VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

NOVÁ GENERACE NÁSTROJE FEDORA MEDIAWRITER

NEXT GENERATION OF FEDORA MEDIAWRITER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **EVŽEN GASTA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. DOMINIKA REGÉCIOVÁ**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Gasta Evžen**
Program: Informační technologie
Název: **Nová generace nástroje Fedora MediaWriter**
Next Generation of Fedora MediaWriter
Kategorie: Softwarové inženýrství

Zadání:

1. Seznamte se s knihovnou Qt a programovacím jazykem QML a jeho použití v aplikaci Fedora MediaWriter.
2. Pište články (blogy) o průběhu vašeho vývoje a podělte se o vaše výsledky s komunitou pro získání zpětné vazby a umožněte uživatelům vaši práci předčasně vyzkoušet.
3. Implementujte nové grafické rozhraní aplikace Fedora MediaWriter dle grafického zadání a dle komunikace s grafickými návrháři:
   a. Upravte aktuální grafický styl (Adwaita) pro použití na Linuxu
   b. Použijte nový nativní grafický styl (Windows, MacOS) dostupné v Qt 6.
4. Implementujte nové žádané funkce dle závislosti na obtížnosti předchozího bodu a po konzultaci s vedoucím a konzultantem.
5. Upravte instalátory pro Windows a MacOS pro Qt 6 a vaše změny.
6. Vydejte novou verzi aplikace Fedora MediaWriter a opravte uživateli nahlášené chyby.

Literatura:

- https://github.com/FedoraQt/MediaWriter
- Stephen Prata: Mistrovství v C++ 4. aktualizované vydání, Cpress (2013). ISBN 978-80-2513-828-1.
- Nibedit Dey: Cross-Platform Development with Qt 6 and Modern C++: Design and build applications with modern graphical user interfaces without worrying about platform dependency, Packt Publishing (2021), ISBN 978-1800204584.
- Příklady žádostí o nové funkce. Dostupné na URL: https://github.com/FedoraQt/MediaWriter/issues
- Dokumentace pro Qt. Dostupné na URL: https://doc.qt.io/qt-6/

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvního, druhého a částečně třetího bodu (3a).

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Regéciová Dominika, Ing.**
Konzultant: Grulich Jan, RedHatCZ
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 14. října 2021

Abstrakt

Cílem této práce je vytvoření nové verze nástroje Fedora Media Writer, pomocí kterého se vytváří bootovací disk USB s operačním systémem Fedora. Hlavním důvodem předělaní je podpora nativního vzhledu na operačních systémech Windows a macOS. Nová verze frameworku Qt tohoto může dosáhnout. Výsledkem je nový vzhled aplikace s dalšími vylepšeními. Přínosem této práce je aktualizace open source aplikace, o kterou měla komunita zájem. Výsledná aplikace umožňuje používat Fedora Media Writer s nativním designem na různých operačních systémech.

Abstract

This bachelor thesis aims to create a new version of the tool Fedora Media Writer, used for creating bootable USB with the operating system Fedora. The main purpose of this rework is the support of native look on operating systems Windows and macOS. The new version of the framework Qt is able to do that. The result is a new look of the application with other upgrades.The benefit of this bachelor thesis is the actualization of an open-source application, about which the community was interested in. The final application allows using Fedora Media Writer with native design on different operating systems.

Klíčová slova

Qt, QML, CMake, Fedora, GUI, C++, Operační systém

Keywords

Qt, QML, CMake, Fedora, GUI, C++, Operating system

Citace

GASTA, Evžen. *Nová generace nástroje Fedora Mediawriter*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Dominika Regéciová

Nová generace nástroje Fedora Mediawriter

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením paní Ing. Dominiky Regéciové. Další informace mi poskytl pan Bc. Jan Grulich. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Evžen Gasta
7. května 2022

Poděkování

Děkuji paní Ing. Dominice Regéciové, za ochotu při vybírání tématu této bakalářské práce a zároveň za milé a pohotové jednání v průběhu celé práce. Dále bych chtěl poděkovat svému technickému vedoucímu panu Bc. Janu Grulichovi za to, že jsem se na něj mohl obrátit kdykoliv jsem měl potíže a za to, že jsem se mohl od něj mnohé přiučit. Také bych chtěl poděkovat svým rodičům za to, že mi umožnili toto studium a za morální podporu v průběhu celého studia. Poslední poděkování patří mé přítelkyni Haně Šoupalové, která mě podporovala, i když jsem se jí nemohl vždy věnovat tak jak bych si přál.

