Mirek Fídler* a *Tomáš Rylek* – jsou i hlavními členy *Ultimate++ teamu* [5].

Aktuální verze je v době psaní bakalářské práce vydaná v roce 2019 (rev. 13068) [25].

4.6.1 Podporované operační systémy

Mezi plně podporované operační systémy patří *Microsoft Windows*, *GNU/-Linux*, *FreeBSD* a *Mac OS*. Knihovna funguje jako wrapper na *WindowsAPI*, *X11*, *GTK* a *Cocoa* [34].

4.6.2 Komunita

Existuje oficiální aktivní fórum dostupné na `https://www.ultimatepp.org/forums/`.

4.6.3 Licence

Ultimate++ je open source projekt dostupný pod BSD licencí [25].

4.6.4 Podporované jazyky

Jediným podporovaným jazykem je *C++*.

4.7 Unity

Vývojová platforma pro vývoj 2D a 3D aplikací. Poskytuje vše potřebné pro vývoj, provoz a následné zpeněžení. V Unity se vyvíjí aplikace pro herní, automobilový, dopravní a výrobní průmysl. Dále slouží ve filmovém průmyslu pro tvorbu animací. Své zastoupení má v architektuře, inženýrství a v návrhu konstrukcí [35]. Z výčtu těchto zaměření lze konstatovat, že knihovna je primárně určena pro grafické aplikace, než pro tvorbu okenních aplikací s formuláři – lze ji však použít i pro tento účel. Unity obsahuje editor dostupný pro *Windows*, *Mac* a *Linux*. Slouží k návrhu herních světů včetně herní logiky. Má implementovaný režim *Play* pro rychlý náhled dosud odvedené práce v reálném čase [14].

Aktuální verze Unity v době psaní bakalářské práce je 5.3.4 [41].

4.7.1 Podporované operační systémy

Aktuálně je podporovaných více než 25 platforem napříč mobilními, stolními, konzolovými, televizními a webovými. Podpora nechybí ani pro VR (*Virtual reality*) a AR (*Augmented reality*) platformy [14].

4.7.2 Komunita

Existuje oficiální fórum dostupné na `https://forum.unity.com`, které slouží jako hlavní místo pro komunitní diskuze a chatování. Dále existuje komunitní server pro pokládání otázek a jejich zodpovídání na `https://answers.unity.com`. Možné je i spojení s ověřeným odborníkem při řešení našeho problému s daným projektem [47].

4.7.3 Licence

Jedná se o proprietární software, který je volně dostupný a použitelný. Licence *Unity Personal* je určena pro jednotlivce a malé organizace s příjmy nižšími než 100 000 $ za posledních 12 měsíců [49]. Po překročení příjmů je nutný výběr vhodného placeného plánu pro danou organizaci.

4.7.4 Podporované jazyky

Unity v současné době podporuje jen *C#* [6].

4.8 NCurses

NCurses se bez hlubší analýzy jeví jako knihovna pro tvorbu uživatelského textového rozhraní neboli TUI. Umožňuje však vytvářet další instance oken o různých velikostech – díky této vlastnosti lze knihovnu považovat za hybrid mezi TUI a GUI, i když z hlediska použití má blíže ke GUI. Jedná se o novou verzi starší knihovny *curses*, a proto název NCurses (*new curses*). Obsahuje velkou sadu funkcí známých z běžných knihoven. Umožňuje práci se vstupními periferiemi (myš a klávesnice), změnu barev, kreslení po obrazovce a vytváření tvarů. Jak již bylo zmíněno, umožňuje vytvářet další instance oken. Každá z těchto instancí si alokuje vlastní paměť, a proto svým obsahem neovlivní obsah jiného okna [21]. NCurses je součástí projektu GNU a emuluje chování knihovny z unixového systému s názvem *System Release 4.0* [54]. Je napsána v programovací jazyce *C*.

Aktuální verze knihovny v době psaní bakalářské práce je ve verzi 6.1 a byla vydána 27. 1. 2018 [54].

4.8.1 Podporované operační systémy

Od roku 1995 bylo NCurses migrováno do mnoha operačních systémů. Mezi podporované operační systémy lze zařadit všechny systémy založené na *linuxovém jádře* vyjma některých embedded zařízení. Mezi další patří *OpenBSD*, *FreeBSD* a *OSX*. Používá se také na všech *unixových systémech*. Mezi takové operační systémy patří *AIX*, *HP-UX*, *IRIX64*, *SCO*, *Solaris*, *Tru64* [54].

4.8.2 Komunita

Neexistuje žádné oficiální fórum. NCurses je však dále používáno pro vývoj známých nástrojů spadajících pod projekt GNU. Mezi hlavní patří správci souborů *Midnigh Commander* a *Vifm*. Dále je knihovna použita u známého textového editoru *Vim* nebo hudebních aplikací jako je equalizér od *PulseAudio* nebo hudební přehrávač *Mp3blaster* [21].

4.8.3 Licence

Jedná se o svobodný software, který je součástí projektu GNU a má svou vlastní licenci [53].

4.8.4 Podporované jazyky

Hlavním podporovaným programovacím jazykem je *C*, ale v průběhu let vznikly i vazby na mnoho dalších. Mezi ty známější programovací jazyky patří *Python*, *Java*, *JavaScript*, *Perl* a *PHP* [54].

4.9 Newt

Newt je knihovna určena pro tvorbu barevného TUI založených na widgetech. Mezi hlavní patří přepínače, zaškrtávací políčka, štítky, posuvníky a vstupní pole. Má základ na knihovně *S-lang* [57]. Ze vstupních periférií podporuje jen klávesnici.

Autor knihovny *Miroslav Lichvar* ji původně navrhl pro instalaci kódu *Red Hat Linux* se zaměřením na jednoduché rozhraní a ovládání. Knihovna je napsána v programovacím jazyce *C* [48].

Aktuální verze knihovny v době psaní bakalářské práce je 0.52.21 a byla vydána před 6 měsíci [8].

4.9.1 Podporované operační systémy

Jediným podporovaným operačním systémem je *Linux* [48].

4.9.2 Komunita

Žádné oficiální fórum ani jiné diskuze kolem knihovny Newt neexistují. Vlastnosti knihovny jsou optimální pro proces instalace. Kromě instalátorů knihovnu používá např. program *Partimage* [48].

4.9.3 Licence

Knihovna je svobodným softwarem spadajícím pod projekt GNU. Publikována je pod licencí GPLv2 [57].

4.9.4 Podporované jazyky

Hlavním programovacím jazykem je *C* [48].

5 Další knihovny

5.1 V C/C++

1. **Cegui** – poslední verze knihovny 0.8.7 byla vydána 28. dubna 2016 (Vývoj již nejspíše skončil.). Multiplatformní knihovna je zaměřena na vývojáře her. Je šířena pod licencí MIT. Obsahuje WYSIWYG[1] editor pro vytváření rozvržení a umístění grafických objektů [32].

2. **FLTK** – multiplatformní sada nástrojů s podporou 3D grafiky přes *OpenGL*. Knihovna je navržena modulárně s důrazem na její celkovou velikost. Obsahuje editor uživatelského prostředí s názvem *FLUID*. Jedná se o open source knihovnu pod licencí GPLv2 [37].

3. **JUCE** – knihovna určená primárně pro vývoj hudebních aplikací. Úprava uživatelského rozhraní probíhá pomocí kódovacího modulu *Projucer* a může integrovat *OpenGL*. Je šířena pod volnou licencí GPLv3 (edukativní účely) a komerční licencí [45].

4. **Nana** – multiplatformní knihovna pro vytváření uživatelských rozhraní. Aktuálně podporuje platformy *Windows*, *Linux* (pouze zobrazovací server *X11*) a experimentálně *Mac OS*. Knihovna je open source a šířena pod licencí BSL (*Boost Software License*) [1].

5.2 V ostatních jazycích

1. **Swing** – knihovna napsaná v programovacích jazyce *Java* je předchůdcem knihovny *JavaFX*. Nyní již není vyvíjena, ale stále je používána u dříve vzniklých projektů – pro nově vzniklé projekty by se již používat neměla. Využití má nyní jen pro edukativní účely.

2. **SWT** – knihovna Standard Widget Toolkit je napsána v programovacím jazyce *Java*, která v současné době už slouží jen pro vývoj vývojového prostředí *Eclipse IDE*.

3. **GTK#** – sada nástrojů pro GUI patřící open source projektu *Mono*, který je vlastněn *Xamarinem*. Funguje jako wrapper nad *GTK* a ve spojení s knihovnami *GNOME* je možné vyvíjet plně graficky nativní

[1] „What you see is what you get.“

aplikace pro toto grafické prostředí. Hlavním používaným programovacím jazykem je *C#*, ale je možné použít pro vývoj i jiné programovací jazyky [20].

4. **Kivy**, **PyQT**, **Tkinter**, **WxPython** – populární používané *Python frameworky* pro tvorbu uživatelského prostředí. Většina z nich tvoří wrapper kolem některé z používaných knihoven pro tvorbu GUI.

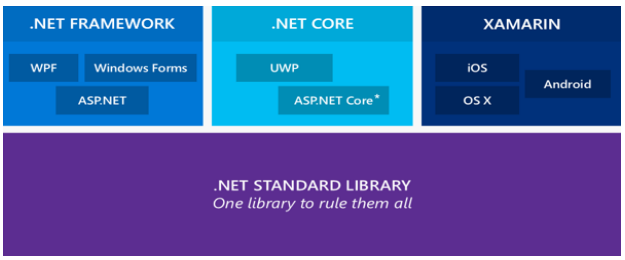
5.3 Windows API

Windows API (Win32 API nebo WinAPI) je rozhraní vyvinuté pro operační systém *Windows*. Aplikace napsaná v libovolném programovacím jazyce musí v systému komunikovat prostřednictvím Windows API [56]. Jedná se o původní platformu pro aplikace napsané v *C/C++*, které vyžadují přímý přístup k *Windows* a *hardwaru* [19].

API (*Application Programming Interface*) lze rozdělit do 8 hlavních kategorií. Jedna z těchto kategorií se týká uživatelského rozhraní. Poskytuje funkce pro vývoj okenních aplikací včetně všech komponent. Zároveň zpracovává vstup z periferií počítače. S touto kategorií úzce souvisí GDI (*Graphics Device Interface*) poskytující funkce pro výstup grafického obsahu na monitory, tiskárny a další výstupní periferní zařízení [56].

5.3.1 Knihovny založené na .NET Frameworku

1. Windows Forms (WinForms)
2. WPF (*Windows Presentation Foundation*)
3. UWP (*Universal Windows Platform*)
4. Xamarin



Obrázek 5.1: Schématické znázornění knihoven v .NET Frameworku [56]

5.3.2 Windows Forms (WinForms)

WinForms tvoří wrapper kolem *Windows API*. Slouží k vývoji okenních aplikací pro operační systém *Windows*. Komponenty do uživatelského rozhraní se pozicují absolutně do pixelové mřížky a mají pevně danou velikost. Tento fakt působí problémy na zařízeních s jiným zobrazením – problém začal nastávat hlavně s příchodem mobilních zařízení. Dále má problémy s upscalingem[2], kdy nelze používat stejnou aplikace na mobilním telefonu, tabletu nebo stolním počítači z důvodu jiného rozlišení – na obrazovkách s vyšším rozlišením má aplikace nižší DPI (*Dots Per Inch*). Z tohoto důvodu WPF zavádí jako jednotku délky DIP (*Device Independent Pixel*) a jen vektorovou grafiku. Dalším omezením je pomalé vykreslovací rozhraní GDI. S příchodem *.NET 2.0* došlo k rozsáhlé aktualizaci a zároveň k částečnému nahrazování za *WPF* [12].

Používat WinForms u nově vznikajících aplikací není již vhodné – používá se v již existujících aplikacích. Pro svou jednoduchost může být použit pro edukativní účely.

5.4 Windows Presentation Foundation (WPF)

WPF vzniklo jako náhrada za *WinForms*. Je součástí *.NET Frameworku*, kde je obsaženo *ASP.NET* nebo předchozí knihovna WinForms pro případ, že je vhodnější pro daný účel použití této knihovny. Pro deklaraci GUI je přítomen značkovací jazyk XAML(*Extensible Application Markup Language*) [11]. Využívá se pro vývoj desktopových aplikací. Migrace z tohoto frameworku je méně obtížná než z WinForms [19].

5.4.1 Podporované operační systémy

Aplikace vyvíjené pomocí této knihovny jsou spustitelné jen pod operačním systémem *Windows*. V každé verzi Windows musí být nainstalován *.NET Framework* ve správné verzi [4].

5.4.2 Komunita

Komunita používá oficiální *WPF forum* dostupné na `https://social.msdn.microsoft.com/Forums/en-US/home?forum=wpf` pro vzájemnou diskuzi a řešení problémů.

[2]Přepočet z nižšího rozlišení na rozlišení vyšší.

5.4.3 Podporované jazyky

Pro vývoj je možné použít programovací jazyk *C#* nebo *Visual Basic* [42].

5.4.4 Universal Windows Platform (UWP)

Univerzální platforma vznikla pro vývoj aplikací nezávislých na platformě. Je určena pro *Windows 10*, *Windows 10 Mobile*, *Xbox One*, *Hololens* a *Surface Hub* [12]. Vývoj *Windows Phone* byl ukončen a nyní je *Xamarin* jedinou platformou pro vývoj mobilních aplikací. Dalším problémem je nekompatibilita se staršími operačními systémy Windows – verze Windows 10 je jediná kompatibilní.

5.5 Aqua

Aqua je GUI pro operační systém *Mac OS* společnosti *Apple* a prvně bylo využito v roce 2001. Pod stejným názvem se skrývá i knihovna pro vývoj aplikací do tohoto prostředí. Název GUI koresponduje se vzhledem rozhraní, protože je celé založeno na vodní tématice. Tvar kapky se odráží v ovládacích prvcích rozhraní a průhlednost např. v oknech. Za pomocí pokročilého použití barev, animací, efektů (např. odraz) a funkcí orientovaných na uživatele bylo vytvořeno přitažlivé prostředí. Aqua je objektově orientované prostředí, které je snadno programovatelné – stejnou vlastnost mají i starší Java knihovny jako je *Swing* nebo *AWT*. Stejně jako konkurenční GUI (např. Windows) je zaměřeno na uživatele a veškerá funkčnost je podřízena v jeho prospěch. Rozhraní bylo tehdy nejsnadněji použitelným rozhraním, a proto bylo implementováno i do unixových systémů – *Mac OS X* také vychází z Unixu. [55]. Postupem času s vydáváním nových verzí operačního systému docházelo ke grafickým modifikacím, ale většina vlastností původního GUI je implementována i v aktuální verzi (Např. tvar kapky vody je stále zachován pro ovládací prvky okna). V době psaní bakalářské práce je Aqua stále součástí nejnovější verze operačního systému s názvem *Catalina*.

Rozhraní bylo opravdu na vysoké úrovni a postupem času se začaly konkurenční operační systémy inspirovat vzhledem a funkčností. V roce 2007 společnost *Microsoft* představila GUI s názvem *Windows Aero*, které také využívá např. průhlednost, podobné efekty oken, animace apod.

6 Prvky uživatelského rozhraní

Prvků uživatelského rozhraní neboli UI existuje mnoho a řada knihoven přidává některé své proprietární. V současné době se však objevují rozdíly mezi komponentami pro mobilní a stolní zařízení a s tím odpovídající problémy. Např. přenosem mobilní aplikace do desktopového prostředí mohou vzniknout vizuální problémy, ale také problémy týkající se UX, protože dochází ke změně rozložení widgetů. U většiny nově vznikajících aplikací je tlak na responzivní design a společnost *Google* propaguje myšlenku „Mobile First“, která apeluje na designery, aby začínali s návrhem pro nejmenší displeje – nedochází k problémům při transformaci z vysokého rozlišení na nízké. Podíl mobilních zařízení na trhu roste a stále více uživatelů jej používá k pracovním účelům, ale i ke každodenní konzumaci webového obsahu. S tímto faktem se pojí i minimální rozdíl komponent pro dotyková a nedotyková zařízení. Díky těmto aspektům se většina knihoven snaží být *multiplatformní*.

Při návrhu uživatelského rozhraní je potřeba být konzistentní a snažit se předvídat chování uživatelů. Uživatelé se s prvky seznámili určitým způsobem, takže jejich správný výběr vede ke zvýšení efektivity a uspokojení při práci s daným uživatelským rozhraním [36].

Ovládací prvek uživatelského rozhraní neboli widget je analogií k hardwarovým zařízením (myš, klávesnice, joystick, ...) nebo k dříve používaným elektronickým součástkám (spínač, žárovka, potenciometr, ...). Vývoj ovládacích prvků probíhal paralelně s vývojem počítačů.

6.1 Button (tlačítko)

Tlačítko reaguje na stisk vyvoláním události. Typicky má svůj popis, který může být doplněn o piktogram [36]. V jistých situacích může být použit jen piktogram. Tvar komponenty není unifikovaný, ale většinou má obdélníkový nebo čtvercový tvar. Tvarem i vzhledem se snaží přiblížit hardwarovému spínači.

Každá komponenta slouží jako prostředek pro interakci uživatele s GUI. Tlačítko je elementární komponentou, ale z pohledu UX je velmi významné. Podle významu v daném kontextu se tlačítka dělí na *primární* a *sekundární*.

6.1.1 Primární tlačítko

Stisk tohoto typu tlačítka vyvolává akce, které pomáhají našim cílům. Primárním tlačítkům jsou přiřazovány akce jako např. registrace, nákup zboží, nahrání obsahu, posílání a sdílení obsahu. Stiskem dochází k vytváření dříve neexistujícího obsahu, a proto chceme, aby uživatelé akci prováděli co nejčastěji [58].

Snaha UX designera je umístit tlačítka na viditelné místo. Tlačítko musí mít velký kontrast vůči okolnímu obsahu a jeho umístění neboli pozice v layoutu musí být na ose interakce nebo alespoň blízko ni, aby ho uživatelé zahlédli prioritně [58].

6.1.2 Sekundární tlačítko

Má opačný význam než primární tlačítko a vyvolává akce opačného charakteru. Obecně lze říci, že ukončuje nebo ruší akce vyvolané primárním tlačítkem. Např. při zobrazení modálního okna pro vytvoření nového adresáře jsou k dispozici dvě tlačítka. Primární tlačítko `OK` vytváří nový adresář a naopak sekundární tlačítko `Cancel` ukončuje modální okno a nedochází k vytvoření nového adresáře.