Obsah

Kapitola 1

Úvod

V dnešní době se považuje počítač za nedílnou součást lidských životů. Většina populace se s počítačem setkává poměrně často, ať už v práci, v komunikaci, ve škole nebo v soukromém životě k hledání různých informací. Chci tím říct, že většina populace se s počítačem setkává neustále. Počítače nám usnadňují naše každodenní činnosti. Abychom si ulehčili práci s těmito počítači a byli je schopni lépe a efektivněji používat, byla vyvinuta skupina programů nazvaných operační systém, které nám tento problém pomohou vyřešit. Většina počítačů (laptopů, mobilních telefonů a desktopových počítačů) přichází s již předinstalovaným systémem. Tento operační systém nám z různých důvodů nemusí vyhovovat, jak pro hardwarovou náročnost, využitelnost systému nebo různé případy použití.

Instalace operačního systému se může zdát pro běžného uživatele poměrně náročnou a riskantní záležitostí. Z toho důvodu se tato operace nechává na zkušenějších uživatelích, kteří mají v této oblastí již určité znalosti. Aby se ulehčilo vytváření tzv. *bootable USB* (přenosný USB disk obsahující operační systém pro instalaci) byly vytvořeny různé programy, které dokážou tuto operaci usnadnit. Mezi tyto programy patří například *Media Creation Tool*, *A Bootable USB*, *Rufus*, nebo také *Fedora Media Writer*.

Cílem této bakalářské práce je vylepšit současnou verzi právě zmiňovaného nástroje Fedora Media Writer. Hlavním důvodem úpravy je podpora nativního vzhledu na různých operačních systémech, což je možné díky nové verzi frameworku Qt, která tyto nativní vzhledy u operačních systémů podporuje, viz Novinky v Qt6 oproti Qt5. Pro řešení této práce je potřeba se seznámit s grafickým frameworkem Qt, nástrojem CMake - qmake a naučit se distribuovat aplikace na různé operační systémy. Důležitou součástí bakalářské práce bude zorientování se a pochopení funkčnosti jednotlivých částí rozsáhlého již existujícího open source projektu.

V kapitole 2 si uděláme rozbor různých operačních systémů včetně jejich historie. Kapitola 3 popisuje, co to vlastně grafické uživatelské rozhraní (anglicky Graphic User Interface, dále jen GUI) je a co znamenají pojmy frontend a backend. Dále se v této kapitole dozvíme o programovacím jazyku s jehož pomocí je možné vytvořit operační systémy a jejich styly. Na tuto kapitolu navazuje kapitola 4, ve které si povíme co to je Qt, QML a kde můžeme najít jejich využití. V 5. kapitole bude představen již výše zmíněný nástroj, díky kterému můžeme nainstalovat operační systém Linux resp. distribuci Fedory. Tato část rozebírá také návrh nového GUI nebo Současné nedostatky aplikace. Samostatný vývoj aplikace popisuji v kapitole 6. Následuje předposlední kapitola Testování aplikace vývojáři a uživateli, na kterou navazuje závěr.

Kapitola 2

Operační systém

Operační systém je nedílnou součástí každého koncového zařízení [4].

Operační systém můžeme oprávněně nazvat jako spojující vrstvu mezi uživatelem a hardwarem systému. Nabízí počet služeb programům a uživatelům skrze *Application Programming Interface* (API). Počítač má množství zdrojů, jako je čas CPU, dostupná paměť, nebo také vstupně výstupní rozhraní. Operační systém slouží jako rozdělovač a správce těchto zdrojů. Dělá rozhodnutí o tom, jaké zdroje, kolik a na jak dlouho budou přiděleny programu, který zrovna běží. Informace byly převzaty z knihy Operating Systems [16].

2.1 Linux

Za zakladatele operačního systému Linux se považuje *Linus Torvalds*, který napsal první verzi Linuxového jádra v roce 1991, jako svůj volnočasový projekt. Od té doby se kolem jádra Linuxu vytvořila obrovská komunita příznivců jak jednotlivců, tak firem (RedHat), která do jádra Linuxu přispívala.

Linux je open source operační systém, který spadá pod licenci GNU. To znamená, že je možné operační systém Linux stahovat, distribuovat, kopírovat, upravovat a zkoumat zdrojové kódy dle libosti, bez jakéhokoliv postihu. Linux jako takový je zdarma, existují ale firmy, které tento systém prodávají. Je to z důvodu přidané hodnoty, jako jsou knihy, technická podpora, manuály, nebo zaučení zákazníka. Mohlo by se zdát, že stáhnutí takového systému bude nevýhodné, z pohledu stability, poruchovosti, nebo hardwarové náročnosti, když do vašeho jádra přispívají lidé z celého světa. Opak je ale pravdou. Při jakémkoliv problému stačí kontaktovat komunitu dané distribuce a odpoví vám tisíce nadšenců, kteří se dají s nadsázkou považovat za 24h technickou podporu.

	CPU- dual Core	RAM	ROM
Linux (Fedora)	1 GHz	2 GB	10 GB
Tiny Core Linux	$<$ 500 MHz	46 MB	15 MB
Windows 10	1 GHz	1 GB	16 GB - 20 GB
macOS Big Sur	-	4 GB	35,5 GB

Tabulka 2.1: Porovnání systémových požadavků

Linux se nachází nejen na desktopových systémech, ale i na mobilních zařízeních a to v podobě Androidu. Ten je založený právě na modifikovaném jádru Linuxu.

Pod slovem Linux se neskrývá pouze 1 systém, ale několik desítek, možná i stovek různých distribucí. Tyto distribuce jsou vytvářeny různými komunitami a firmami, které svou distribuci udržují aktuální a poskytují ji určitou podporu. Mezi nejpoužívanější distribuce patří Fedora, Ubuntu a nebo Debian. Každá distribuce Linuxu má své výchozí GUI, které může být dostupné ve více distribucích, v případě nespokojenosti je možné toto GUI libovolně měnit.

2.1.1 Fedora

Jak již bylo zmíněno výše, Fedora je jednou z nejpoužívanějších distribucí Linuxu. Tato distribuce je používána zejména vývojáři a mimo jiné i zakladatelem Linuxu, Linusem Tovaldsem, což svědčí o její kvalitě. Je to díky jejímu stabilnímu a bezpečnému prostředí, které je již od základu nastaveno tak, aby ulehčilo programátorům jejich vývoj. Navíc dostává pravidelně bezpečnostní záplaty a opravy chyb.

Fedora jak ji známe vznikla roku 2003 spojením open source projektu *RHL* (Red Hat Linux) a *Fedora Linux Projektem*. Tímto spojením vznikl open source projekt nazvaný *Fedora Project*[1], který je nadále sponzorovaný Red Hatem a podporovaný komunitou[2]. Ačkoliv je Red Hat sponzorující firma, počet zaměstnanců vyvíjející open source Fedoru je pouze 35 procent, zbylá část firmy vyvíjí komerční verzi Fedory a to *RHEL*. Více informací v knize Red Hat Enterprise Linux 8 Essentials [21].

Abychom mohli spravovat balíčky (instalovat aplikace), bylo potřeba vytvořit různé Package Managery. *RPM* je Package Manager vyvinutý Red Hatem pro Fedoru. Všechny RPM balíčky (přípona `.rpm`) včetně závislostí, jsou uloženy v lokální databázi. Při správě RPM balíčků si můžeme práci usnadnit pomocí aplikací *DNF* a *PackageKit*. Velikou výhodu RPM je bezpečná práce s konfiguračními soubory, díky které se při updatu nepřepíší vaše úpravy aplikace. Další výhodou RPM je schopnost instalovat nové verze aplikace, aniž by došlo ke kompletní re-instalaci aplikace.

2.2 Windows

V dnešní době můžeme považovat za nejpoužívanější operační systém z rodiny Windows. Pod pojmem Windows se uvažuje skupina operačních systémů vyvíjených firmou Microsoft. Tato firma byla založena v roce 1975 Billem Gatesem a Paulem Allenem. Díky tomu, že je možný běh více aplikací současně (na novějších verzích Windows i více ploch současně), má Windows vysokou úroveň multitaskingu. Oproti operačnímu systému Linux je Windows proprietární a je poměrně složité provádět s tímto operačním systémem větší úpravy.

První verzí Windows byla verze MS-DOS, která byla vydána roku 1981 a založena na CLI viz GUI. Komerčního úspěchu se dočkaly až verze Windows 3.0 a Windows 95. Tyto verze zahájily sérii poměrně úspěšných verzí a variant operačního systému Windows nazvanou Windows NT (new technology). Tento název nese označení pro 32 a 64bitovou řadu operačního systému s podporou preemptivního multitaskingu. Do této úspěšné série verzí patří například legendární verze Windows XP, Windows 7 nebo Windows 10. Tyto verze byly většinou náhradou za nepříliš úspěšné verze, jako třeba Windows Vista nebo Windows 8.

[1] https://web.archive.org/web/20031001204515/http://www.fedora.us/
[2] https://web.archive.org/web/20030923215031/http://fedora.redhat.com/

Mezi známou variantu Windows patří varianta na mobilní telefony a to Windows Mobile. Tato varianta byla založena na operačním systému Windows 7 a nesklidila příliš velký úspěch.

2.3 macOS

Posledním systémem, o kterým si povíme, bude operační systém macOS. Ten je vyvíjen firmou Apple, která byla založena Stevem Wozniakem a Steavem Jobesem. První verze macOS tedy (Mac OS) byla představena spolu s osobním počítačem Macintosh v roce 1984. Na rozdíl od MS-DOS od Windowsu, měl macOS již od první verze GUI, ačkoliv měl původně podporu běhu pouze jedné aplikace. Začátkem nového století proběhl rozsáhlý redesign GUI a zároveň byl MacOS přejmenován na macOS. Verze macOS jsou založené na UNIXovém jádru, podporují preemptivní multitasking a jsou naprogramovány, aby pracovaly výhradně na zařízeních Apple.