Pokud je nežádoucí, aby uživatel akci vyvolanou primárním tlačítkem zrušil, mohou být sekundární tlačítka méně viditelná a umístěna v layoutu mimo osu interakce. Uživatel tlačítko nepoužije intuitivně, ale pouze v případě, že ho bude hledat. Může však nastat situace, že chceme, aby tlačítko mělo primární styl, ale sekundární pozici. Takový typ je označován jako *speciální tlačítko* a zpravidla má varovnou barvu (červená nebo oranžová). Vyjadřuje závažnost situace a nutí uživatele přemýšlet nad danou situací (zrušení uživatelského účtu, zrušení předplatného služby, ...) [58].

6.2 Check box (zaškrtávací pole)

Zaškrtávací pole dává uživateli možnost, zdali jej stiskem zaškrtne. Vedle pole se nachází popis možnosti. Vyhodnocení pole obvykle probíhá až při potvrzení. Pole má obvykle čtvercový tvar. Analogii nenalezneme v podobě hardwarového tlačítka, ale můžeme jej objevit ve formulářích v papírové podobě (daňové přiznání, sčítání obyvatel, ...), kde je možnost zatrhnout více možností u dané otázky.

Při shluku více zaškrtávacích políček má uživatel na výběr ze sady možností. Obvykle je nejlepší umístit políčka ve svislém seznamu. Pokud je seznam příliš dlouhý a docházelo by ke scrollování, je přípustné vytvořit více

sloupců [36]. Všechna políčka by měla být umístěna na ose interakce a jejich styl musí být dostatečně kontrastní vůči pozadí GUI.

6.3 Radio buttons (přepínače)

Přepínače mají vizuální podobu téměř shodnou se zaškrtávacími poli. Jediným rozdílem je výběrové pole, které má obvykle podobu kružnice. Přepínače slouží k výběru pouze jedné ze sady možností. Analogii lze opět nalézt jen v podobě papírového formuláře, kde máme možnost označit jedinou možnost.

6.4 Toggle button (přepínací tlačítko)

Umožňuje uživateli vybrat mezi dvěma stavy. Vhodné je každý stav vizuálně odlišit [36]. Obvykle se používá pro stavy jako např. vypnuto/zapnuto. Z pohledu reálného světa se funkčností podobá hardwarovému přepínači, resp. se jedná o tlačítko, které drží svůj aktuální stav.

6.5 List box (seznam)

Seznam povoluje stejně jako zaškrtávací pole vybrat jednu nebo více (záleží na implementaci) ze sady možností. Seznamy jsou kompaktnější než zaškrtávací pole a jsou vhodné pro delší seznam možností. [36].

6.6 Combo box

Combo box je rozbalovací seznam doplněný o textové pole. Rozbalovací seznam slouží uživateli jako sada standardních možností pro výběr. V případě, že nelze vybrat ze seznamu, uživatel pro svůj vstup využije textové pole.

6.7 Dropdown list (rozbalovací seznam)

Dropdown list neboli select box je rozevírací seznam umožňující výběr jedné ze sady možností. Funguje podobně jako přepínač, ale je kompaktnější, a proto umožňuje šetřit místo. Do pole je vhodné přidat popisek jako např. „Vybrat“, aby uživatel rozpoznal nezbytnou akci [36].

6.8 Date/time picker (výběr data a času)

Komponenta umožňující vybrat datum, popř. čas. Při výběru je datum i čas formátován do příslušného formátu, který vstupuje následně do systému [36]. Výběr data a času je tímto způsobem uživatelsky přívětivější a programátorovi odpadá starost s parsováním řetězce.

6.9 Scrollbar (posuvník)

Pomocí ukazatele na posuvníku se vybírá hodnota z daného rozsahu. Hodnotu je někdy možné nastavit i zadáním čísla do vstupního pole a v některých případech lze nastavit velikost kroku při posunu, který je zobrazen vedle posuvníku. Existují vodorovné a horizontální posuvníky. Pomocí této komponenty se lze také pohybovat v rámci GUI a měnit tím grafický obsah.

Analogii lze nalézt v *posuvném potenciometru* jako elektronické součástce.

6.10 Text field (textové pole)

Textové pole umožňuje uživateli zadávat text. Povoluje jeden řádek textu s omezením znaků. Víceřádkové pole je textovou oblastí nazývanou *text area*. Textové pole může mít ve své blízkosti popis, co má dané pole obsahovat. Popis může zastoupit *placeholder*, který umisťuje popis přímo do textového pole v méně kontrastní podobě. V případě číselného formátu je obvykle možné přidávat/ubírat hodnotu pomocí tlačítek v poli. Dostupné jsou vlastnosti pro zadávání kontaktních údajů (telefonní číslo a e-mail).

Textové pole neslouží jen pro prostý text. Dle nastavené vlastnosti formátuje vstupní text. Může sloužit pro zadávání hesla, kdy jsou jednotlivé znaky zobrazovány jako znak $*$.

6.11 Menu

Menu je hlavní navigační a ovládací komponentou vyvolávající akce, které uživateli dovolují procházet jednotlivými sekcemi, resp. přepínat scény uživatelského rozhraní. Prvky menu mohou být v *horizontální* a *vertikální poloze*. Speciálním typem je *kontextové menu*, které závisí na aktuálním kontextu a je vyvolané např. pravým tlačítkem myši nebo stiskem příslušného symbolu.

Z pohledu UX designera tvoří navigační prvky důležitou část GUI. Navržení nevhodné navigace povede k tomu, že uživatel nenalezne co hledá a uživatelské rozhraní opustí. Navigace může rozhodovat a úspěchu či neúspěchu celé aplikace. Při návrhu aplikace je potřeba myslet na návrh informační struktury, která ve výsledku reflektuje podobu menu. Informační strukturu lze rozdělit na *plochou* a *hlubokou*. V případě ploché informační struktury máme v nabídce více sekcí, a tím i méně kliknutí k dosažení cíle. Hluboká struktura naopak vyžaduje více kliknutí k dosažení cíle. Není doporučeno využívat informační strukturu plochou a zároveň hlubokou pro jeden daný účel. Dříve v UX existovalo nepsané pravidlo, že dosažení cíle musí být „na dosah tří kliknutí". Dnes je toto pravidlo překonáno a pokud je navigace jednoduchá a čistá, tak není počet kliknutí důležitý [58].

6.12 Label (štítek)

Zobrazuje text relevantní ke komponentě. Obvykle se jedná o statický text, ale je možné ho měnit na základě vykonané události. U hardwarových i digitálních tlačítek označuje akci, kterou vykonají po jejich stisknutí.

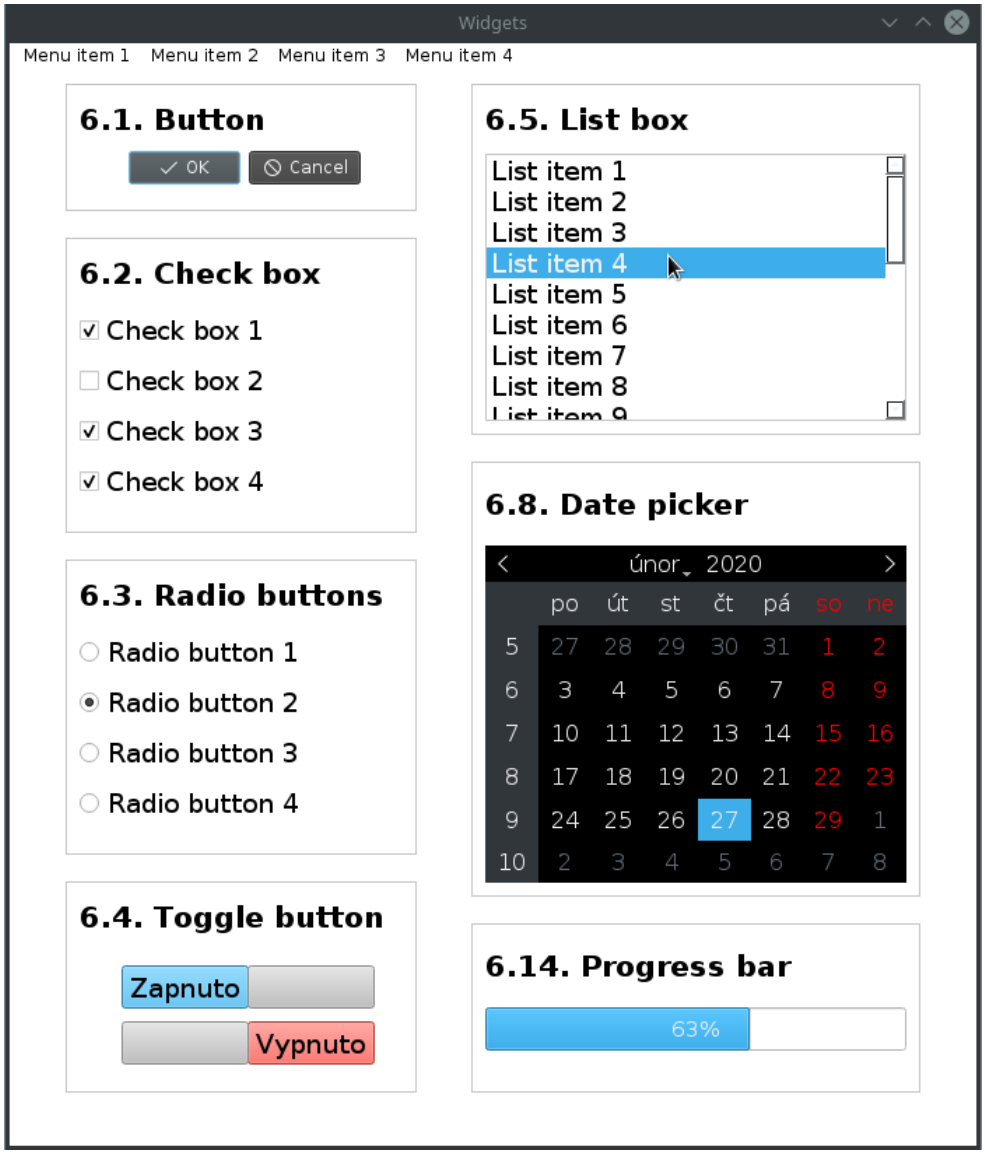
6.13 Tooltip (popis)

Popis sloužící jako nápověda nebo doplňující informace. Zobrazuje se po najetí kurzorem na komponentu, aniž by došlo ke kliknutí. Obvykle má podobu plovoucího okna.

6.14 Progress bar (ukazatel průběhu)

Poskytuje vizualizaci průběhu počítačové operace. Uživatel má přehled o aktuálním stavu operace vyjádřené obvykle v procentech. Využití má tato komponenta při instalaci nebo přenosu souborů. Procentuální vyjádření může být doplněno časovým údajem, který znázorňuje čas zbývající do dokončení operace.

I v UX má ukazatel průběhu velký význam pro zvýšení *konverzního poměru*. Pokud uživatel vidí, že se něco děje a může sledovat aktuální průběh, je větší šance, že neopustí uživatelské rozhraní před dokončením operace. Pro e-shop to může znamenat úspěšné dokončení objednávky. U desktopové aplikace je uživatel informován o stavu daného procesu a neukončí aplikaci při zápisu dat na disk – GUI může zobrazovat zbývající čas nebo množství dat do dokončení a uživatel nenabývá dojmu, že došlo k zamrznutí aplikace.



Obrázek 6.1: Ukázka widgetů z kapitoly 6

7 Referenční aplikace

K detailnějšímu otestování a následnému porovnání knihoven pro tvorbu UI bylo nutné vytvořit aplikaci (souborový manažer) a k ní příslušné uživatelské rozhraní. Jako základ byla využita semestrální práce z předmětu KIV/ZOS, jejímž tématem byl *zjednodušený souborový systém založený na i-uzlech*. Semestrální práce byla značně rozšířena pro potřeby této bakalářské práce, resp. pro potřeby referenčního grafického návrhu aplikace.

7.1 Backend aplikace

Jak bylo zmíněno v kapitole 7, součástí aplikace je *zjednodušený souborový systém založený na i-uzlech*, který tvoří backend[1]. I-uzel je datová struktura uchovávající metadata o souborech a adresářích. Používá se v unixových souborových systémech vycházejících z klasického UFS (*Unix File System*), např. linuxová řada ext2, ext3 a ext4. Každý soubor a adresář je reprezentován jedním i-uzlem. Metadata drží informaci o čísle i-uzlu, velikosti souboru a datu poslední modifikace. Dále obsahují *přímé* a *nepřímé* adresy na datové bloky. V i-uzlu není uveden název souboru – ten lze nalézt v adresáři, ve kterém je každá položka tvořena dvojicí (název souboru a číslo i-uzlu). Při formátování disku je inicializován maximální počet i-uzlů, takže i maximální počet souborů. V případě plné obsazenosti i-uzlů již nemůžeme vytvořit další – pokud zbývají datové bloky, můžeme prodloužit stávající soubor. Souborový systém poskytuje následující funkce:

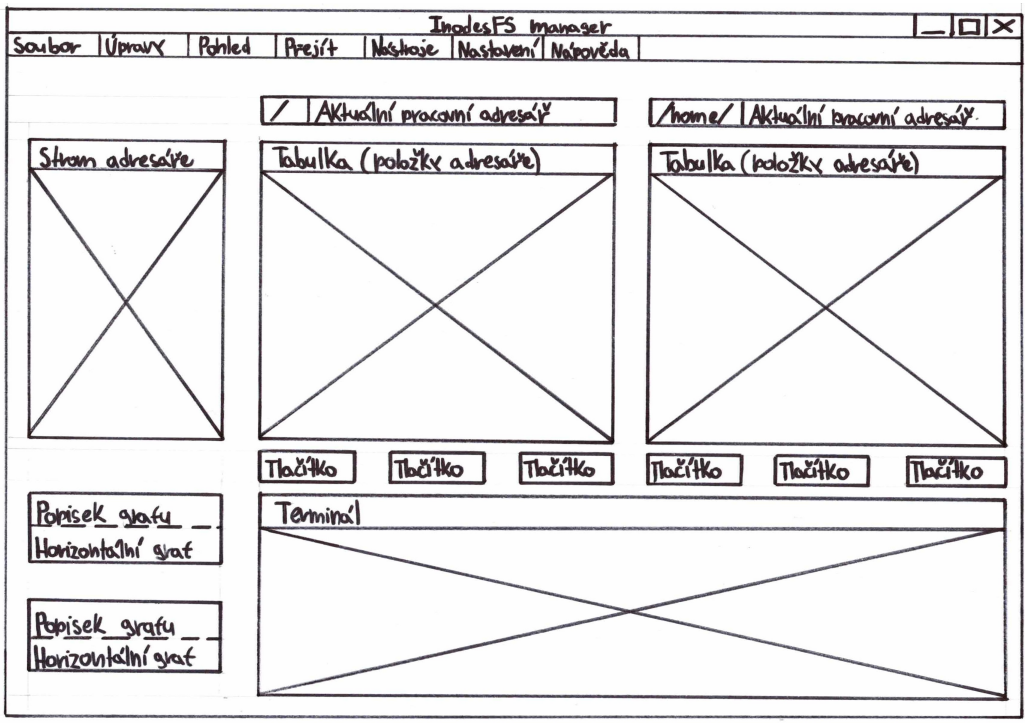
1. Formátování disku – `bool format_disk(...)`
2. Načtení disku – `bool read_disk(...)`
3. Nahrání souboru z lokálního uložiště – `void incp_shell(...)`
4. Stažení souboru ze souborového systému – `void outcp_shell(...)`
5. Výpis obsahu souboru – `char *cat_shell(...)`
6. Smazání souboru – `void rm_shell(...)`
7. Smazání adresáře – `void rmdir_shell(...)`
8. Výpis obsahu adresáře – `void ls_shell(...)`

[1]Část kódu tvořící logickou část aplikace, která není viditelná pro uživatele.

9. Změna adresáře – `void cd_shell(...)`.

7.2 Grafický návrh uživatelského rozhraní

Ovládání souborového systému je uživatelsky přívětivější pomocí nějakého GUI. Nejdříve byl vytvořen wireframe[2]. Na jeho základě byl vytvořen referenční grafický návrh (včetně všech případů užití) pomocí nástroje *Qt Designer*, který sloužil jako základní bod pro tvorbu GUI ve vybraných knihovnách. Níže jsou uvedeny jednotlivé případy užití, které musí výsledné aplikace splňovat.

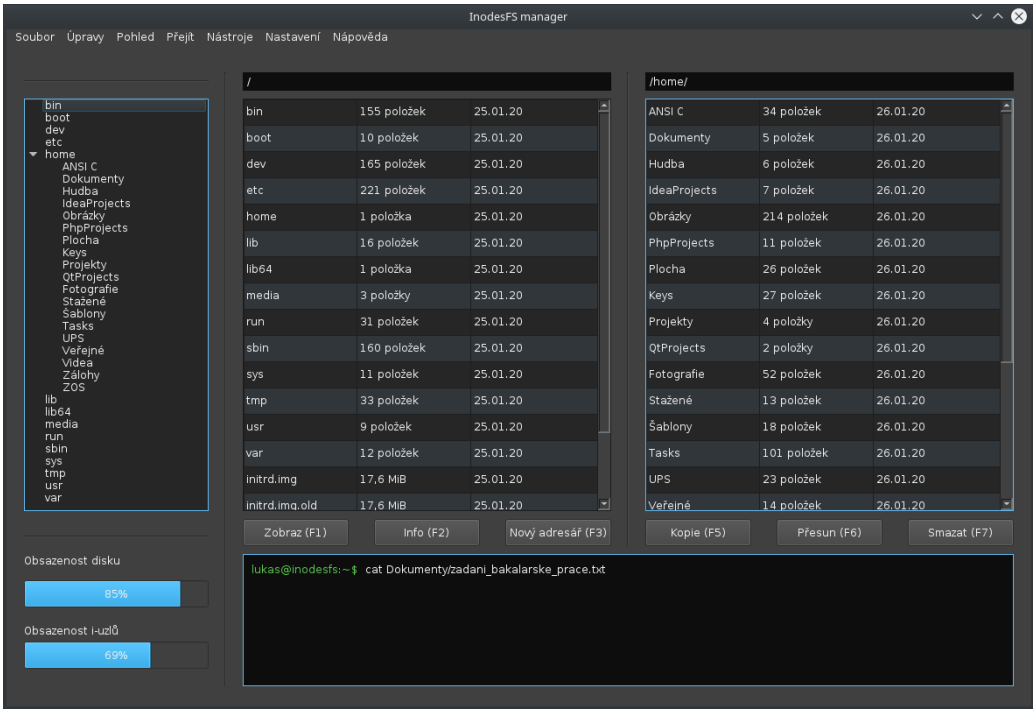


Obrázek 7.1: Wireframe uživatelského rozhraní

[2]Drátěný model, který abstraktně definuje vzhled aplikace a rozložení ovládacích prvků uživatelského rozhraní.