Což může být jak výhodou, tak zároveň nevýhodou. Výhodou je bezpochyby optimalizace. Díky tomu, že si v této době Apple vyvíjí své procesory a staví vše „na míru“, může oproti Linuxu nebo Windows zaručit vyšší kvalitu svých produktů. Díky tomu si drží Apple svou prémiovou značku a je považován za prémiový standard. Toto lze považovat jak za výhodu, tak za její nevýhodou. Důvodem je složitost veškerých oprav, či různých vylepšení systému, které jsou často velice problematické (ne-li zcela nemožné).

Dalšími operačními systémy od společnosti Apple jsou watchOS pro chytré hodinky Apple a iOS pro mobilní telefony a tablety. Všechny tyto operační systémy jsou založeny na macOS.

Kapitola 3

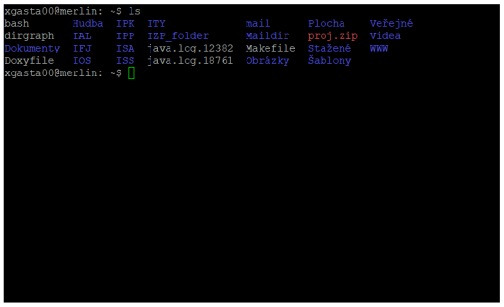
GUI

GUI je forma rozhraní, přes kterou je uživatel schopen ovládat software. Vyskytuje se nejen ve stolních počítačích, ale i v mobilních zařízeních, tabletech, chytrých hodinkách a dalších elektronických zařízeních. Zobrazuje jednotlivé elementy, jako jsou tlačítka, návěští, obrázky a spoustu dalších objektů. Uživatel může dle své potřeby tyto objekty upravovat. Měnit velikosti, barvy, nebo také přesouvat z jednoho místa na druhé. Cílem GUI je umožnit uživateli jednoduchou, přehlednou a rychlou práci s daty.

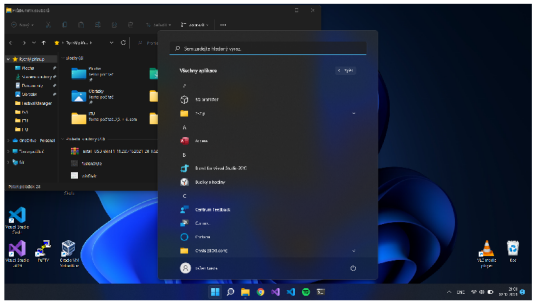
3.1 Historie GUI

V době, kdy neexistovalo GUI, používali lidé konzolové řádkové rozhraní (Command Line Interface - CLI). V případě použití konzolového rozhraní si musel uživatel pamatovat jednotlivé příkazy na ovládání softwaru, ačkoliv i v dnešní době se převážně na operačním systému Linux toto CLI stále aktivně používá. Důvodem je rychlost provedení požadavku, které může být až několikanásobně rychlejší než v podání GUI.

Jeden z prvních počítačů s GUI byl Xerox Alto, ten byl vyvinut v roce 1972 ve výzkumném centru Palo Alto. Xerox Alto byl inspirován oN-Line System (zkráceně NLS), který vyvinul pan Douglas Engelbart[1] roku 1960 a sloužil spíše pro vědecké, než pro komerční použití. První počítač pro veřejnost byl vyvinut společností Apple v čele se Stevem Jobsem. Počítač Apple LISA byl majoritně ovlivněn Xeroxem Altoem, který navíc představil nové koncepty a prvky [18], jako jsou například koš, drag and drop a nebo schránku, pro kopírování textu, či dokumentů, což jsou pro nás již naprosto běžné funkce.



Obrázek 3.1: CLI na školním serveru merlin.fit.vutbr.cz



Obrázek 3.2: GUI operačního systému Windows 11

[1] https://cs.wikipedia.org/wiki/Douglas_Engelbart

3.2 Backend

Jako backend označujeme část programu, ve kterém je uložena veškerá logika aplikace, která není vidět zvenčí. Tato logika může být libovolně komplexní a napsaná v jakémkoliv programovacím jazyce (C++, Java, C#, Ruby, Python, PHP a spoustu dalších). Aplikace pracuje nad daty, které jsou následně zobrazovány uživateli v tzv. prezentační vrstvě. Data bývají uložena v databázích jak na serverech, tak i na lokálních databázích v přístroji na kterých jsou spuštěny. Tyto data jsou obvykle uložena pomocí JSON[2] (JavaScript Object Notation), XML[3], YAML, či jedné z několika SQL databází.