7.2.1 Základní zobrazení

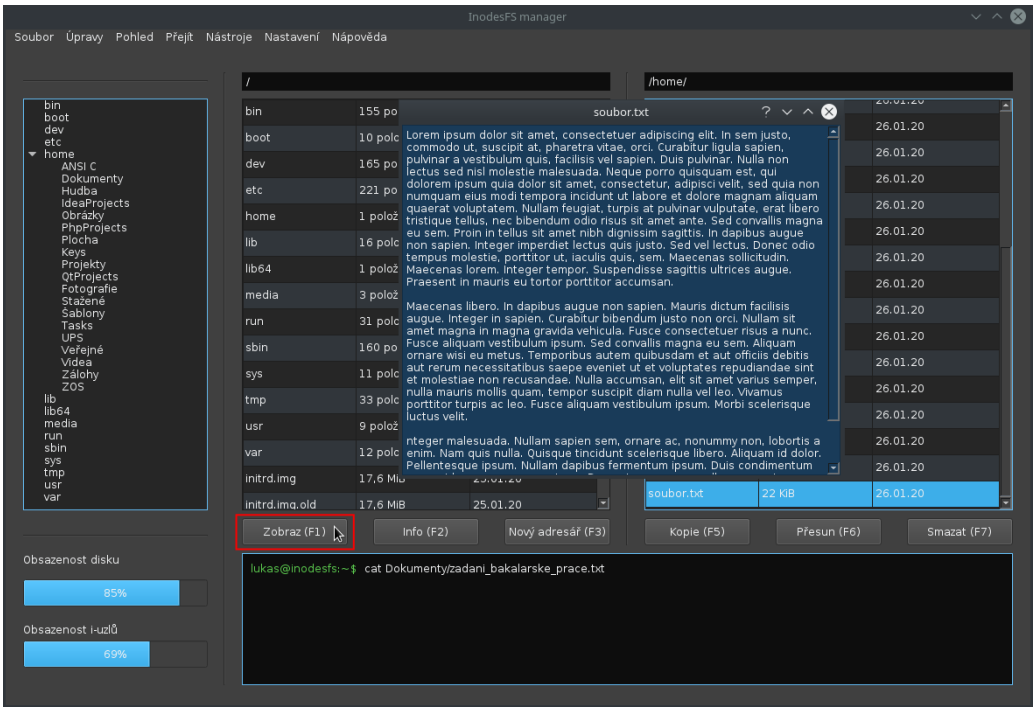
Po spuštění aplikace bude zobrazen níže uvedený návrh základního zobrazení aplikace. Je tvořen horizontálním menu, které slouží pro obsluhu samotné aplikace. Hlavními komponentami jsou dvě tabulky zobrazující položky daných adresářů. Nad každou tabulkou je popisek zobrazující absolutní cestu umístění adresáře. Vedle tabulek je zobrazena stromová struktura aktuálně používaného adresáře. Ve spodní části rozhraní jsou umístěna tlačítka sloužící pro obsluhu VFS. Pod komponentou zobrazující stromovou strukturu jsou umístěny dva grafy znázorňující obsazenost disku a obsazenost i-uzlů. Poslední komponenta emuluje funkci terminálu.



Obrázek 7.2: Základní zobrazení po spuštění aplikace

7.2.2 Zobrazení obsahu souboru

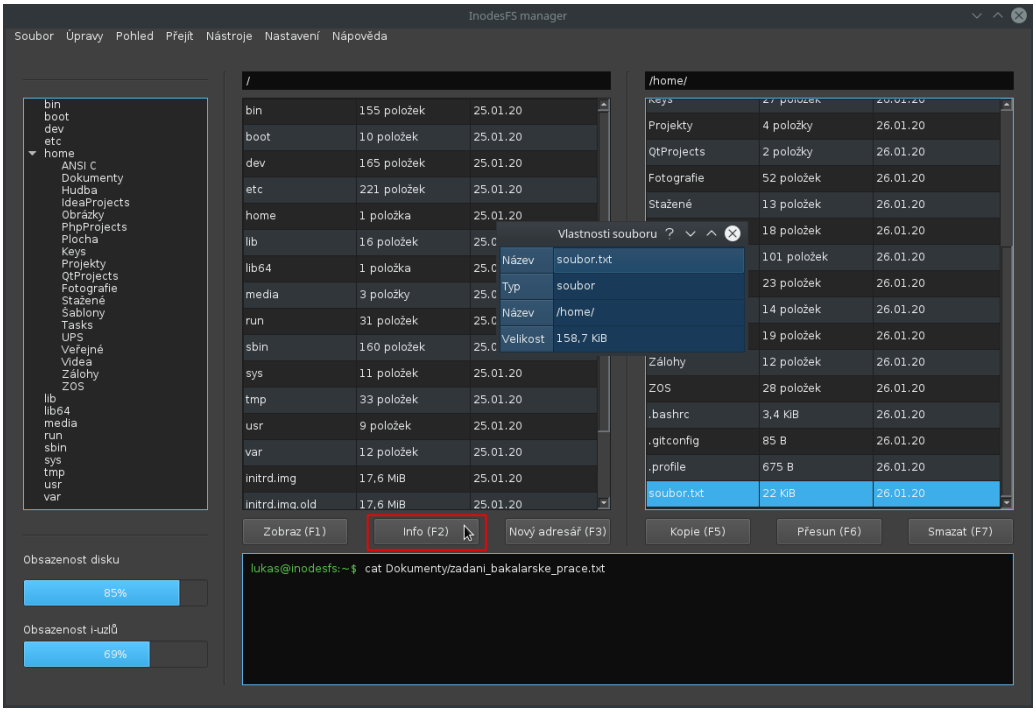
Prvním případem užití je zobrazení obsahu souboru. Po stisku funkční klávesy `F1` nebo stisku patřičného tlačítka dojde k zobrazení modálního okna, ve kterém bude vypsán libovolný obsah souboru. Modální okno bude mít *vertikální scrollbar* (Pro případ většího množství obsahu, který by se nevešel do nativní velikosti modálního okna.), aby byl možný pohyb v obsahu souboru. Na pozadí dochází k volání funkce `cat`, která za pomocí VFS připraví data do příslušných struktur a následně je zobrazí GUI.



Obrázek 7.3: Zobrazení obsahu souboru

7.2.3 Zobrazení vlastností souboru

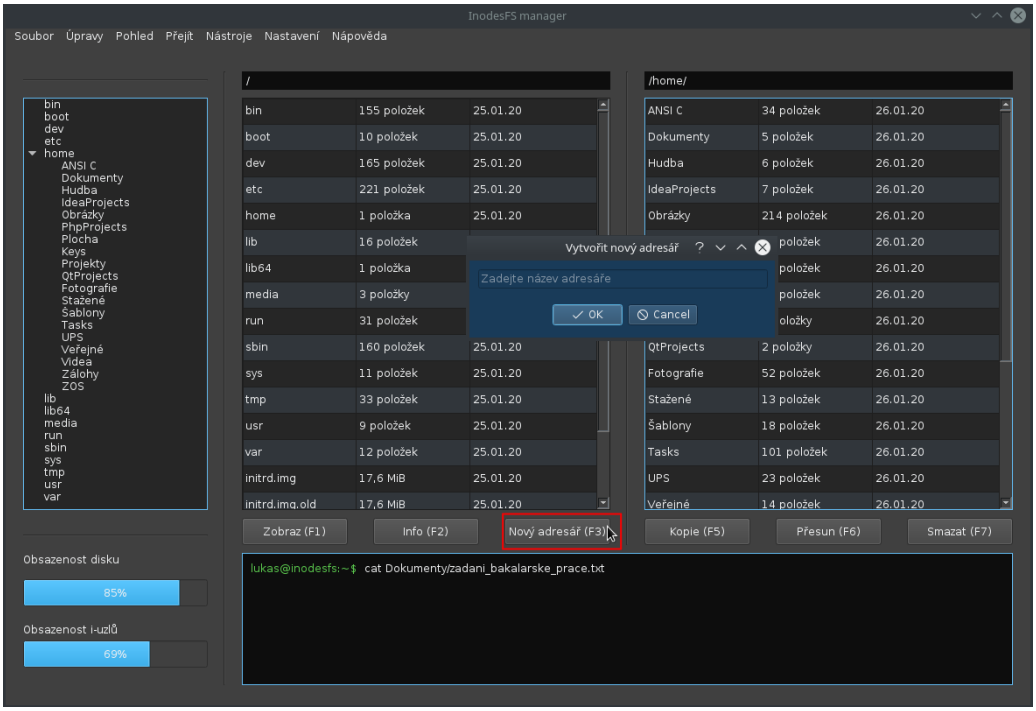
Druhým případem užití je zobrazení vlastností souboru nebo adresáře. Modální okno s vlastnostmi se zobrazí po stisku funkční klávesy `F2` nebo po stisku daného tlačítka. V okně bude tabulka zobrazující vlastnosti jako je název položky adresáře, typ položky (soubor nebo adresář), absolutní cesta k adresářové položce, velikost a poslední datum modifikace. Pro přípravu těchto dat není připravena žádná funkce, ale veškerá data ke každé položce adresáře jsou obsažena v jednotlivých *i-uzlech*. Načtení těchto dat do příslušných struktur dochází voláním funkce `ls`, která byla pro tyto účely upravena.



Obrázek 7.4: Zobrazení vlastností souboru

7.2.4 Vytvoření nového adresáře

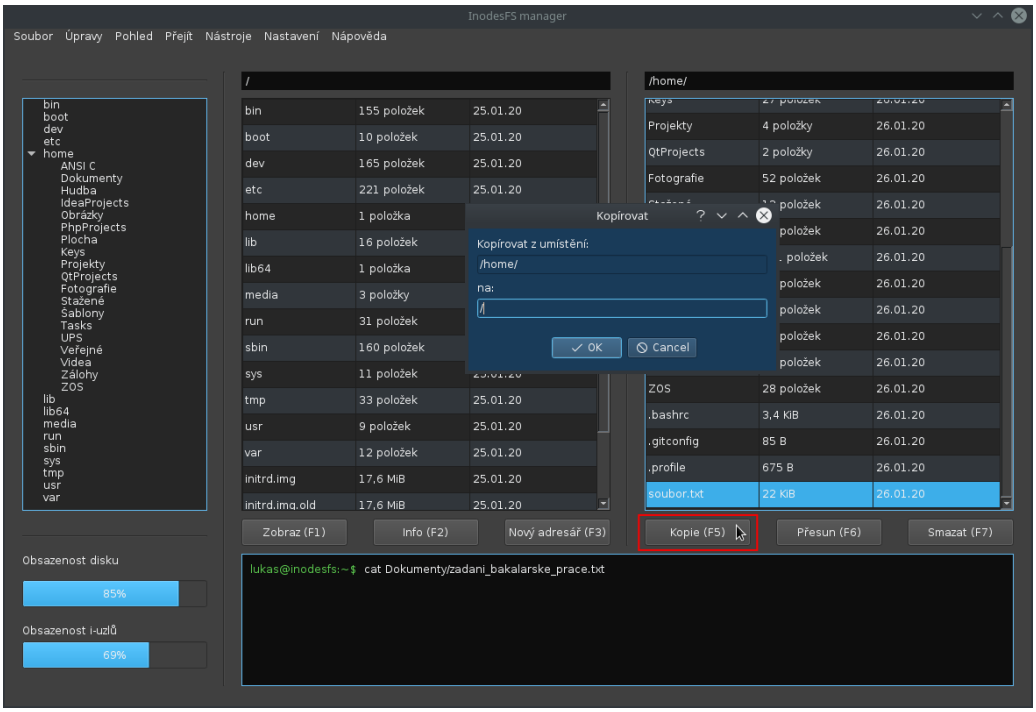
Pro vyvolání modálního okna pro vytvoření nového adresáře je potřeba stisknout funkční tlačítko `F3` nebo příslušné tlačítko. Okno bude obsahovat vstupní textové pole pro zadání názvu adresáře. Nesmí chybět ani dialogová tlačítka pro potvrzení vytvoření adresáře nebo zrušení vyvolané akce a neprovádění žádných změn na disku. Na pozadí dochází k volání funkce `mkdir`, která zapisuje nový adresář na disk.



Obrázek 7.5: Vytvoření nového adresáře

7.2.5 Kopírování souboru

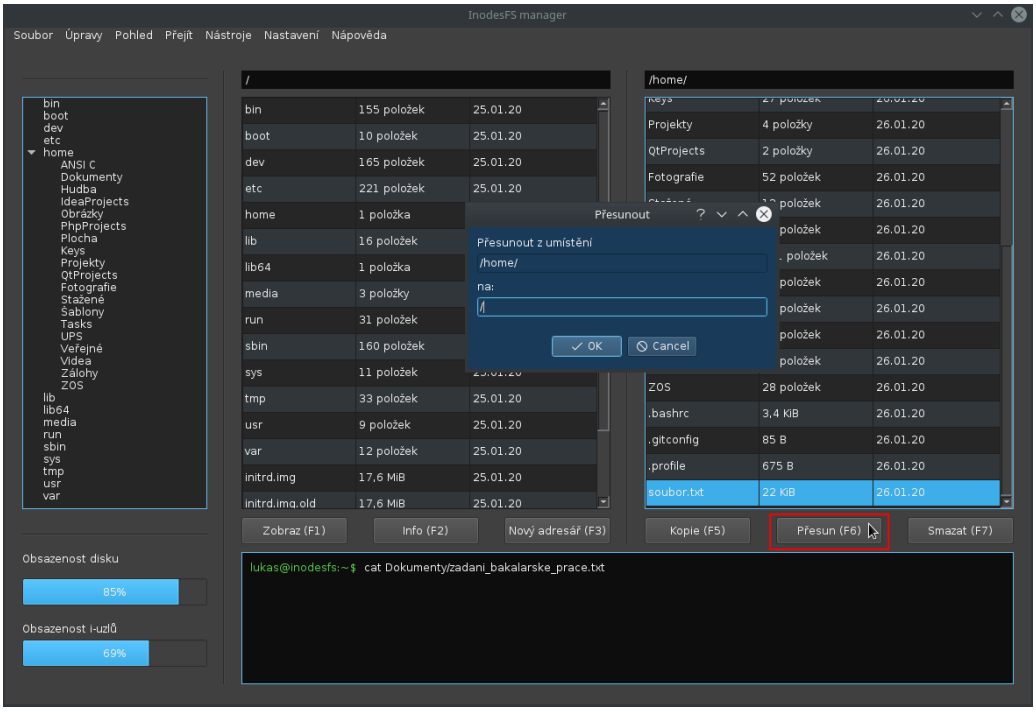
Po stisku funkční klávesy `F5` nebo odpovídajícího tlačítka se zobrazí modální okno pro možnost kopírování souboru. Bude zobrazovat absolutní cestu umístění kopírovaného souboru a cílovou cestu. Pod těmito textovými poli budou umístěna dialogová tlačítka pro potvrzení kopírování nebo zrušení celé akce bez provedení změn na disku. Po úspěšném potvrzení okna bude volána funkce `cp`.



Obrázek 7.6: Kopírování souboru z aktuálního umístění na nové

7.2.6 Přesun souboru

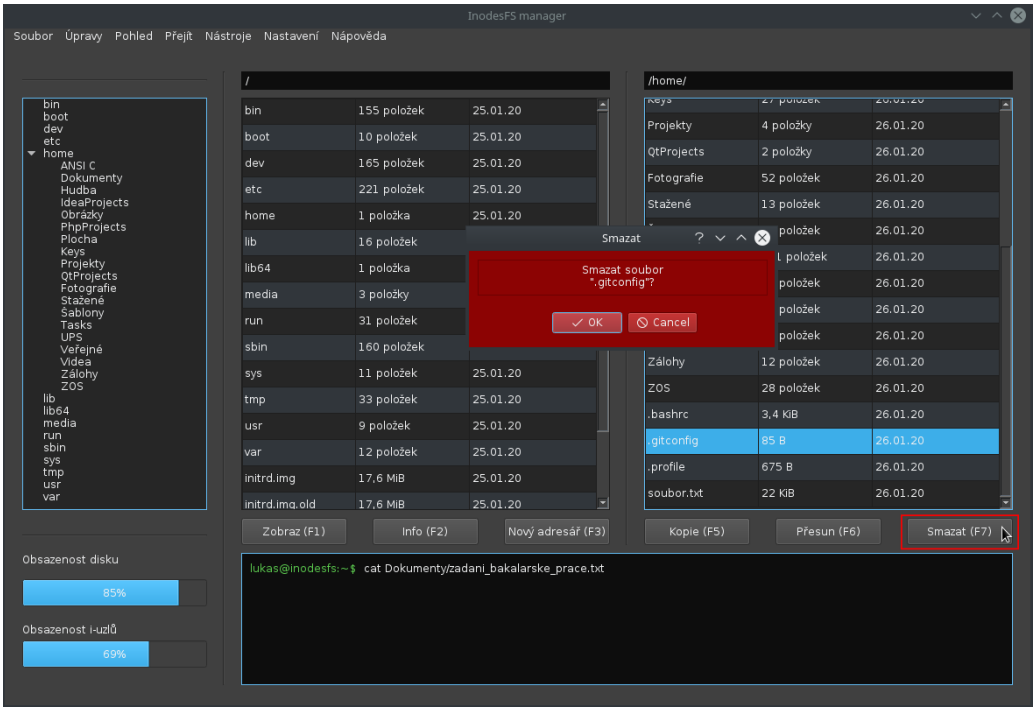
Přesun souborů jé téměř totožný s předchozím případem užití 7.2.5. Stejné rozvržení modální okna se objeví po stisku funkční klávesy `F6` nebo tlačítka. Jedinou změnou je volání funkce `cp` s jinými parametry, kdy dochází k odstranění souboru z původního umístění a přesun na nové.



Obrázek 7.7: Přesun souboru z aktuálního umístění na nové

7.2.7 Mazání adresářové položky

Posledním případem užití je mazání souboru nebo adresáře. K vyvolání modálního okna dochází po stisku funkční klávesy `F7` nebo opět po stisku tlačítka. Okno by mělo mít výraznou barvu (v tomto případě červenou), aby v uživateli vzbudilo pozornost, že se jedná o krok, který nelze vzít zpět. V okně bude zobrazeno jméno mazané položky adresáře a pod ním dialogová tlačítka, aby uživatel potvrdil nenávratné smazání položky. Po potvrzení se rozhodne o funkci, která bude volána. V případě mazání souboru se použije funkce `rm` a pro adresář je připravena funkce `rmdir`.