Díky tomu, že je v této době velice populární architektura klient-server (dvouvrstvá) [22], je co největší část logiky (tedy backendu) implementována na centrálním serveru. Na tento server jsou připojeni jednotliví klienti, kteří mají implementovanou pouze nezbytnou část logiky pro komunikaci s tímto serverem. Většinou platí „čím tlustší klient, tím větší logika“. V praxi je také možné se setkat s třívrstvou architekturou, což je pouze architektura klient-server rozšířená o tzv. datovou vrstvu na straně serveru.

3.2.1 C++

Je kompilovaný programovací jazyk, díky tomu je možné vytvořit velice rychlé programy. Jedná se o silně a staticky typovaný, objektově orientovaný imperativní jazyk. Díky tomu, že je nízkoúrovňový dovoluje uživateli pracovat s pamětí, v tom je jak jeho síla, tak i slabost (dá se lehce provést chyba). Díky jeho rychlosti a univerzálnosti, je častou volbou pro tvorbu operačních systémů, aplikací s GUI nebo her. V této bakalářské práci, je celá logika a funkcionalita implementována v jazyku C++.

Jazyk C++ je rozšířený jazyk C, nebo také „C s třídami“. C++ je standardizováno od roku 1998 International Organization for Standardization (ISO), tato organizace vydává od roku 2012 každé 3 roky nový standard. Nejnovějším standardem tohoto jazyka je *C++20*.

Clang

Jedním z mnoha překladačů pro jazyk z rodiny C (C, C++, Objective C/C++, OpenCL, CUDA) je překladač Clang. Ten je součástí open source projektu LLVM, což je kolekce modulárních, znovupoužitelných překladačů a zřetězených technologií. Tento projekt byl založen na Universitě v Illinoisu, jehož cílem bylo vytvořit SSA (static single assignment) kompilační strategii, která by byla schopna podporovat jak statickou, tak dynamickou kompilaci jazyku C. Do tohoto projektu patří nejen zmiňovaný překladač Clang, ale i knihovny (libc++, aj.), optimalizátory (opt) a další. Tento překladač se používá pro překlad software kritických na výkon jako je Chrome, nebo Firefox. Zároveň se používá pro kompilování Fedora Media Writeru na operačním systému macOS, nebo také na kontrolu formátu v GitHub Workflow více o Workflow v 4.4.5.

Informace byly převzaty ze stránek LLVM [11].

GCC

Dalším způsobem, jak přeložit jazyk C++ je pomocí kompileru *GCC*. Ten byl původně vyvíjený pro GNU operační systémy, z toho vznikl název GCC tedy *GNU Compiler Collection*. Tento překladač podporuje stejně jako Clang jazyky z rodiny C, také Fortran, Ada, Go, D,

[2] https://www.json.org/json-en.html
[3] https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction

nebo knihovny (libstdc++, aj.) pro tyto jazyky. Tento překladač je dostupný jako příkaz `gcc` pro jazyk C nebo `g++` pro jazyk C++. A používá se pro kompilaci nástroje Fedora Media Writer na operačním systému Linux.

Informace byly převzaty ze stránek GNU [5].

MinGW

Na operačním systému Windows se používá kompiler *MinGW*, který je také známý jako *mingw32*. Vzorovým kompilerem byl výše uvedený GCC, ten byl upraven tak, aby se dal používat na operačním systému Windows. Díky tomuto kompileru je možné vytvářet C++ aplikace, které používají pouze Windows API s minimální podporou GNU POSIX vrstvy (od toho název „Minimalist GNU for W32“). Toto přináší značné omezení, ale i výhody. Mezi omezení patří používání funkcí `fork` nebo `mmap`, které není možné při použití MinGW překladače. Pokud bychom chtěli na Windows použít tyto funkce musíme použít překladač *Cygwin*, který tyto funkce implementuje. Výhodou může být, že k používání MinGW není potřeba žádná speciální knihovna ani různé nástroje pro běh aplikace a tím pádem ani GPL licence. V nástroji Fedora Media Writer se používá pro kompilaci a vytváření buildů na Windows.

3.2.2 KDevelop

KDevelop je moderní open source IDE (Integrated Development Environment), licencovaný pod GNU, pro C/C++, QML, JavaScriptu a Pythonu. Patří do rodiny *KDE Aplikací*, které jsou dostupné na KDE platformě. Je dostupný jak na Unixových operačních systémech, tak i na Windows. V době psaní této práce existuje tzv. preview i na macOS. Je vhodný pro veliké projekty a aplikace, nejdůležitější funkcí KDevelopu je, že rozumí C++.

Cílem KDE při tvoření KDevelopu bylo, vytvořit moderní vývojové prostředí, které dovoluje programátorům pracovat na projektech libovolných rozměrů. Srdcem KDevelopu je kombinace vylepšeného editoru se sémantickou analýzou kódu, která poskytuje obohacenou zkušenost při vývoji díky porozumění projektu. Zároveň KDevelop nabízí vývojáři různé pracovní postupy při vývoji. Tímto pomáhá KDevelop zlepšovat kvalitu kódu a ověřuje její funkcionalitu.

Má vestavěnou velikou škálu často používaných nástrojů a funkcí, jako je:

- Správa projektů různých typů, jako CMake a qmake
- Systému zprávy verzí (VCS), jako je např. Git
- Nástroj pro dokumentaci *Doxygen*
- Přidávání nových C/C++, nebo Python tříd pomocí šablon
- Podpora Unit testů a unit testing frameworků, jako *CTest* nebo *PyUnit*

Informace byly převzaty z manuálu *KDevelop4/Manual* [8] a webových stránek *KDevelop* [9].

3.3 Frontend

Je pojem, pod kterým se skrývá vizuální stránka aplikace. Velice důležitou částí frontendu je návrh UX [14] (User experience) a UI (GUI či CLI). Snahou UX je udělat aplikaci

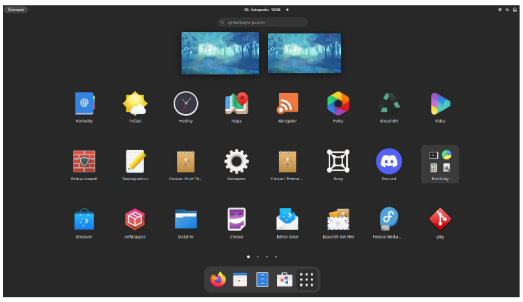
přehlednou, intuitivní a bez zbytečných informací. UX je dobře navrženo když je uživatel spokojený s používáním aplikace a nepřemýšlí nad jejím používáním. Cílem UX je naplnit byznys požadavky (typickým příkladem byznys požadavků může být třeba provedení objednávky na webové stránce). Návrh[4] UX je poté předán UI vývojářům, kteří mají za úkol udělat design aplikace. Jedná se tedy o různé barvy, emotikony, tlačítka, texty atd.

3.3.1 Fedora

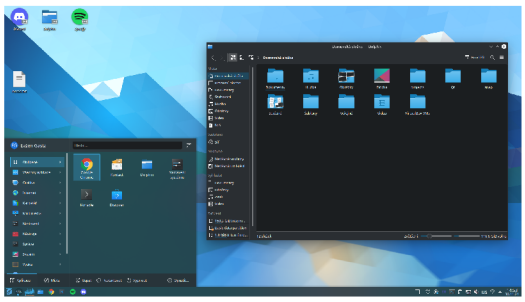
Fedora má dvě majoritní desktopové prostředí a to *Plasma* a *GNOME*. Plasma je napsaná v Qt frameworku a je vyvíjena mezinárodní komunitou vyvíjející open source software, se stylem nazvaným Breeze. Toto prostředí je vyvíjeno mezinárodní komunitou KDE, která také vyvíjí KDE aplikace, jako je souborový manager *Dolphin*, terminál *Yakuake*, editor *Kate* a spoustu dalších. Pro běh těchto KDE aplikací je potřeba tzv. *KDE Platforma*, která se skládá z potřebných knihoven. Informace byly převzaty z bakalářské práce Nástroje pro práci s Google službami v KDE [6].

Naopak GNOME používá framework nazvaný GTK s grafickým rozhraním Adwaita. Jelikož není styl GNOME napsaný jako styl Plasma v Qt, byla vytvořena Adwaita-qt, která se snaží dát Qt aplikacím vzhled Adwaita, aby vypadaly stejně jako aplikace běžící v GNOME. Porovnání těchto desktopových prostředí najdeme v bakalářské práci Prohlížeč obrázků pro desktopové prostředí KDE [7].

Obě tato prostředí mají na výběr ze dvou zobrazovacích protokolů, starší *X11* a novější *Wayland*, které specifikují komunikaci mezi zobrazovacím serverem a jednotlivými aplikacemi.