Obrázek 7.8: Mazání souboru

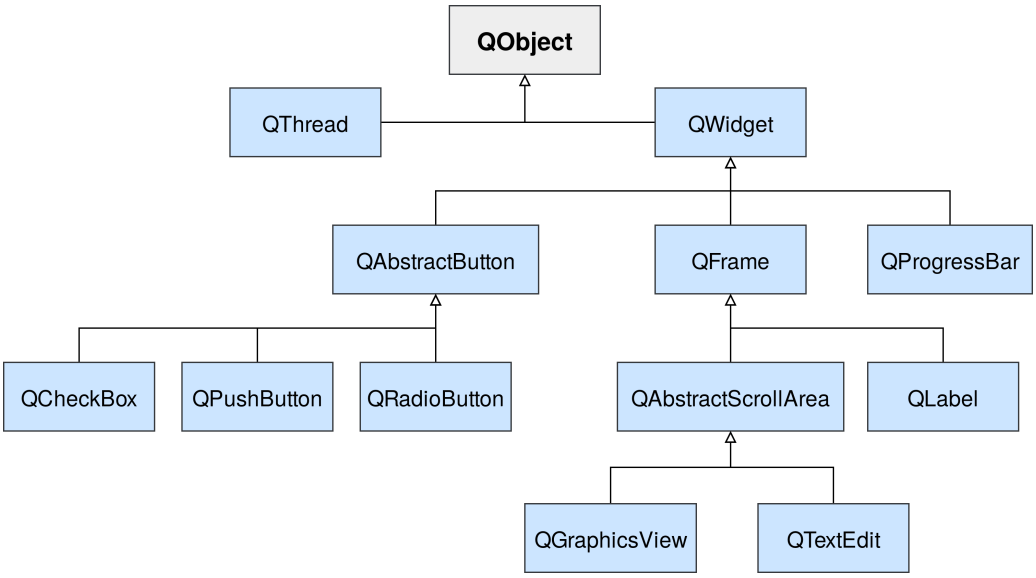
8 Vývoj aplikace v knihovně Qt

Před samotným vývojem v Qt je nutné mít alespoň základní kvantum znalostí spojených s programovacím jazykem *C++*. Až po splnění této podmínky je možné začít s vývojem referenční aplikace. V první podkapitole je aplikace znázorněna abstraktním pohledem jako *hierarchie tříd* pomocí UML (*Unified Modeling Language*) diagramu, aby byla viditelná provázanost (dědičnost) mezi jednotlivými widgety. V následující podkapitole je zobrazen *životní cyklus aplikace*, pomocí kterého je popsán a vizualizován běh aplikace od jejího spuštění až po její ukončení. Jednotlivé etapy životního cyklu jsou dále popsány od inicializace okna aplikace, které představuje hlavní widget a zároveň je kontejnerem pro další widgety tvořící aplikaci. Po inicializaci hlavního okna je nutná implementace ostatních widgetů. Mezi aplikací a uživatelem probíhá interakce, a proto je nutné zajistit, aby aplikace reagovala na uživatelovy podněty pomocí mechanismu signálů a slotů. Qt je využíváno v komerční sféře, takže v posledních kapitolách je vývoj hodnocen z ekonomického pohledu, resp. časové náročnosti vývoje.

Aplikace je tvořena dvěma částmi, kdy backend tvoří VFS napsaný v programovacím jazyce *C*, jehož soubory jsou externě vloženy do souborů programovacího jazyka *C++*, ve kterém je napsáno GUI. VFS zajišťuje veškeré operace nad virtuálním diskem vyvolané skrze GUI.

8.1 Hierarchie tříd widgetů v Qt

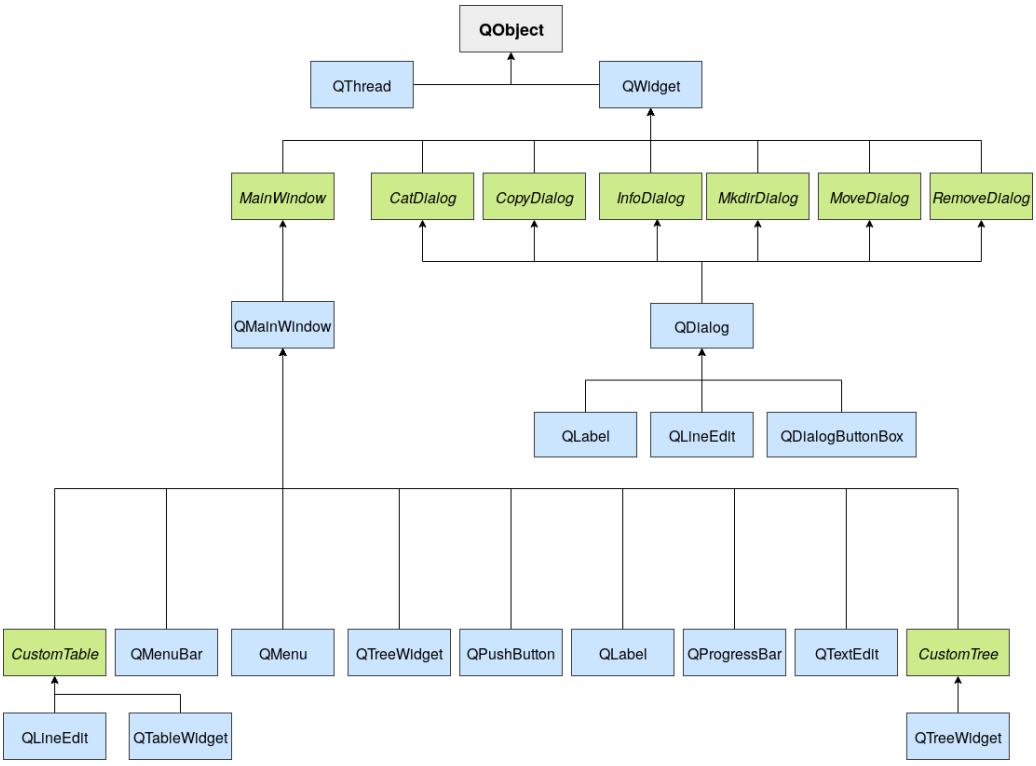
Qt využívá dědičnost, a to zejména v modulu *Widgets*, kdy na vrchu hierarchie stojí třída `QObject`. Jedná se o nejzákladnější třídu v Qt a většina tříd dědí implementaci právě od ni. Obsahuje výkonné funkce a mezi ty nejvýznamnější patří: rodičovský systém, mechanismus signálů a slotů a řízení událostí. Všechny widgety dědí od třídy `QObject` a nejzákladnější widget tvoří třída `QWidget`. Tato třída obsahuje většinu vlastností (kurzor myši, poloha, velikost, popisky, ...) sloužících pro popis okna nebo widgetu [23]. V UML diagramu tříd 8.1 je pro představu uvedeno jen několik tříd widgetů.



Obrázek 8.1: Hierarchie tříd widgetů v knihovně Qt znázorněná pomocí UML diagramu [23]

8.2 Hierarchie tříd widgetů v aplikaci

Na UML diagramu tříd 8.2 jsou zobrazeny všechny widgety, resp. jejich třídy implementované v rámci aplikace. Modře znázorněné třídy jsou v Qt nativní. Zeleně znázorněné jsou naopak vlastní třídy. Na rozdíl od předchozího UML diagramu 8.1 věrně nezobrazuje dědičnost tříd. Cílem není zobrazit dědičnost, ale použité widgety v rámci daného layoutu. Třída `MainWindow` tvoří hlavní okno aplikace a sdružuje všechny widgety vyjma modálních oken. Kromě nativních tříd obsahuje vlastní třídu `CustomTable` (Vlastní widget sloužící pro zobrazení prvků v adresáři.) a `CustomTree`. Třídy `CatDialog`, `CopyDialog`, `InfoDialog`, `MkdirDialog`, `MoveDialog` a `RemoveDialog` jsou modální okna dědící implementaci od třídy `QDialog`. Pomocí těchto modálních oken je možné vykonávat operace nad VFS, resp. vykonávat případy užití definované v podkapitolách výše. V layoutu modálního okna jsou obsaženy třídy `QLabel`, `QLineEdit` a `QDialogButtonBox`, které informují uživatele o dané operaci a poskytují vstupy pro data.



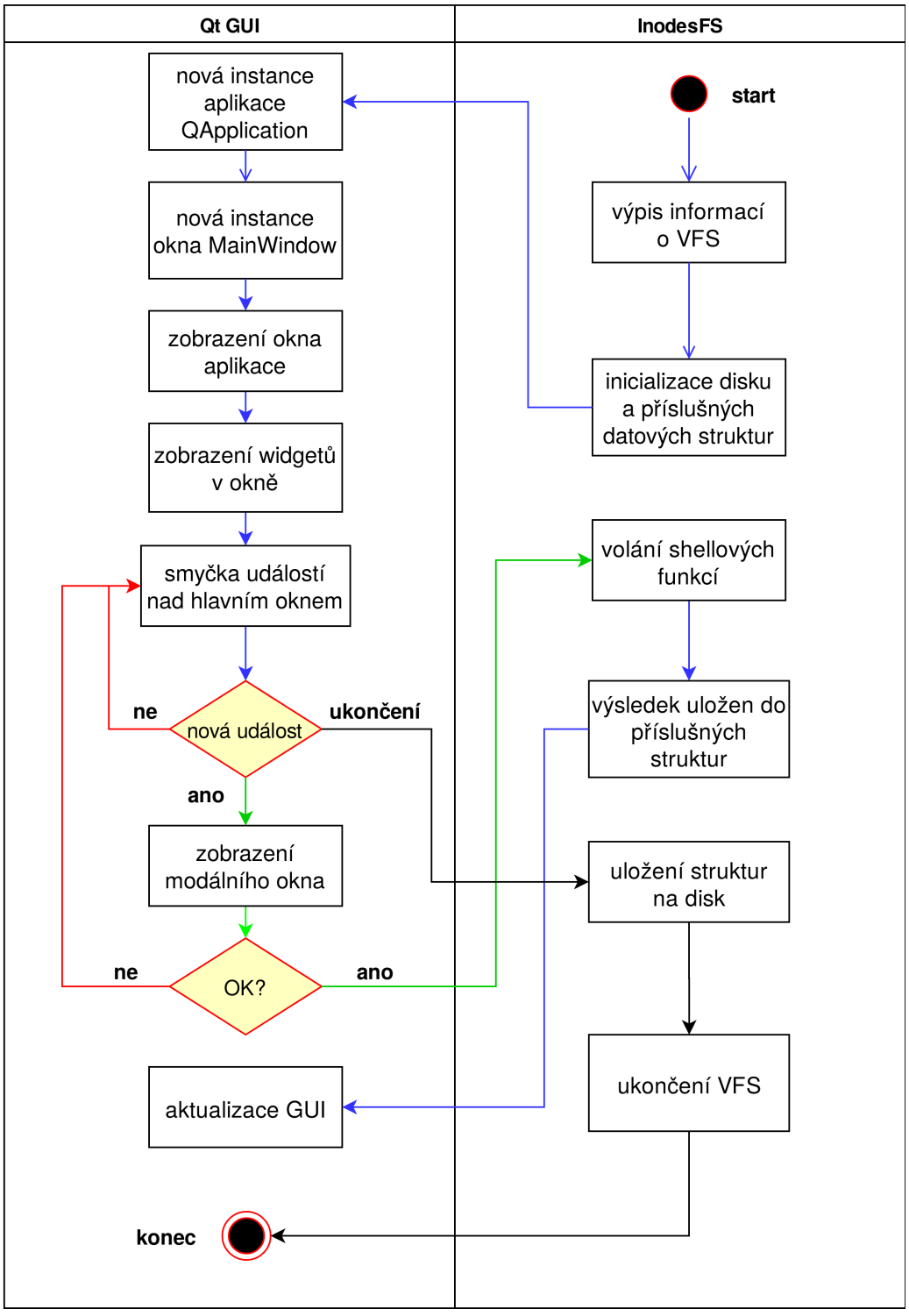
Obrázek 8.2: Hierarchie tříd widgetů v knihovně Qt implementovaných v aplikaci

8.3 Životní cyklus aplikace

Životní cyklus aplikace je protokol, podle kterého se aplikace chová od jejího spuštění až po vypnutí. Diagram 8.3 je rozdělen na dvě části, protože backend aplikace tvoří VFS v diagramu nazvaný jako *InodesFS* a frontend[1] *Qt GUI*.

Po spuštění aplikace jsou nejdříve do konzole vypsány základní informace o VFS a následně je inicializován virtuální disk a příslušné datové struktury. Po inicializaci je vytvořena nová instance třídy `QApplication` dědící implementaci od třídy `QGuiApplication`. Třída `QApplication` poskytuje některé funkce potřebné pro aplikace založené na widgetech, resp. na třídě `QWidget`, od které dědí implementaci veškeré třídy widgetů. Dále je vytvořena instance vlastní třídy `MainWindow` dědící od třídy `QMainWindow`, která vytváří hlavní okno aplikace. Toto okno zároveň tvoří kontejner pro další widgety zobrazující se v hlavním layoutu. Po inicializaci všech widgetů je nad instancí třídy `QApplication` vytvořena smyčka událostí přijímající události ze systémů okna a následně je odesílá do widgetů aplikace – smyčka zajišťuje interakci mezi uživatelem a aplikací. Při vytvoření nové události dojde k zobrazení modálního okna dědící implementaci od třídy `QDialog`. Do modálního okna zadá uživatel své vstupy. Při kladném potvrzení dochází k volání shellových funkcí. Funkce zpracuje uživatelovy vstupy, výsledek uloží do příslušných struktur a následně dochází k aktualizaci GUI. Při negativním potvrzení modálního okna se program vrací do čekací smyčky událostí nad hlavním oknem. Může nastat událost ukončující běh aplikace. Po této události dojde k uložení aktuálního stavu struktur na virtuální disk, ukončuje se VFS a následně i GUI.

[1] Část aplikace viditelná pro uživatele.



Obrázek 8.3: Životní cyklus aplikace v Qt [23]

8.4 Obsluha událostí

Po inicializaci aplikace a načtení všech struktur se čeká na vznik události. Události vznikají nad hlavním oknem, ale i nad samotným aktivním widgetem. U každého widgetu je tedy možné definovat vzniklé události. Mimo třídu widgetu se definují události nad ostatními třídami, od kterých daný widget dědí implementaci. Vzniká určitá forma zapouzdření – nad třídou widgetu jsou definovány události spjaté právě s daným widgetem a nad třídami `QWidget` a `QObject` definujeme obecné události. Pro obsluhu událostí je v Qt využíván *mechanismus signálů a slotů* nebo obsluha za pomocí objektů odvozených z abstraktní třídy `QEvent`.

8.4.1 Mechanismus signálů a slotů

Většina knihoven pro tvorbu GUI má mechanismus pro detekci uživatelské akce a reakci na tuto akci. Využívají mechanismu *zpětného volání* nebo *posluchače*, ale všichni jsou inspirováni vzorem *pozorovatele*. Tento vzor definuje pozorovaný objekt (např. tlačítko) a při změně jeho stavu jsou upozorněny příslušné další objekty (např. zobrazení modálního okna) [23].

Qt využívá mechanismus signálů a slotů, který je alternativou ke zpětnému volání. Signál je zpráva, kterou může odeslat za účelem změny jeho stavu. Slot je naopak metoda, která přijímá signál a reaguje na něj. Je možné definovat i vlastní signály a sloty, ale v rámci aplikace byly využity jen ty nativní. Na ukázce kódu 8.1 je první metoda slotem, která je reakcí na signál kliknutí do oblasti tabulky zobrazující adresářovou strukturu. Druhá metoda na ukázce je slotem pro signál generovaný stiskem tlačítka pro vyvolání modálního okna vytvářející nový adresář.

Mechanismus signálů a slotů je implementován v rámci oficiálního IDE pro Qt s názvem *Qt Creator*. Tato skutečnost zlepšuje spolupráci UI/UX designera a programátora. V podkapitole 8.5 je zmíněno, že oficiální IDE obsahuje integrovaný *Qt Designer* pro vytváření GUI. Skrze něj má designer možnost vygenerovat metodu pro slot reagující na příslušný signál, a tím definovat případy užití rozhraní. Nepotřebuje mít žádné znalosti z programování, ale jen nad aktivním widgetem zvolí možnost *Go to slot* a vybere ze seznamu. Každý widget má řadu předefinovaných slotů, které je možné využít. Připravených slotů je opravdu velké množství a ani v rámci vývoje referenční aplikace nebylo potřeba definovat vlastní. Programátor následně dopíše tělo metody, a tím je zajištěna funkčnost slotu.

```
// customtable.h
private slots:
    void on_tableWidget_clicked();

// custumtable.cpp
void CustomTable::on_tableWidget_clicked() {
    if (listPosition == LEFT) {
        is_left_list_active = true;
        is_right_list_active = false;
    }
    else if (listPosition == RIGHT) {
        is_right_list_active = true;
        is_left_list_active = false;
    }
}

// mainwindow.h
private slots:
    void on_mkdirButton_clicked();

// mainwindow.cpp
void MainWindow::on_mkdirButton_clicked() {
    MkdirDialog mkdirDialog;
    mkdirDialog.setModal(true);
    mkdirDialog.exec();

    if (is_left_list_active) {
        ls_shell(NULL, 1);
        ui->leftList->prepareListData();
    }
    else if (is_right_list_active) {
        ls_shell(NULL, 2);
        ui->rightList->prepareListData();
    }
}
```

Listing 8.1: Ukázka reakcí na signály

8.4.2 Události řízené abstraktní třídou QEvent

Dalším mechanismem pro řízení událostí jsou objekty odvozené od abstraktní třídy `QEvent`. Jedná se o události vznikající v rámci aplikace v důsledku vnějších aktivit. Vzniklé události může přijímat a zpracovávat libovolná instance dědící implementaci od třídy `QObject`, ale je to relevantní hlavně pro widgety [7].

Události řízené abstraktní třídou `QEvent` a za pomocí *signálů a slotů* jsou

dva paralelní mechanismy sloužící ke stejné věci. Obecně platí, že události řízené třídou `QEvent` jsou generovány vnější entitou, takže se jedná o stisknutí různých kláves nebo událost vyvolaná myší (stisk tlačítka nebo rolování kolečkem). Pro generování událostí je nutné upravit předchozí kód aplikace s použitím metody `QObject::installEventFilter()`.

Naproti tomu se signály a sloty snáze generují a přijímají a je možné propojit dvě libovolné podtřídy `QObject`. Většina komunikace mezi třídami probíhá za pomocí signálů a slotů. Signály mohou být doručeny okamžitě – při použití vláken mohou být zařazeny do fronty a zpracovány až později.

```
// customtable.cpp
void CustomTable::keyPressEvent(QKeyEvent *event) {
    int index = tableWidget->currentRow();

    if ((event->key() == Qt::Key_Enter) ||
        (event->key() == Qt::Key_Return)) {
        cdShellList(index);
    }
}
```

Listing 8.2: Události nad abstraktní třídou QEvent

8.4.3 Spojení signálu a slotu

Po vytvoření nového signálu a slotu musí dojít ke spojení obou metod. Qt poskytuje metodu `QObject::connect`, která má 3 důležité parametry. Prvním z nich je *objekt*, který je odesílatelem signálu. Druhý parametr tvoří *metoda generující signál* a poslední parametr je *metoda slotu*. Po úspěšném zavolání metody s příslušnými parametry dojde ke vzájemnému spojení signálu a slotu. Nyní slot může reagovat na signál, ale zároveň může být spojení využito k předávání informací. Na ukázce kódu 8.3 se jedná o instanci třídy `QTableWidget`, resp. o atribut této třídy tvořící instanci třídy `QHeaderView`. V aplikaci nebylo spojení v signálu a slotu využito, ale na ukázce je indikace změny řazení tabulky podle nějakého parametru. Při změně řazení dochází k volání funkce aktualizující data pro tabulku představující adresářovou strukturu.

```
// customtable.cpp
QTableWidget *tableWidget = ui->tableWidget;
QHeaderView *headerView = tableWidget->horizontalHeader();
ui->tableWidget->installEventFilter(this);

connect(headerView, &QHeaderView::sectionClicked,
        [this](int logicalIndex) {
```

```
8       QString text = tableWidget->horizontalHeaderItem(
        logicalIndex)->text();
9
10      refreshListTable();
11  });
```

Listing 8.3: Spojení signálu a slotu

8.5 Vytvoření uživatelského rozhraní

Pro vytváření GUI v Qt existují dva oficiální nástroje. Prvním z nich je *Qt Designer*, který je primárně určen pro vytváření desktopových GUI složených z widgetů. Druhým nástrojem je *Qt Quick* sloužící pro vytváření QML[2] aplikací se složitějším GUI nevyužívajícím standardní widgety, které mají větší nároky na vizuální stránku aplikace a použité efekty. Qt Quick je používán spíše pro vývoj mobilních aplikací. Oba dva nástroje mohou být integrovány do IDE *QT Creator*.

Pro vytvoření GUI referenční aplikace byl využit nástroj *Qt Designer*, který nabízí širokou paletu použitelných widgetů a kontejnerů definující jejich rozložení. Nejdříve byly do okna aplikace vloženy kontejnery – volba kontejnerů a jejich pozice je velice důležitá pro responzivitu celé aplikace a k následné implementaci widgetů do kontejnerů. Pomocí tohoto nástroje lze definovat vzhled jednotlivých widgetů, a to i za pomocí vloženého CSS (*Cascading Style Sheets*). Dále bylo potřeba vytvořit modální okna a k ním příslušné sloty, aby je bylo možné po stisku tlačítka vyvolat. V rámci Qt Designeru lze generovat základní deklaraci metody slotu k příslušnému signálu a poté dopsat její obsluhu.

Výstupem Qt Designeru je soubor s koncovkou `.ui`, ve kterém je kód napsán ve značkovacím jazyce XML (*Extensible Markup Language*), který generuje sám nástroj. XML kód je možné psát rovnou do souboru – při vývoji aplikace se do kódu zasahovalo jen při ladění nebo pokud bylo nutné duplikovat část kódu. Není ani doporučeno zasahovat do vygenerovaného XML a lze tím předejít potenciálním chybám.

8.6 Responzivita uživatelského rozhraní

Cílem nebylo vytvořit plně responzivní GUI, které by se přizpůsobovalo všem zařízením, ale bylo žádoucí, aby jednotlivé widgety reagovaly na aktuální ve-

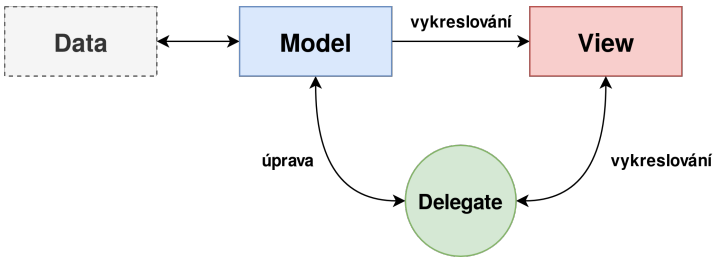
[2]Qt Modeling Language – značkovací jazyk pro návrh aplikací zaměřených na uživatelské rozhraní.

likost okna. V kapitole 8.5 bylo zmíněno, že Qt Designer obsahuje různé kontejnery sloužící pro rozmístění widgetů. Kromě pozicování widgetů mohou při správném použití zajistit responzivitu aplikace.

Při návrhu hlavního okna i jednotlivých modálních oken se osvědčilo, aby hlavním obalovacím kontejnerem byl `Grid Layout` nebo `Form Layout`. V obou případech se jedná o kontejner vytvářející mřížku, do které lze vkládat widgety nebo další kontejnery. V případě modálních oken byly do mřížky vkládány rovnou widgety. Naopak do hlavního okna aplikace byl použit kontejner `Vertical Layout` pro řádkové rozložení a `Horizontal Layout` pro vertikální rozložení widgetů. Mimo správného použití kontejnerů je nutné, aby žádný z widgetů neměl nastavenou absolutní velikost. Kontejnery mají vestavěnou vlastnost `layoutStrech`, pomocí které lze nastavit poměr velikostí prvků nacházejících se v kontejneru.

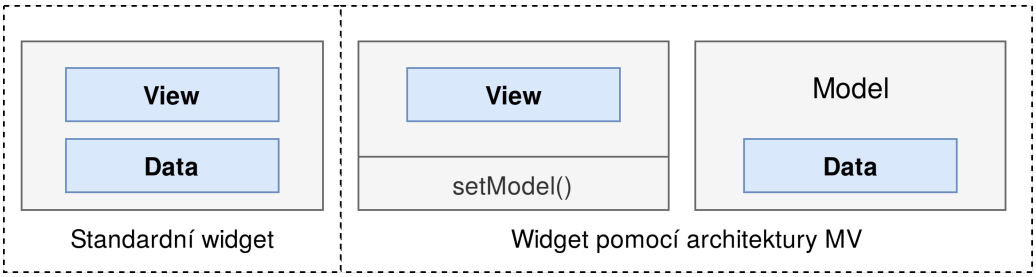
8.7 Datový model widgetů

Qt nepoužívá klasický návrhový vzor MVC (*Model-view-controller*), který je často používán při vytváření uživatelských rozhraní. Samotné MVC se skládá ze tří druhů objektů. *Model* je aplikační objekt, který obsahuje veškerou logiku a provádí operace s daty. *View* zobrazuje na obrazovce např. widget s výstupními daty. *Controller* definuje způsob reakcí na uživatelovy podněty a funguje jako „prostředník“ mezi modelem a controllerem. Qt používá návrhový vzor MV (*Model/View*), ve kterém má model stejnou funkčnost jako v architektuře MVC, ale objekt view představuje kombinaci samotného view s controllerem. Funkci controlleru zastává instance třídy `Delegate` – při úpravě položky komunikuje přímo s modelem, resp. modelem indexů [39].



Obrázek 8.4: Model/View architektura [39]

Widgety lze vytvářet i bez architektury MV, protože Qt stále podporuje standardní widgety, jejichž součástí jsou i data. Cílem bakalářské práce bylo otestovat možnosti GUI, a proto byly využity standardní widgety, protože jejich vzhled je totožný a liší se jen pohled na data. Pokud by aplikace např.



Obrázek 8.5: Dva způsoby implementace widgetu [24]

načítala velké množství dat z databáze pomocí SQL, použití architektury MV by bylo vhodnější a načítání dat o několik řádů rychlejší. Na ukázce kódu 8.4 je ve zkrácené formě zobrazeno vytvoření widgetu tabulky pomocí architektury MV a standardní cestou [39].

```
// ownmodel.h
class OwnModel : public QAbstractTableModel {
    Q_OBJECT
public:
    OwnModel(QObject *parent);
};

// tableview.cpp
OwnModel *ownModel = new OwnModel();
QTableView *qTableView = new QTableView();
qTableView->setModel(ownModel)
qTableView.show;

// tablewidget.h
class CustomTable : public QWidget {
    Q_OBJECT
public:
    explicit CustomTable(QWidget *parent = nullptr);
};

// tablewidget.cpp
QTableWidget *tableWidget = new QTableWidget();
tableWidget->setRowCount(itemsCount);

for (int row = 0; row < itemsCount; row++)
    tableWidget->setItem(row, 0, new QTableWidgetItem(items[
   row]);
```

Listing 8.4: Vytvoření widgetu oběma způsoby

8.8 Dostupnost a použití widgetů

Pro vývoj referenční aplikace byla nabídka připravených widgetů dostačují a každý z nich byl použit ke svému účelu. Při jejich výběru nebylo nutné studovat dokumentaci, protože *Qt Creator* je má logicky dělené do skupin a každý z nich má u sebe piktogram zobrazující jeho tvar nebo funkčnost. Tabulka 8.1 zobrazuje využité widgety včetně jejich třídy v Qt a v dalších podkapitolách jsou popsány jen ty funkčně zajímavé.

Komponenta aplikace	**Odpovídající widget**	**Widget v Qt**
horizontální menu	horizontální menu	`QMenuBar`
řádkový vstup/výstup	textový vstup/výstup	`QLineEdit`
tabulka adresář. položek	tabulka	`QTableWidget`
stromová adresář. struktura	strom	`QTreeWidget`
ovládací tlačítka	tlačítko	`QPushButton`
horizontální graf	ukazatel průběhu	`QProgressBar`
terminál	textové pole	`QTextEdit`
modální okno	modální okno	`QDialog`
informativní popisek	popisek	`QLabel`
zobrazení obsahu souboru	scrollovací textové pole	`QTextBrowser`

Tabulka 8.1: Tabulka využitých widgetů v aplikaci

8.8.1 Tabulka QTableWidget

Tabulka je widget odpovídající třídě `QTableWidget` a byl vybrán k popisu, protože v aplikaci představuje hlavní komponentu a jeho výskyt v knihovnách pro tvorbu GUI není samozřejmostí. Mezi programátory je tento widget velmi oblíbený pro svou univerzálnost a jeho absence představuje problémy, které je potřeba řešit úpravou dostupných widgetů nebo vývojem vlastních.

V aplikaci je widget využit pro zobrazení prvků adresáře a umožňuje pohyb mezi adresáři. Dále zobrazuje základní informace o adresářových položkách (název, velikost a poslední modifikace). V aplikaci je tabulka implementována hned dvakrát, aby bylo možné provádět některé druhy operací (přesun a kopírování souborů) nad aktuálně označeným prvkem adresáře. Tabulka může mít řadu dalších funkcionalit, jako je úprava jednotlivých buněk, události vzniklé nad danou buňkou nebo celou řádkou, řazení tabulky podle určitého sloupce atd.

Tabulka byla implementována jako standardní widget, takže data jsou součástí widgetu. Instance třídy `QTableWidget` je rodičovským prvkem pro

všechny prvky tabulky. Prvek tabulky je reprezentován třídou `QTableWidget-Item` a parametry konstruktoru jsou jednotlivé buňky tabulky, resp. jejich hodnoty. Signálem nad aktivní tabulkou je stisk klávesy `ENTER`. Slotem je metoda, která volá funkci pro vstup do nového adresáře s indexem aktuálně označeného řádku tabulky. Při každém signálu jsou volány funkce souborového systému, které připraví data do příslušných struktur a pomocí slotu je widget aktualizován o nová data.

8.8.2 Strom QTreeWidget

Mezi další hlavní komponenty patří stromová struktura poskytovaná třídou `QTreeWidget`. Stejně jako v předchozím případě není přítomnost tohoto widgetu v knihovnách samozřejmostí.

V aplikaci byl widget využit k zobrazení stromové adresářové struktury. Při spuštění aplikace jsou vypsány a načteny jen položky kořenového adresáře – položky jsou reprezentovány třídou `QTreeWidgetItem`. Bylo zde využito líného načítání položek adresáře, protože adresářové struktury mohou být značně rozsáhlé a načítání celé struktury disku by zpomalilo aplikaci při jejím spouštění. Po každém rozbalení adresáře je volána funkce `ls` (Pokud zatím nebyl rozbalen a není již načten ve stromové struktuře.), která připraví obsah adresáře a položky adresáře přiřadí jako *potomky* k *rodiči*, kterým je nadřazený adresář.

Z pohledu datového modelu se také jedná o standardní widget. Instance třídy `QTreeWidget` je rodičovským prvkem pro prvky stromu, které tvoří instance třídy `QTreeWidgetItem`. Prvně vytvořená instance této třídy tvoří kořenový adresář `/`. Adresáře umístěné v kořenovém adresáři jsou potomky nadřazeného adresáře a také reprezentovány třídou `QTreeWidgetItem`. Signálem je kliknutí na určitý prvek stromu a slotem metoda, která zobrazí potomky tohoto adresáře.

8.8.3 Ukazatel průběhu QProgressBar

Qt nabízí řadu grafů a ukazatel průběhu je také jen typem grafu (horizontální). Vůči ostatním grafům je nejjednodušší na implementaci, protože jako jediný z typu grafů je dostupný v nabídce *Qt Designeru*.

V aplikaci se vyskytuje hned dvakrát. V prvním případě zobrazuje obsazenost disku a v druhém obsazenost i-uzlů. Pro správnou funkčnost stačí při vytváření GUI definovat mezní hodnoty a formát zobrazených jednotek. Následně se za pomocí instanční metody nastaví aktuální hodnota grafu.

8.8.4 Textové pole QTextEdit

Posledním zmíněným widgetem je editovatelné textové pole zprostředkované třídou `QTextEdit`. Bylo vybráno pro svou univerzálnost a množství poskytovaných funkcí. Má pokročilé funkce pro nastavení textu (např. druh a řez fontu, velikost a barva) nebo možnost vložení HTML kódu. Dále nabízí nastavení kurzoru, kdy kromě nastavení vizuální podoby je možné nastavovat jeho pozici na základě vzniklé události.

Díky těmto vlastnostem byl widget vybrán pro zobrazení terminálového okna. Do okna se zadávají shellové příkazy zmíněné v podkapitole 7.1. Za jediný nedostatek lze považovat čtení textu z okna, kdy nejsou dostupné metody pro čtení řádku nebo určitého bloku textu a je nutné zadaný příkaz číst za pomocí cyklu znak po znaku.

8.9 Zhodnocení z pohledu programátora

Vývoj z perspektivy programátora bude hodnocen autorem bakalářské práce. Lze tedy předpokládat, že následující text může být mírně subjektivní a závislý na předchozích zkušenostech autora s danými technologiemi.

Programátor by si měl uvědomit, že knihovna Qt je napsána v programovacím jazyce *C++*, jedná se tedy o nativní programovací jazyk pro tuto knihovnu a před samotným vývojem je nutné tento jazyk znát. Lze začít vyvíjet pouze se znalostí programovacího jazyka *C*, ale je vhodné rozumět alespoň základním principům programovacího paradigma OOP (*Object-oriented programming*). Znalost jazyka C a OOP bylo počáteční kvantum znalostí autora. Ze zkušenosti lze však říci, že i při těchto znalostech je užitečné si nastudovat alespoň základní syntaxi a konstrukce jazyka C++. Při tomto postupu lze předejít potenciálním chybám a rychleji pochopit způsob vývoje v této knihovně. Bez úplné znalosti zmíněných věcí je vhodnější začít vyvíjet v jazyce *Python*, který má oficiální podporu. Je možné vyvíjet i v řadě dalších programovacích jazyků, ale ty jsou zaštítěny třetími stranami, takže nelze očekávat plnou podporu z pohledu kvality a rozsahu dokumentace nebo zainteresovaných programátorů na diskuzních fórech.

Po vyřešení věcí kolem programovacího jazyka je možné začít vyvíjet. Bez předchozích zkušeností s vývojem v Qt je velmi přínosné si přečíst článek dostupný na `https://wiki.qt.io/Qt_for_Beginners`. Popisuje základní principy knihovny od hierarchie tříd až po mechanismy událostí a dává programátorovi jistý nadhled. Pro praktický vývoj je užitečné studium několika počátečních kapitol dokumentace, které jsou doplněny o videa a mohou pomoci správně založit nový projekt, lépe se zorientovat v oficiálním IDE

Qt Creator nebo pochopit práci s mechanismy knihovny.

Samotný vývoj lze označit za velmi přívětivý. Veškeré mechanismy jsou dobře popsány v dokumentaci nebo na oficiální wiki, takže jejich použití nepředstavuje žádnou překážku. Kolem knihovny existuje velká aktivní komunita a veškeré problémy lze řešit na oficiálním fóru `https://forum.qt.io`. Ve většině případů hledaný problém již někdo řešil, takže vlákno existuje nebo je řešení dostupné na jiném programátorském fóru, např. na `https://stackoverflow.com`. V případě, že hledaný problém není k nalezení, není nutné ihned zakládat nové vlákno a ptát se na řešení. S největší pravděpodobností problém vznikl nesprávným návrhem aplikace nebo nevhodným použitím mechanismů.

Napojení na backend aplikace nepředstavovalo žádné problémy. Do příslušných souborů jazyka C++ s příponou `.cpp` byly vloženy hlavičkové soubory[3] jazyka C. Backend lze považovat za kvalitně vyvinutý, takže kvůli GUI nepotřeboval žádné úpravy týkající se funkčnosti. Byl jen rozšířen o další struktury, které uchovávají připravená data pro výpis do GUI.

8.10 Zhodnocení z pohledu designera

Při vytváření wireframu zmiňovaného v k podkapitole 7.2 a následně grafického návrhu včetně případů užití není UI/UX designer příliš limitován. Při návrhu desktopové aplikace je vhodné, aby se podíval na existující widgety a jejich nativní vzhled. Každý z widgetů je možné do jisté míry upravit za pomocí jeho vlastností a CSS stylů. Při dodržení těchto zásad lze vytvořit GUI, které bude věrně napodobovat grafický návrh. Je možné navrhnout i nestandardní GUI obsahující složitější grafické prvky a efekty. Při tomto scénáři musí programátor počítat, že nelze použít připravené widgety a rozhraní musí definovat pomocí QML, které je možné generovat pomocí nástroje *Qt Quick*. QML je podobné *JSONu* (*JavaScript Object Notation*) a *JavaScriptu*, a proto má téměř stejné grafické možnosti jako webové aplikace. Tato skutečnost může být zásadní pro designery webových aplikací, protože mobilní nebo desktopová aplikace může mít podobný grafický design i stejné chování za použití programovacího jazyka *JavaScript* jako webová aplikace.

Výsledné GUI referenční aplikace má stejnou podobu jako grafický návrh – widgety mají stejné tvary a je dodržena barevná paleta. Při správném návrhu GUI může designer počítat s responzivním zobrazením, jak již bylo detailněji popsáno v podkapitole 8.6.

[3]Soubory obsahující deklarace tříd, funkcí, konstant, proměnných a identifikátorů.