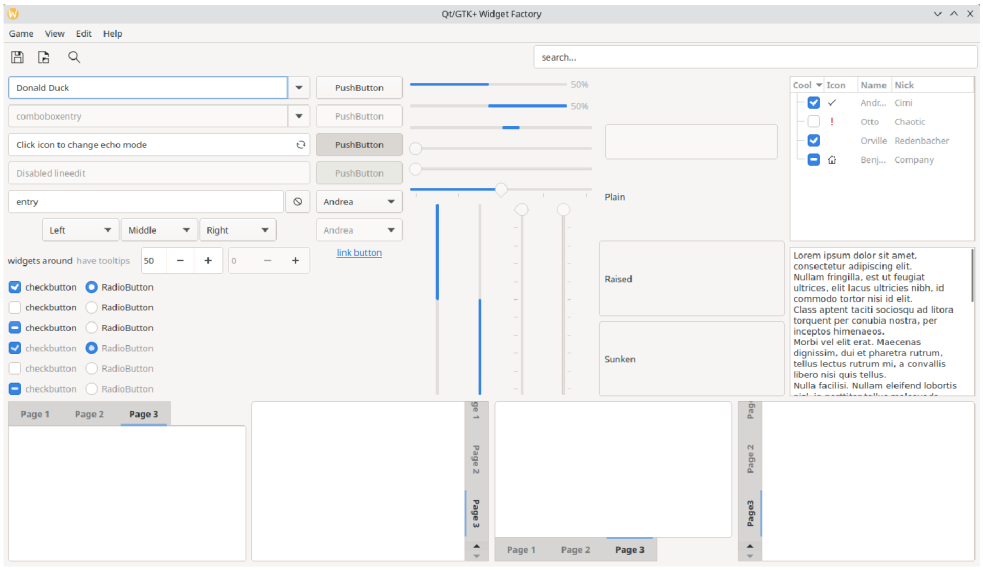


Obrázek 3.3: Výchozí prostředí GNOME



Obrázek 3.4: Volitelné prostředí KDE Plasma

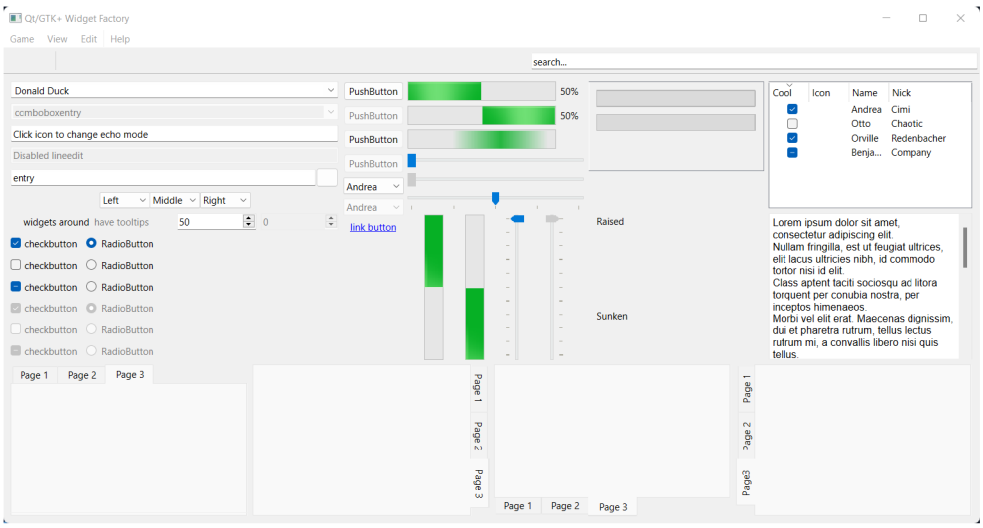
[4] Černobílý, bez barev, aby se neovlivnil UI designer



Obrázek 3.5: Styl Adwaita

3.3.2 Windows

Hlavním cílem při tvorbě Windows GUI je držet se tzv. „Fluent Design System“. Což je designový open source standard vyvinutý firmou Microsoft, který poskytuje UI vývojářům framework pro tvorbu intuitivního, výkonného a multiplatformního prostředí. Pomocí tohoto standardu jsou vytvořené aplikace dostupné na Windows 10 a novějších, jako jsou Mapy, Mail, To Do, Obrázky a další. Fluent Design System je založený na pěti základních principech a to „světla, hloubky, měřítka, základních materiálech a pohybu“. Při vývoji na Windows je tento standard obsažen ve WinUI (Windows UI Library), což je UX framework pro Windows, který se používá pro vývoj Windows a UWP aplikací. Pro renderování GUI se používá *API DirectX* a zobrazovací protokol *Desktop Window Manager*.