8.11 Zhodnocení z pohledu ekonoma

Při vývoji softwaru v komerční sféře jsou pečlivě sledovány a následně vyhodnocovány všechny vývojové fáze a jejich časová náročnost neboli ekonomická zátěž. Snížení času jednotlivých fází vývoje může mít pozitivní vliv na výslednou cenu produktu pro klienta a větší zisk pro společnost. Při vývoji rozsáhlejších aplikací a práci v týmu je důležitým aspektem možnost dělení práce. Jedním z důvodů je urychlení vývoje a dodání produktu v kratší časové lhůtě. Druhým důvodem je odlišná kvalifikace lidí (jiné platové třídy) v týmu a s tím související možnosti snížení nákladů na daný projekt. Bylo by velice neekonomické, kdyby senior programátor kódoval GUI pro mobilní aplikaci, když tuto práci může zastat kódér. Každý z nich má jinou sazbu na MD[4] (Někdy je také uváděna sazba v MH[5].), a proto může být fáze vývoje dokončena za určitých podmínek s polovičními náklady.

Qt splňuje veškeré předpoklady, aby bylo ekonomicky přínosné. Vývoj aplikace probíhá za pomoci programovacího paradigma OOP. Kromě programátorských výhod (objekty, abstrakce, zapouzdření, polymorfismus, ...) podporuje metodu „rozděl a panuj“, kdy jsou problémy většího charakteru děleny na podproblémy. Tyto podproblémy neboli dílčí části jsou pak efektivněji řešeny v rámci týmu a je možné delegovat práci na základě kvalifikace člověka. Menší MD nebo MH sazbu mají obecně kodéři vytvářející GUI a větší programátoři vyvíjející jádro aplikace. Rozdělení práce definuje také samotná softwarová architektura *Model/View*, kdy model je aplikačním objektem a pohled jeho reprezentací na obrazovce [39].

Pro ekonoma je dalším důležitým aspektem, že Qt patří mezi nejpoužívanější knihovny a je využíváno napříč průmyslovými odvětvími. Své zastoupení má např. v automobilovém průmyslu, kde je využíváno značkou *Mercedes-Benz* pro GUI obrazovek svých informačních systémů nebo v lékařském odvětví. Tato informace může být určitou zárukou z pohledu ekonomických hledisek. Společnost *Forrester Consulting* provedla studii, ve které zkoumala kvantifikované a nekvantifikované hodnoty od komerčních zákazníků. Výsledkem studie je, že společnost používající Qt může ušetřit 30 % nákladů na vývoj a až 80 % nákladů na hardware. Při vývoji embedded zařízení je primární udržet cenu za hardwarové komponenty na minimu, aby výsledné zařízení mohlo nabízet konkurenční cenu. Pomocí Qt lze vyvinout velmi efektivní a výkoný software, který bude mít minimální nároky

[4]Man day – V překladu člověko-den je pracovní čas jedné osoby, který odpovídá jednomu pracovnímu dni. V ČR tuto hodnotu lze zobecnit na 8 hodin.

[5]Man hour – Člověko-hodina je občas používaná místo MD, protože nutně nemusí představovat 60 minut.

na výkon hardwaru. Organizace tímto způsobem získá stejně výkonné zařízení s levnější hardwarem. Méně výkonný hardware nebo jeho menší velikost přináší i další výhody. Menší hardware šetří prostorové kapacity organizace a méně výkonný hardware vyžaduje méně elektrické energie, a proto vyzařuje i méně tepla. Úspora 80 % za hardwarové vybavení platí pro organizace přecházející ze softwaru založeném na webových technologiích. Naopak pro organizace používající průmyslové panely vyvinuté pomocí nativních vývojových nástrojů může být úspora jen 10 % [22]. Ekonoma bude zajímat ukazatel ROI (*Return on Investment*) vyjadřující návratnost investice, resp. poměr mezi ziskem a vynaloženými náklady. Pro odhad ROI před vývojem je možné využít tuto kalkulačku `https://www.qt.io/roi-calculator`. Po zadání parametrů vývoje je během chvíle vygenerován report a zaslán na e-mail. Report kromě ROI zobrazuje další benefity a možnosti snížení nákladů.

8.12 Celkové zhodnocení

Při vývoji referenční aplikace v Qt nedocházelo k větším komplikacím a výsledné GUI je téměř totožné s návrhem. Hodnocení z různých perspektiv lze považovat za velmi kladná, a proto může být Qt přínosné pro společnosti i jednotlivce. Společnost může vyvíjet kvalitní a optimalizované multiplatformní aplikace s vysokým ekonomickým potenciálem. Programátor jako jednotlivec získává mocný nástroj v podobě ekosystému, které Qt nabízí. K dispozici je kvalitní IDE včetně přidružených nástrojů a kvalitně zpracovaná dokumentace. Pokud se programátor naučí pracovat s touto knihovnou, může sám vyvíjet kvalitní software nebo se bez problému uplatnit na trhu práce.

9 Vývoj v knihovně NCurses

Následující text o vývoji v NCurses již detailně nepřibližuje vždy dané téma, ale odráží spíše rozdíly oproti vývoji v Qt. Základním předpokladem pro vývoj v NCurses je znalost programovacího jazyka *C*. Bez této znalosti nelze vyvíjet GUI, protože i jeho definování probíhá za pomocí tohoto jazyka.

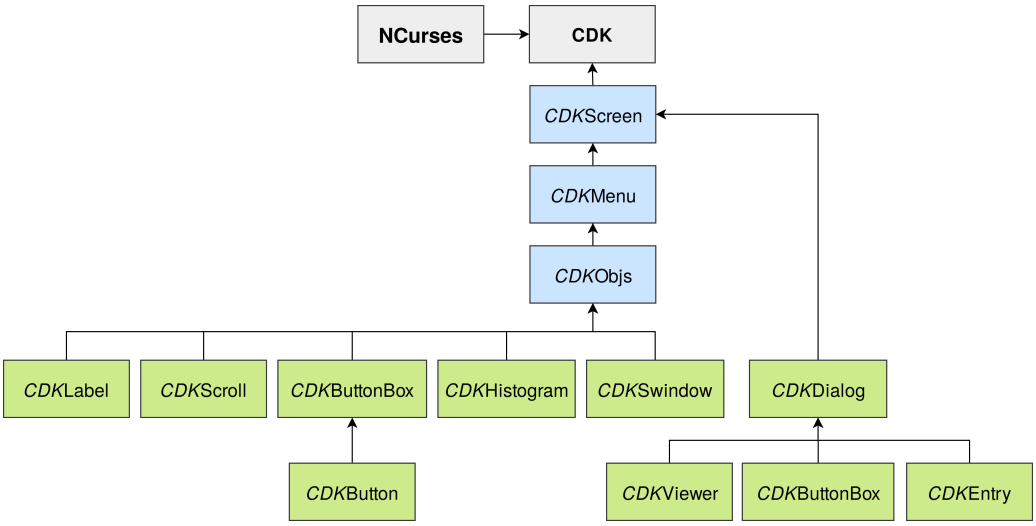
Znázornění aplikace pomocí UML diagramu je oproti Qt velmi zjednodušeno, protože jazyk C je strukturální a nemá mechanismy programovacího paradigma OOP. Fáze životního cyklu jsou v podobném rozsahu. Další podkapitoly jsou jisté míry totožné s Qt, aby bylo možné udělat přímé srovnání výhod a nevýhod obou knihoven. Pro porozumění dalších kapitol je nutné uvést, že pro vývoj GUI byl využit toolkit[1] CDK (Curses Development Kit), protože samotné NCurses žádné widgety neobsahuje – má jen knihovny pro tvorbu menu, formulářů a manipulaci s okny. Následující texty budou věnovány primárně CDK.

9.1 Hierarchie widgetů v aplikaci

Na UML diagramu 8.2 jsou uvedeny všechny implementované widgety v rámci aplikace. Na rozdíl od UML diagramu 8.2 u knihovny Qt zde boxy nereprezentují *třídu* ale *strukturu*[2]. Na vrchu celé hierarchie je právě knihovna *NCurses*, díky které je možné přeložit toolkit *CDK*. V modrém boxu je zmíněn `CDKScreen`, který tvoří „plátno“ pro widgety a slouží pro jejich manipulaci. Dále je zde `CDKMenu` sloužící pro rozložení jednotlivých widgetů. Poslední strukturou je `CDKObjs`, pomocí které je vytvořeno pole sdružující všechny widgety zobrazené za pomocí `CDKMenu`. V zelených boxech jsou již widgety viditelné v GUI aplikace. Hierarchicky níže pod `CDKMenu` jsou struktury `CDKLabel`, `CDKScroll` a `CDKButtonBox`, které tvoří widgety zobrazující se při spuštění aplikace. Struktura `CDKDialog` slouží pro vytvoření modálního okna pro všechny případy užití uvedené v podkapitole 7.2. Vzájemnou kombinací struktur `CDKViewer`, `CDKButtonBox` a `CDKEntry` jsou vytvořeny všechny typy modálních oken.

[1] Sada základních stavebních jednotek (widgetů) pro vývoj GUI.

[2] Při programování v jazyce C je strukura deklarace nového složeného datového typu obalující další proměnné různých datových typů.

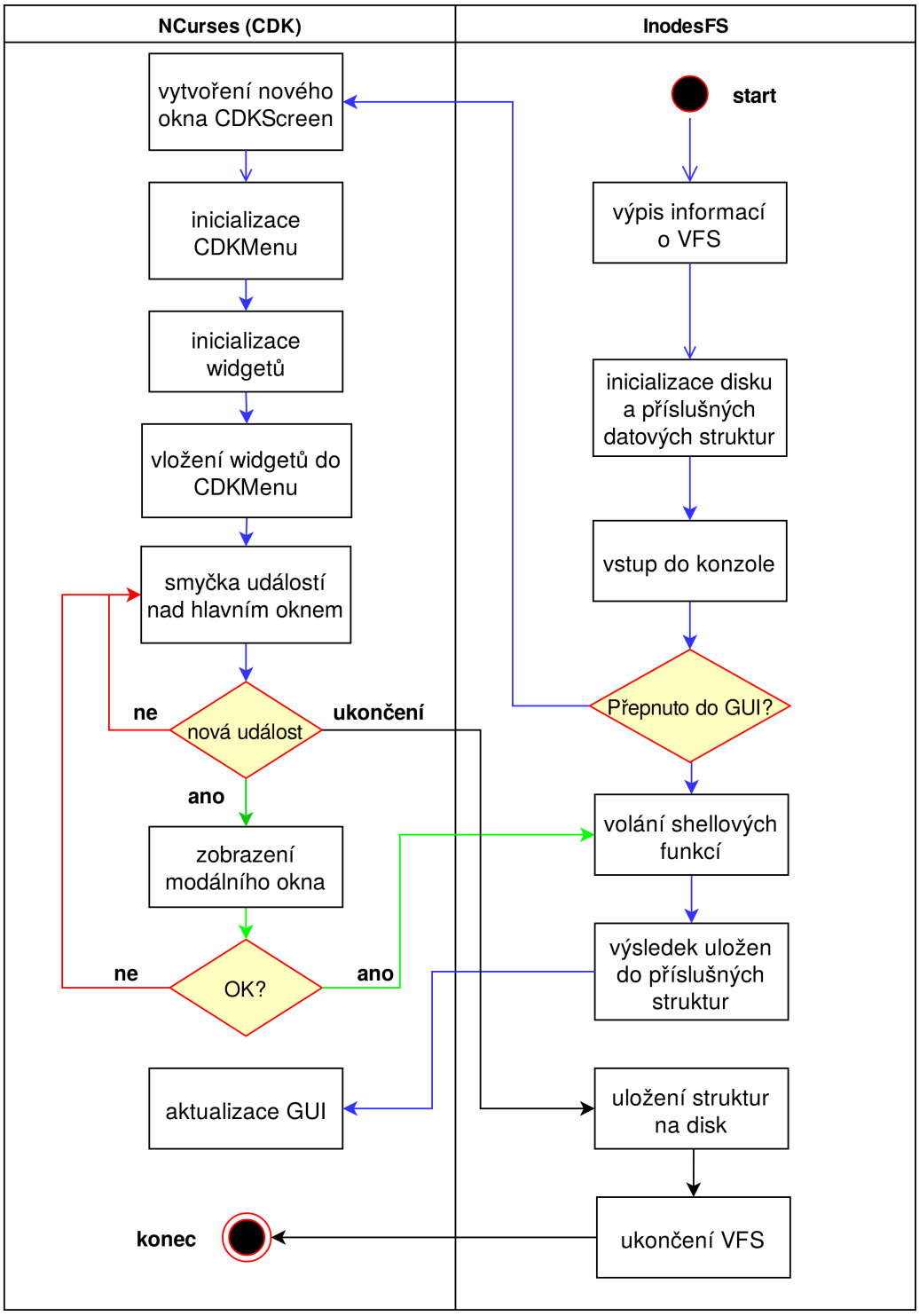


Obrázek 9.1: Hierarchie struktur widgetů v toolkitu CDK implementovaných v aplikaci

9.2 Životní cyklus aplikace

Diagram 9.2 je stejně jako u předchozí knihovny rozdělen na dvě části. Backend aplikace tvoří opět VFS *InodesFS* a frontend *NCurses (CDK)*.

Po spuštění aplikace jsou do konzole nejdříve vypsány základní informace o VFS a následně je inicializován virtuální disk a příslušné datové struktury. Nyní má uživatel možnost ovládat aplikaci skrze konzoli nebo může pomocí příkazu `gui` přepnout do grafického režimu – to je hlavní odlišnost oproti předchozí aplikaci. V případě přepnutí do GUI dojde k vytvoření nového okna `CDKScreen` a následné inicializaci `CDKMenu`, které slouží pro rozvržení widgetů. Dále dochází k inicializaci widgetů a jejich vložení do `CDKMenu`. Po inicializaci všech prvků rozhraní je zavolána funkce `traverseCDKScreen-(CDKScreen *screen)` umožňující pohyb mezi widgety. Kromě ovládání aktivního widgetu je možné pomocí funkčních tlačítek vytvářet nové události. Při stisku některého z funkčních tlačítek dojde k zobrazení modálního okna `CDKDialog`. Uživatel poté zadává své vstupy a při kladném potvrzení dochází ke zpracování a opakuje se stejný postup jako v podkapitole 8.3. Jedinou změnou je možnost přepnutí do textového režimu a pokračovat v ovládání aplikace skrze konzoli. Dochází jen ke skrytí GUI a při opětovném návratu do GUI je načten předchozí stav (např. pracovní adresáře), protože při přepínání mezi režimy nedochází k dealokaci struktur.



Obrázek 9.2: Životní cyklus aplikace v NCurses (CDK)

9.3 Obsluha událostí

Pro CDK neexistují sofistikované mechanismy pro obsluhu událostí. Každý widget má pouze své předdefinované chování a další chování se musí naprogramovat.

Za pomocí CDK lze definovat vazby kláves nebo klávesových kombinací. Díky této schopnosti je CDK dynamičtější a použitelnější pro širokou škálu možných událostí. Důležitou funkcí pro vytváření vazeb je `bindCDKObject(...)`. Na ukázce kódu 9.1, která byla pro názornost zjednodušena, nejdříve funkce `make_scroll(...)` obaluje funkcionalitu pro vytvoření widgetu scrollovacího seznamu. Widget je vytvořen pomocí funkce `newCDKScroll(...)`. Následně funkce `bindCDKObject(...)` vytvoří vazbu na klávesu `ENTER`, ale vazeb může být vytvořeno i více. Při každém stisku této klávesy je volána funkce `cd_scroll(void *object)` zajišťující přechod do jiného adresáře. Po úspěšném vstoupení do nového adresáře je překreslen widget.

Při vytváření vazby na klávesu se v tomto případě vkládá do funkce `cd_scroll` jako parametr ukazatel na widget. Do posledního parametru funkce je možné vložit další data nespecifikovaného datového typu `void`.

```
1  // layout.c
2  static CDKOBJS *make_scroll(int x, int y) {
3      // vytvoreni noveho widgetu
4      CDKSCROLL *scroll = newCDKScroll(...);
5      // vytvoreni vazby na klavesu enter
6      bindCDKObject(vSCROLL,scroll,KEY_ENTER,cd_scroll,NULL);
7
8      return ObjPtr(scroll);
9  }
10
11 static int cd_scroll(void *object) {
12     CDKSCROLL *scroll = (CDKSCROLL *) object;
13     // ziskani ID aktivniho adresaroveho prvku
14     int item_id = getCDKScrollCurrentItem(scroll);
15     list_dir_item *item = list_dir_items[item_id];
16
17     // nastaveni noveho obsahu do widgetu
18     setCDKScroll(scroll,list,list_size,FALSE,A_REVERSE,TRUE);
19     // aktualizace widgetu
20     refreshCDKScreen(ScreenOf(scroll));
21
22     return TRUE;
23 }
```

Listing 9.1: Obsluha události po stisku klávesy

9.4 Vytvoření uživatelského rozhraní

Pro vytváření GUI neexistuje žádný nástroj, který by generoval XML či jiný značkovací jazyk definující rozhraní. NCurses nelze kompilovat s připojeným souborem obsahující značkovací jazyk – to je hlavním důvodem neexistence žádného nástroje. Jedinou možností vytvoření GUI je volání funkcí pro vykreslování widgetů a ukládání dat do příslušných struktur.

Další stěžejní věcí je absence kontejnerů pro rozložení widgetů, protože to nepřináší jen problémy s pozicováním prvků v rozhraní, ale váže se s tím i nemožnost přecházet mezi jednotlivými widgety, resp. možnost určení toho aktivního. Po důkladné analýze dostupných mechanismů a zveřejněných příkladů se během vývoje došlo k závěru, že widget `CDKMenu` může do jisté míry zastoupit kontejner pro rozložení prvků v GUI. Místo prvků menu byly vloženy vytvořené widgety. Po inicializace všech položek menu je volána funkce `traverseCDKScreen(CDKScreen *screen)`, díky které může uživatel mezi widgety přecházet za použití klávesy `TAB`.

9.5 Responzivita uživatelského rozhraní

NCurses má dvě proměnné, díky kterým lze dosáhnout responzivního zobrazení. Proměnná `COLS` udává počet sloupců a `LINES` naopak počet řádků daného terminálu. Každý widget má parametr pro horizontální a vertikální umístění. Horizontální umístění může být nastaveno pomocí konstant `LEFT`, `CENTER` a `RIGHT` a horizontální pomocí `BOTTOM`, `HORIZONTAL` a `VERTICAL`. Do parametrů lze zadat i konkrétní číslo sloupce a řádku.

Chování samotného widgetu již nelze žádným způsobem ovlivnit. Pozitivním je, že každý widget má své předefinové responzivní chování, takže se do jisté míry přizpůsobuje velikosti okna. Problém vzniká při změně velikosti okna a již načtené aplikaci. Rozhraní nereaguje na změny a jeho prvky si zachovávají velikost jako při spuštění aplikace. Při vývoji referenční aplikace byl tento problém testován na vzorových programech, ale se stejným negativním výsledkem – jedná se tedy o problém CDK spjatý s jeho začátkem vývoje v polovině 90. let minulého století. Základ tookitu se v novém tisíciletí již tolik neměnil a řešily se především chyby a rozšíření. Možností by bylo vyvíjet aplikace vícevláknově, kdy by jedno vlákno volalo funkci `refreshCDKSCreen(CDKScreen *screen)` pro překreslení celého okna nebo překreslovací funkce samotných widgetů.

9.6 Dostupnost a použití widgetů

V úvodní kapitole 9 této knihovny byla popsána absence widgetů a výběr toolkitu. Autor knihovny *Thomas E. Dickey* uvádí v dokumentaci dva toolkity, u kterých je sám spoluautorem. Prvním z nich je již zmiňovaný CDK a druhým je *Dialog*. CDK představuje plnohodnotný toolkit, jehož vývoj je stále aktivní – v době psaní bakalářské práce proběhla poslední aktualizace 28. 2. 2020. Knihovnu Dialog lze také považovat za toolkit, ale je to spíše kompletní aplikace zobrazující různé modifikace dialogových oken, protože jednotlivé widgety jsou objekty, u kterých můžeme za pomocí parametrů upravovat vzhled a chování. Ve spojení s *bashem*[3] může být použit jako průvodce instalací softwaru. Poslední aktualizace proběhla 30. 12. 2012.

V tabulce 9.1 jsou zobrazeny použité widgety z toolkitu CDK. Podrobněji budou popsány jen komponenty, pro které neexistoval odpovídající widget nebo byly zajímavé z hlediska funkčnosti.

Komponenta aplikace	**Odpovídající widget**	**Widget v CDK**
horizontální menu	horizontální menu	`CDKMenu`
řádkový vstup	vstup	`CDKEntry`
tabulka adresář. položek	tabulka	`CDKScroll`
stromová adresář. struktura	strom	-
ovládací tlačítka	tlačítko	`CDKButton`
horizontální graf	ukazatel průběhu	`CDKHistogram`
terminál	textové pole	`CDKSWindow`
modální okno	modální okno	`CDKDialog`
informativní popisek	popisek	`QLabel`
zobrazení obsahu souboru	scrollovací textové pole	`CDKViewer`

Tabulka 9.1: Tabulka využitých widgetů v aplikaci

9.6.1 Tabulka CDKScroll

Pro zobrazení adresářových položek je optimální využít tabulku. V toolkitu CDK není widget reprezentující tabulku, a proto byl využit `CDKScroll`. Ten sice představuje jen scrollovací seznam položek, ale za pomocí vhodného formátování výpisu položek a hlavičky seznamu mohl být použit pro zobrazení a práci s adresářovými položkami.

[3]Jeden z nejpoužívanějších unixových shellů.

9.6.2 Stromová adresářová struktura

Pro zobrazení stromové struktury neexistuje v toolkitu odpovídající widget ani jeho případná náhrada.

Řešením by bylo použít stromovou strukturu z jiného toolkitu pro knihovnu NCurses – *Dialog* má ve své nabídce widget `TreeView`. Dalším řešením je použití hotového widgetu z existující aplikace s otevřeným zdrojovým kódem využívající knihovnu NCurses. Správce souborů *Midnight Commander* tuto podmínku splňuje a obsahuje widget pro zobrazení stromové struktury. U aplikace není využit toolkit třetí strany, ale má vyvinuté vlastní widgety. V obou případech však nastává problém s implementací do referenční aplikace, protože Dialog a Midnight Commander mají vlastní ekosystém pro widgety a použití jednoho widgetu by znamenalo přenést i jejich závislosti.

Z pohledu časové náročnosti lze po zkušenostech s vývojem konstatovat, že by bylo přijatelnější vytvořit nový widget v toolkitu CDK nebo využít pro vývoj jinou knihovnu, popř. toolkit.

9.6.3 Textové pole CDKSwindow

Při vývoji referenční aplikace byl vytvořen mechanismus na přecházení mezi textovým a grafickým režimem, a tím zůstala zachována 100% funkcionalita shellových příkazů. Widget `CDKSWindow` představující textové okno s horizontálním posuvníkem byl použit z důvodu, aby GUI bylo více autentické s grafickým návrhem.

9.6.4 Ukazatel průběhu CDKHistogram

Obsazenost disku a i-uzlů je zobrazena pomocí widgetu `CDKHistogram`. Jeho incializace není složitá a po nastavení jednotek, mezních a aktuálních hodnot je plně funkční. Nejedná se ale o jediný widget pro zobrazení grafu. Dostupný je ještě widget `CDKGraph`, který zobrazuje průběh dané funkce na ose x a y.

9.6.5 Scrollovací textové pole CDKViewer

`CDKViewer` není jen scrollovací textové pole, ale je přímo určen pro prohlížení obsahu souborů. Prohlížení obsahu souboru je zprostředkováno za pomocí modálního okna. V jeho záhlaví se kromě názvu souboru nachází celkový počet řádek, aktuální řádka a pozice souboru v procentuálním vyjádření. V zápatí okna je tlačítko `OK` pro ukončení modálního okna.

9.7 Datový model widgetů

CDK nepoužívá žádný návrhový vzor, který by odděloval logickou část kódu od grafického zobrazení widgetu. Na ukázce kódu 9.1 je vidět, že jedním z parametrů funkce pro inicializaci nového widgetu jsou data. Pro aktualizaci dat jsou volány příslušné funkce nastavující nové hodnoty widgetu. Funkce mají kromě parametru pro nová data také parametr ukazatele na strukturu stávajícího widgetu. Po nastavení nových dat je překreslen celý widget, popř. celá obrazovka aplikace a nová data jsou zobrazena.

9.8 Zhodnocení z pohledu programátora

Knihovna NCurses je napsána v programovacím jazyce *C*. Existují vazby i na další programovací jazyky, ale stejně jako u předchozí knihovny bylo preferencí programovat v nativním jazyku knihovny. Dalším důvodem byla autorova zkušenost s tímto programovacím jazykem a backend napsaný v totožném jazyce. Pokud se programátor rozhodne pro výběr stejného jazyka, musí počítat s tím, že se jedná o nízkoúrovňový jazyk a jeho neznalost by při vývoji přinášela problémy.

Po překonání této bariéry je možné začít s vývojem. Hned na začátku je programátor postaven před otázku ohledně výběru vhodného toolkitu z důvodu popsaného v podkapitole 9.6. Po výběru toolkitu se může pustit do samotného vývoje aplikace. Jako zdroj informací slouží oficiální dokumentace k toolkitu `invisible-island.net/cdk/manpage/` a NCurses `tldp.org/HOWTO/NCURSES-Programming-HOWTO/`, popř. ještě neoficiální dokumentace `invisible-island.net/ncurses/man/`. Neoficiální dokumentace pro NCurses je ze všech zmíněných nejkvalitněji zpracována a kromě popisu mechanismů a funkcí obsahuje i vzorové příklady – oficiální dokumentace jsou strohé a chybí jim vzorové příklady. Jako optimální řešení pro pochopení implementace widgetů se ukázalo studium vzorových programů přibalených k toolkitu. Důležité části kódu jsou komentovány a zbytek informací lze dohledat v některé z dokumentací. Při vývoji musí programátor neustále kontrolovat úniky paměti (např. pomocí programu *Valgrind*), aby nedocházelo k náhodným pádům GUI při používání aplikace.

Za největší problém lze označit malou komunitu kolem toolkitu, resp. kolem celé knihovny NCurses. Neexistuje žádná wiki ani fórum zabývající se touto problematikou. Při hledání vzniklých problémů nelze nalézt relevantní vlákna ani na známých programátorských fórech.

Alternativní cestou pro vývoj by bylo použití programovacího jazyka *Python*, který má modul `curses` poskytující rozhraní pro knihovnu NCurses

[51]. Pro Python existuje toolkit *Urwid*, který odstraňuje některé nevýhody CDK. Obsahuje kontejnery pro rozložení widgetů, takže lze lépe řešit responzivitu aplikace a má integrovanou podporu myši [13]. Pro některé programátory bude programování v Pythonu přijatelnější a díky vzestupu tohoto programovacího jazyka je kolem toolkitu Urwid větší komunita. Nevýhodou toolkitu je velmi omezená nabídka widgetů, i když přímo toolkit umožňuje vytvářet vlastní. Za další nevýhodu lze považovat samotný vysokoúrovňový jazyk Python, který může být naopak pro některé aplikace nevhodný.

9.9 Zhodnocení z pohledu designera

UI/UX designer je v tomto případě limitován ve všech směrech. Může pracovat jen s omezenou paletou barev, musí pracovat jen s jednoduchými tvary a použití různých efektů nebo animací je téměř vyloučeno. Většina widgetů má předefinovaný vzhled, který již nelze příliš modifikovat. S těmito omezeními musí designer při návrhu počítat, a proto je před vývojem aplikace vhodné zhodnotit nutnost grafického návrhu. Ve většině případů by byl dostačující wireframe (viz podkapitola 7.2), aby se v GUI zamezilo UX chybám.

Výsledné GUI referenční aplikace odpovídá definovanému rozložení, ale widgety nemají shodný tvar a není dodržena barevná paleta. Designer může do jisté míry počítat s responzivním zobrazením, i když s mnoha omezeními.

9.10 Zhodnocení z pohledu ekonoma

Knihovna NCurses ve spojení s libovolným toolkitem nemá potenciál být ekonomicky zajímavá hned z několika důvodů. Prvním důvodem je poměrně malý trh lidí, kteří by chtěli využívat aplikace v konzolovém prostředí – GUI není z pohledu UX moc přívětivé a jeho používání vyžaduje zkušenějšího uživatele. S tímto aspektem se váže i podpora operačních systémů, která je jen pro systémy založené na linuxovém jádře a několik dalších minoritně používaných, viz podkapitola 4.8.1. Linuxové distribuce tvoří dle dostupných dat jednotky procent. V prosinci 2016 tvořil podíl na trhu 3,8 %, ale v průběhu dalších let se snižoval a v únoru 2020 měl podíl 3,34 %. V době psaní bakalářské práce se nenachází v tabulce deseti nejpoužívanějších operačních systémů [46]. Druhým důvodem může být drahý vývoj, protože knihovna nemá žádnou softwarovou architekturu oddělující aplikační objekt od reprezentace na obrazovce. Veškerou práci musí odvést programátor a nelze snížit náklady delegováním práce na kodéra s nižší sazbou na MD.

Knihovna NCurses je svobodným softwarem jako součást projektu GNU

[53]. GNU má řadu licencí upravující používání a distribuci softwaru a lze aplikaci vniklou pod GNU využít ke komerčním účelům. Mezi uživateli Linuxu jsou aplikace postavené na této knihovně stále využívány a díky tomu i jejich vývoj je aktivní. Mezi oblíbené a stále využívané aplikace patří např. správce souborů *Midnight Commander*, pokročilý textový editor *Vim*, správce procesů *htop* a mnoho dalších. Většina těchto programů je dostupných zdarma, takže by aplikace musela být opravdu přínosná, aby za ni komunita byla ochotna platit.

9.11 Celkové zhodnocení

Vývoj v knihovně NCurses, resp. CDK nelze označit z pohledu programátora za přívětivý. Při vývoji referenční aplikace docházelo ke komplikacím, ke kterým nebylo možné dohledat řešení a ani jedna z dokumentací nebyla příliš nápomocná – jediným řešením programátora je spolehnout se sám na sebe a zkoušet... Rozložení GUI ve srovnání s wireframem lze označit za totožné i přes absenci widgetu reprezentující stromovou strukturu. Při srovnání s referenčním grafickým návrhem není dodržena barevná paleta a vzhled widgetů. Z ekonomického pohledu nemá knihovna příliš velký potenciál pro společnosti ani jednotlivce.

Výhodu by knihovna mohla přinášet pro čistě konzolové aplikace, které by staly „hybridními“ z pohledu GUI. Z výše uvedených tvrzení vyplývá, že vývoj komplexních aplikací je poměrně složitý a výsledný produkt není z pohledu GUI bezchybný. V referenční aplikaci je vytvořen mechanismus pro přepínání mezi textovým a grafickým režimem – tuto funkcionality by mohly využívat všechny konzolové aplikace pro určité případy užití. Např. v souvislosti se souborovým systémem by se při výběru kopírovaného souboru zobrazil widget pro výběr souboru z lokálního úložiště a uživatel by nemusel zadávat jeho absolutní cestu. Po dokončení kopírování by se aplikace vrátila zpět do textového režimu a do grafického by se přepnula při dalším případu užití s příslušným widgetem.

9.12 Přímé srovnání obou knihoven

Obě testované knihovny byly proti sobě srovnány v několika aspektech. Následující tabulka 9.2 porovnává měřitelné hodnoty a klíčové vlastnosti. Zajímavým údajem je *časová náročnost vývoje*, která je téměř totožná – do tohoto údaje je do jisté míry promítnuto studium dané knihovny. Dále bylo měřeno celkové vytížení CPU (*Central Processing Unit*) na notebooku *Dell La-*

titude 7480 (Intel Core i5-7300U, 8GB RAM DDR4, Intel HD Graphics 620 a 256 GB SSD) s operačním systémem *Debian GNU/Linux 9* a měření bylo prováděno pomocí nástroje *htop*. V několika iteracích byl opakován scénář, kdy byl daný program spuštěn a následně vyhledán soubor a jeho obsah zobrazen. Cílem bylo zamezit výkonnostně náročným operacím nad diskem, aby se projevilo jen vytížení CPU způsobené vykreslováním GUI. Byla také testována komunikace s aplikací přes síť, resp. přes SSH (*Secure Shell)*. Aplikace vyvíjená v *NCurses (CDK)* běží čistě v terminálu, a proto i skrze SSH je zajištěna její funkčnost včetně zobrazení GUI. Funkčnost aplikace byla otestována na školním serveru `ares.fav.zcu.cz`. U aplikace napsané v *Qt* nebylo možné nic testovat, protože GUI nelze zobrazit skrze terminál. Vytvořením instance třídy `QProcess` může být aplikace napsaná v Qt ovládána skrze terminál, protože má předefinované výstupní kanály `stdout` a `stderr` [9].

Vlastnost	**Qt**	**NCurses (CDK)**
shodnost s wireframem	ano	ano
shodnost s návrhem	ano	ne
splnění případů užití	7/7	7/7
responzivita	ano	ano (nereaguje na resize)
dostupnost widgetů	10/10	9/10
časová náročnost vývoje	107 hod.	101 hod.
velikost binárního souboru	4,7 MiB	602,5 KiB
vytížení CPU	3,9 %	0,7 %
komunikace přes síť (SSH)	ano (`QProcess`)	ano

Tabulka 9.2: Srovnání knihoven v měřitelných hodnotách a klíčových vlastnostech

10 Závěr

Cílem práce bylo seznámit se s problematikou uživatelských rozhraní (UI) a použitelnosti (UX). Nejdříve byl uskutečněn teoretický rozbor, který se zabývá uživatelskými rozhraními (UI) včetně všech jejich typů. Stejným způsobem byl zpracován uživatelský prožitek (UX) a jak ho ovlivňuje psychologie, použitelnost, design a copywriting.

Po úvodu do UI a UX proběhla analýza existujících a používaných knihoven pro tvorbu GUI. U vybraných knihoven byly shrnuty nejdůležitější věci se zaměřením na podporované operační systémy, komunitou kolem knihovny, licence a podporované programovací jazyky. Následně byla provedena analýza prvků (widgetů) uživatelských rozhraní a paralelně vznikla aplikace, jejíž GUI obsahuje téměř všechny popisované widgety, aby měl čtenář lepší představu o jejich funkčnosti a odlišnostech.

Dalším krokem bylo porovnání vybraných knihoven s ohledem na náročnost jejich použití i s ohledem na tvorbu přístupných a použitelných GUI. Po konzultaci s vedoucím bakalářské práce došlo k výběru dvou odlišných knihoven. Bylo vybráno *Qt* jako zástupce moderní knihovny pro vývoj multiplatformních aplikací napříč operačními systémy a obsahující velké množství widgetů. Protipólem se stalo *NCurses* jako zástupce knihovny pro tvorbu aplikací s textovým rozhraním (terminálové aplikace).

Ve vybraných knihovnách vznikla aplikace pro správu souborů a jako základ byla využita semestrální práce z předmětu KIV/ZOS, jejímž tématem byl *zjednodušený souborový systém založený na i-uzlech*. Při vývoji referenční aplikace byl kladen důraz na samotnou problematiku UI a UX. Nejdříve vznikl wireframe a na základě něho grafický návrh aplikace včetně všech případů užití. Po schválení grafického návrhu došlo ke značnému rozšíření semestrální práce pro potřeby grafického návrhu a následně vznikla v každé z vybraných knihoven aplikace.

U vzniklých aplikací došlo k porovnání jejich vlastností. Nejdříve byla analyzována aplikace vzniklá v Qt. U aplikace je např. popisován životní cyklus, obsluha událostí, vytvoření GUI nebo dostupnost a použití widgetů. Nakonec je celý vývoj hodnocen z pohledu programátora, UI/UX designera a ekonoma. U druhé aplikace vzniklé v knihovně *NCurses* již nejsou daná témata detailně přibližována, ale odráží rozdíly oproti vývoji v Qt. Nakonec byly obě knihovny porovnány v měřitelných hodnotách a klíčových vlastnostech. Vzniklé aplikace jsou plně funkční a splňují definované případy užití.

Přehled zkratek

UI	User Interface
UX	User Experience
TUI	Textual User Interface
CLI	Command-line Interface
GUI	Graphical User Interface
BIOS	Basic Input-Output System
TUI	Textual User Interface
UEFI	Unified Extensible Firmware Interface
PARC	Palo Alto Research Center
MOC	Meta-Object Compiler
GPL	General Public License
LGPL	Library General Public License
TUI	Textual User Interface
QML	Qt Modeling Language
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JRE	Java Runtime Environment
JDK	Java Development Kit
XML	Extensible Markup Language
GNU	GNU's Not Unix!
BSD	Berkeley Software Distribution
SQL	Structured Query Language
IDE	Integrated Development Environment
VR	Virtual Reality
AR	Augmented Reality
BSL	Boost Software License
API	Application Programming Interface
GDI	Graphics Device Interface
WPF	Windows Presentation Foundation
UWP	Universal Windows Platform
DPI	Dots Per Inch
DIP	Device Independet Pixel
XAML	Extensible Application Markup Language
UFS	Unix File System
VFS	Virtual File System
UML	Unified Modeling Language

MVC	Model-view-controller
MV	Model/View
OOP	Object-oriented programming
JSON	JavaScript Object Notation
MD	Man-day
MH	Man-hour
ROI	Return on Investment
CDK	Curses Development Kit
CPU	Central Processing Unit
SSH	Secure Shell

Literatura

[1] *Nana C++ Library - a modern C++ GUI library* [online]. 2019. [cit. 2019-12-29]. Dostupné z: http://nanapro.org/en-us/.

[2] *A/B Testing* [online]. Optimizely, c2020. [cit. 2020-03-24]. Dostupné z: https://www.optimizely.com/optimization-glossary/ab-testing/.

[3] *Language Bindings* [online]. The GTK Team, c2017–2019. [cit. 2019-10-12]. Dostupné z: https://www.gtk.org/language-bindings.php.

[4] *.NET Framework system requirements* [online]. Microsoft, 2019. [cit. 2019-10-12]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/get-started/system-requirements.

[5] *Ultimate++ team* [online]. c1998–2019. [cit. 2019-10-19]. Dostupné z: https://www.ultimatepp.org/appideAbout$en-us.html.

[6] *Experienced programmer, but new to Unity? You're already ahead of the game.* [online]. c2019. [cit. 2019-10-20]. Dostupné z: https://unity3d.com/programming-in-unity.

[7] *The Event System* [online]. The Qt Company, c2016. [cit. 2020-03-12]. Dostupné z: https://doc.qt.io/archives/qt-4.8/eventsandfilters.html.

[8] *Releases* [online]. Red Hat, Inc. [cit. 2019-11-03]. Dostupné z: https://pagure.io/newt/releases.

[9] *QProcess Class* [online]. The Qt Company, c2020. [cit. 2020-04-25]. Dostupné z: https://doc.qt.io/qt-5/qprocess.html.

[10] *What is GTK, and how can I use it?* [online]. The GTK Team, 2017–2019. [cit. 2019-10-11]. Dostupné z: https://www.gtk.org/.

[11] *Getting Started (WPF)* [online]. Microsoft, 2018. [cit. 2019-10-12]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/.

[12] *Úvod do WPF (Windows Presentation Foundation)* [online]. ITnetwork.cz, 2015. [cit. 2019-12-30]. Dostupné z: https://www.itnetwork.cz/csharp/formulare/wpf/c-sharp-tutorial-wpf-uvod-a-prvni-formularova-aplikace.

[13] *Widgets* [online]. c2014. [cit. 2020-04-11]. Dostupné z: http://urwid.org/manual/index.html.

[14] *Unity 2019: Performance by default, high-fidelity real-time graphics, and artist tools* [online]. Unity Technologies, 2019. [cit. 2019-10-20]. Dostupné z: https://unity3d.com/unity/.

[15] *Welcome to the Tcl Developer Xchange!* [online]. Tcl Developer Xchange, 2019. [cit. 2019-10-19]. Dostupné z: https://www.tcl.tk/.

[16] *Language Bindings* [online]. The Qt Company, 2019. [cit. 2019-10-08]. Dostupné z: https://wiki.qt.io/Language_Bindings.

[17] *Overview* [online]. WxWidgets, 2019. [cit. 2019-10-11]. Dostupné z: https://www.wxwidgets.org/about/.

[18] *What is ASCII Art?* [online]. Computer Hope, 2019. [cit. 2019-12-14]. Dostupné z: https://www.computerhope.com/jargon/a/asciiart.htm.

[19] *Choose your app platform* [online]. Microsoft, 2019. [cit. 2019-12-30]. Dostupné z: https://docs.microsoft.com/en-us/windows/apps/desktop/choose-your-platform.

[20] *What is Gtk#?* [online]. Mono Project, c2019. [cit. 2019-12-29]. Dostupné z: https://www.mono-project.com/docs/gui/gtksharp/.

[21] *NCurses* [online]. 2017. [cit. 2019-11-02]. Dostupné z: https://www.sallyx.org/sally/c/linux/ncurses.

[22] *White paper: Forrester Study | The Total Economic Impact^{TM} of Qt for Device Creation* [online]. The Qt Company, 2018. [cit. 2020-04-06]. Dostupné z: https://resources.qt.io/whitepaper/white-paper-forrester-study-the-total-economic-impact-of-qt-for-device-creation.

[23] *Qt for Beginners* [online]. The Qt Project, 2019. [cit. 2020-03-04]. Dostupné z: https://wiki.qt.io/Qt_for_Beginners.

[24] *Model/View Tutorial* [online]. The Qt Company, c2016. [cit. 2020-04-24]. Dostupné z: https://doc.qt.io/archives/qt-4.8/modelview.html.

[25] *Ultimate++ is a C++ cross-platform rapid application development framework* [online]. 2019. [cit. 2019-10-19]. Dostupné z: https://www.ultimatepp.org/.

[26] *GNU General Public License, version 2, with the Classpath Exception* [online]. Oracle, 1991. [cit. 2019-10-11]. Dostupné z: http://openjdk.java.net/legal/gplv2+ce.html.

[27] *What is Tk* [online]. 2018. [cit. 2019-10-19]. Dostupné z: https://wiki.tcl-lang.org/page/What+is+Tk.

[28] *Supported Build Platforms* [online]. Attlasian, 2019. [cit. 2019-10-10]. Dostupné z: https://wiki.openjdk.java.net/display/Build/Supported+Build+Platforms.

[29] *GTK* [online]. Wikimedia Foundation, 2013–2019. [cit. 2019-10-11]. Dostupné z: https://en.wikipedia.org/wiki/GTK.

[30] *Tcl/Tk Community Resources* [online]. Tcl Developer Xchange, 2019. [cit. 2019-10-19]. Dostupné z: https://www.tcl.tk/community/.

[31] *JavaFX* [online]. Wikimedia Foundation, 2019. [cit. 2019-10-19]. Dostupné z: https://en.wikipedia.org/wiki/JavaFX.

[32] *Getting Started* [online]. Paul D Turner & The CEGUI Development Team, c2004-2015. [cit. 2019-12-29]. Dostupné z: http://cegui.org.uk/content/getting-started.

[33] *Runs everywhere!* [online]. The GTK Team, 2017–2019. [cit. 2019-10-11]. Dostupné z: https://www.gtk.org/screenshots/index.php.

[34] *Supported platforms* [online]. 2019. [cit. 2019-10-19]. Dostupné z: https://www.ultimatepp.org/www\protect\T1\textdollaruppweb\protect\T1\textdollarSupportedPlatforms\protect\T1\textdollaren-us.html.

[35] *Unity for all* [online]. Unity Technologies, c2019. [cit. 2019-10-20]. Dostupné z: https://unity.com/.

[36] *User Interface Elements* [online]. U.S. Department of Health & Human Services, c2019. [cit. 2019-11-03]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/user-interface-elements.html.

[37] *Fast Light Toolkit* [online]. The FLTK Team, 2019. [cit. 2019-12-29]. Dostupné z: https://www.fltk.org/index.php.

[38] *License* [online]. WxWidgets, 2019. [cit. 2019-10-11]. Dostupné z: https://wiki.wxwidgets.org/License.

[39] *Model/View Programming* [online]. The Qt Company, c2019. [cit. 2020-04-05]. Dostupné z: https://doc.qt.io/qt-5/model-view-programming.html.

[40] *About Qt* [online]. The Qt Company, 2019. [cit. 2019-10-08]. Dostupné z: https://wiki.qt.io/About_Qt.

[41] *Unity 5.3.4* [online]. Unity Technologies, c2019. [cit. 2019-10-20]. Dostupné z: https://unity3d.com/unity/whats-new/unity-5.3.4.

[42] *Přehled grafického subsystému WPF (Windows Presentation Foundation)* [online]. Microsoft, 2016. [cit. 2019-10-13]. Dostupné z: https://docs.microsoft.com/cs-cz/visualstudio/designers/introduction-to-wpf?view=vs-2019.

[43] *FAQ* [online]. The Qt Company, 2019. [cit. 2019-10-08]. Dostupné z: {https://www.qt.io/FAQ/},.

[44] *Get Assistance* [online]. The GTK Project, c2019. [cit. 2019-10-12]. Dostupné z: https://www.gtk.org/support.php.

[45] *JUCE* [online]. c2019. [cit. 2019-12-29]. Dostupné z: https://juce.com/.

[46] *Browser & Platform Market Share* [online]. [cit. 2020-04-11]. Dostupné z: https://www.w3counter.com/globalstats.php.

[47] *Unity Community* [online]. Unity Technologies, c2019. [cit. 2019-10-20]. Dostupné z: https://unity3d.com/community.

[48] *Newt (programming library)* [online]. Wikimedia Foundation, 2019–2019. [cit. 2019-11-03]. Dostupné z: https://en.wikipedia.org/wiki/Newt_(programming_library).

[49] *Unity Personal* [online]. Unity Technologies, c2019. [cit. 2019-10-20]. Dostupné z: https://store.unity.com/products/unity-personal.

[50] *License* [online]. 2016. [cit. 2019-10-19]. Dostupné z: https://wiki.tcl-lang.org/page/license.

[51] A.M. Kuchling, E. S. R. *Curses Programming with Python* [online]. Python Software Foundation, c2001-2020. [cit. 2020-04-11]. Dostupné z: https://docs.python.org/3/howto/curses.html.

[52] Chroboczek, M. *Grafická uživatelská rozhraní v Qt a C*. Computer Press, 1 edition, 2013. ISBN 978-80-251-4124-3.

[53] Dickey, T. E. *Licensing* [online]. 2018. [cit. 2019-11-02]. Dostupné z: https://invisible-island.net/ncurses/ncurses-license.html.

[54] Dickey, T. E. *Announcing ncurses 6.1* [online]. GNU, 2018. [cit. 2019-11-02]. Dostupné z: https://www.gnu.org/software/ncurses/.

[55] Garfinkel, S. – Mahoney, M. K. *Building Cocoa Applications: A Step-by-Step Guide*. "O'Reilly Media, Inc.", 2002.

[56] Landwerth, I. *Windows API* [online]. Wikimedia Foundation, 2007-2017. [cit. 2019-12-30]. Dostupné z: https://cs.wikipedia.org/wiki/Windows_API.

[57] Lichvar, M. *New* [online]. Red Hat, Inc., 2016. [cit. 2019-11-02]. Dostupné z: https://pagure.io/newt.

[58] Marsh, J. *UX pro začátečníky*. Zoner Press, 1 edition, 2019. ISBN 978-80-7413-397-8.

[59] Vojtaj, J. *Programování v JavaFX: úvod, příprava systému a prostředí* [online]. 2015. [cit. 2019-10-11]. Dostupné z: https://www.root.cz/clanky/programovani-v-javafx-uvod-priprava-systemu-a-prostredi/.

Přílohy

A Uživatelská dokumentace

Celý vývoj probíhal na operačním systému *Debian GNU/Linux 9*, protože aplikace napsané v knihovně NCurses jsou spustitelné jenom na operačních systémech založených na linuxovém jádře. Veškeré testování funkčnosti tedy probíhalo na Linuxu a následující podkapitoly popisující sestavení a používání aplikací předpokládají běh aplikací na tomto operačním systému. První podkapitoly jsou věnovány aplikaci napsané v Qt a po nich následující informace k aplikace napsané v NCurses.

Aplikace nejsou hlavním produktem bakalářské práce, sloužily primárně k porovnání vlastností obou knihoven, a proto jejich používání není detailně popisováno.

A.1 Sestavení Qt aplikace

Ve složce `assembly` je připraven soubor `Makefile`, který byl vygenerován programem *qmake* podle projektového souboru `inodes_manager_qt.pro`. Tento soubor definuje standardní proměnné pro označení zdrojových a hlavičkových souborů, které jsou použity v projektu.

Provedením příkazu `make` ve složce `assembly` dojde k sestavení programu a vytvoření spustitelného souboru `inodesfs_manager_qt`. Pro sestavení je nutné mít nainstalované veškeré závislosti Qt, a proto je ve složce `inodesfs_manager_qt` připraven i spustitelný soubor se stejným názvem. Ve obou složkách se nachází soubor `disk` představující virtuální disk a obsahující testovací data pro práci se souborovým systémem. Aplikaci lze spustit z terminálu následujícím způsobem `./inodesfs_manager_qt disk`.

A.2 Používání Qt aplikace

Po spuštění lze aplikaci ovládat klávesnicí i myší a provádět veškeré operace nad VFS definované v případech užití v podkapitole 7.2. Vypnutí se uskuteční stisknutím křížku hlavního okna nebo skrze horizontální menu, kdy po rozbalení nabídky `Soubor` je možnost `Ukončit`. Při ukončení dochází k uložení aktuálního stavu struktur na virtuální disk. Součástí GUI je komponenta emulující terminál – uživatel může zadáváním shellových příkazů ovládat VFS i touto cestou. V terminálu jsou podporovány tyto příkazy:

- `cp soubor_1 umístění_1` – kopírování souboru_1 na umístění_1
- `mv soubor_1 soubor_2` – přesun souboru_1 na umístění_1
- `rm soubor_1` – smazání souboru_1
- `mkdir adresář_1` – vytvoří adresář_1
- `rmdir adresář_1` – smaže adresář_1
- `ls adresář_1` – vypíše obsah adresáře_1
- `cat soubor_1` – vypíše obsah souboru_1
- `cd adresář_1` – změní aktuální cestu do adresáře_1
- `pwd` – vypíše aktuální cestu
- `info adresář_1/soubor_1` – vypíše informace o souboru_1/adresáři_1
- `incp soubor_1 umístění_1` – nahraje soubor_1 z pevného disku do umístění_1
- `outcp soubor_1 umístění_1` – nahraje soubor_1 z VFS na umístění_1 pevného disku
- `load soubor_1` – načte soubor_1 z pevného disku, ve kterém budou jednotlivé shellové příkazy, které se sekvenčně vykonají
- `format 600MB` – formátování disku na velikost 600 MB.

A.3 Sestavení NCurses aplikace

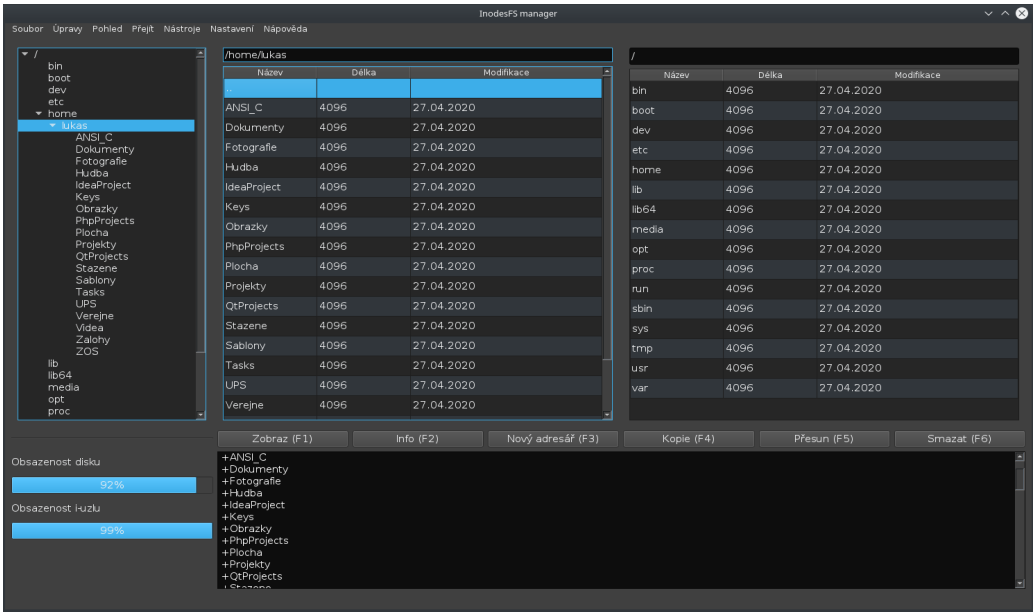
V kořenovém adresáři aplikace je připraven soubor `Makefile`, pomocí kterého se provede automatické přeložení programu příkazem `make`. V sestavovacím souboru jsou závislosti na knihovnu NCurses a toolkit CDK. Pokud by čtenář nechtěl uvedené závislosti instalovat, může rovnou využít spustitelný soubor a virtuální disk a aplikaci spustit z terminálu následujícím způsobem `./inodesfs_manager_ncurses disk`.

A.4 Používaní NCurses aplikace

Aplikaci je možné používat v textovém a grafickém režimu. V textovém režimu jsou dostupné shellové příkazy uvedené v podkapitole A.2. Po zadání příkazu `gui` se aplikace přepne do grafického režimu. Ovládaní je srovnatelné s Qt aplikací, ale nelze k ovládání používat myš. Mezi jednotlivými widgety GUI se přechází pomocí klávesy `TAB` a pomocí funkčního tlačítka `F1` dojde k zobrazení nápovědy k ovládání aplikace.

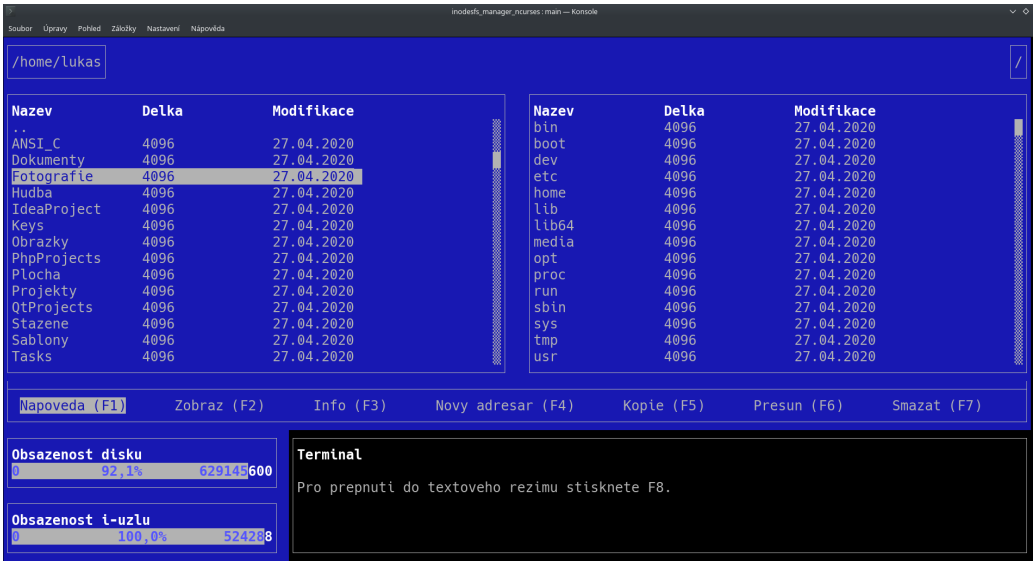
B Obrazová příloha

B.1 Aplikace vyvíjená v Qt



Obrázek 1: Ukázka aplikace vyvíjené knihovně v Qt

B.2 Aplikace vyvíjená v NCurses (CDK)



Obrázek 2: Ukázka aplikace vyvíjené knihovně v NCurses (CDK)