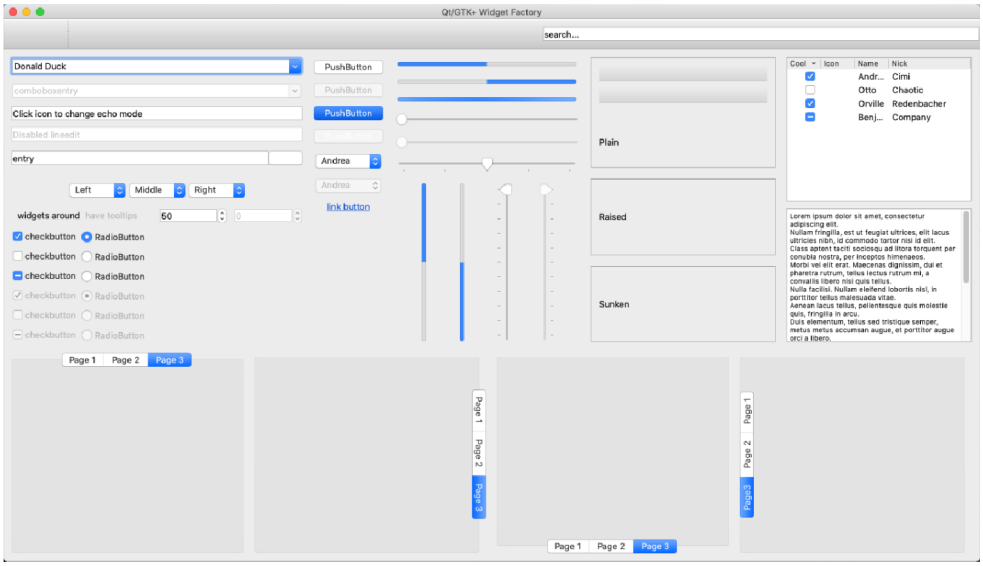


Obrázek 3.6: Windows nativní styl

3.3.3 macOS

Grafickým stylem macOS je *Aqua*, který je originálně založený na stylu vody. Jeho cíl byla „průhlednost, hloubka, zapadající barva do prostředí a komplexní textury vytvářející přitažlivé prostředí“. Typickými barvami tohoto stylu je bílá, šedá nebo modrá.

Pro zobrazení GUI, macOS používá zobrazovací protokol *Quartz*, který je často označovaný, jako tzv. *Core Graphics*. Ten se skládá ze dvou částí a to *Quartz Compositor* spolu s *Quartz 2D*, pomocí kterých je renderováno (Quartz 2D) a zobrazováno (Quartz Compositor) UI a také styl Aqua.



Obrázek 3.7: macOS nativní styl

Kapitola 4

Grafický framework Qt

Za framework, považujeme abstrakci souborů knihoven, doporučených postupů a návrhových vzorů. Framework poskytuje programátorovi již předdefinovanou strukturu souborů a složek. Cílem takových frameworků je usnadnit práci programátorům jak pokročilým, tak začátečníkům. Mezi aplikační rámce patří třeba Qt postavené na C++, Symfony na PHP, Django na Pythonu, nebo také JavaScriptovské rámce (Node.js, React či Angular) a mnoho dalších. Není ale podmínkou použití pouze jednoho frameworku, běžně najdeme kombinaci více frameworků, jako je třeba Django pro implementaci backendu, který je doplněný o Frontend implementovaný v Reactu. Většina těchto frameworků má na svých stránkách kvalitní dokumentaci pro snadné a rychlé porozumění použití daného frameworku.

Informace byly převzaty z knihy *Cross-Platform Development with Qt 6 and Modern C++* [3]. Qt je multiplatformní aplikační framework pro počítače, vestavěné systémy a mobilní zařízení. Podporuje platformy, jako je Linux, macOS, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS a další. Qt není programovací jazyk, je to framework napsaný v C++. Samotný framework a knihovny, mohou být kompilovány pomocí C++ kompilerů viz 3.2.1.

Informace byly převzaty z bakalářské práce Grafický prohlížeč a jednoduchý editor elf souboru [17]. Vlastní jazyk QML pro deklaraci vzhledu uživatelského rozhraní a nástroj MOC[1] (Meta-Object Compiler), který rozšiřuje C++ o sloty a signály, překlad řetězců a další. Knihovna má také dobrou dokumentaci, řadu modulů a díky její rozšířenosti je k dispozici velké množství návodů.

Jak již bylo uvedeno, Qt je primárně zaměřeno na programovací jazyk C++, existují ale také tzv. *bindings* (namapování knihovny z jednoho programovacího jazyka do druhého) pro jiné programovací jazyky, jako je Python (PySide[2]), Java (Jumbi), nebo C# (QtSharp). Framework Qt je velmi rozšířený v oblastech, jako je automobilní průmysl, zdravotní průmysl, spotřebitelská elektronika, či desktopové aplikace. Důkazem mohou být aplikace jako třeba Skype, Team Speak, VLC player, Virtual Box, Autodesk Maya a další. Qt úzce spolupracuje s firmami jako je Panasonic, AMD, Harman, Lg, Mercedes, nebo také Rimac[3].

[1] https://doc.qt.io/qt-5/moc.html
[2] https://www.qt.io/qt-for-python
[3] https://resources.qt.io/customer-cases?

4.1 Qt Quick

Zatímco Qt poskytuje rozšíření jazyka C++ a QML spolu s QtWidgets dovolují vytvářet programátorovi GUI, Qt Quick poskytuje uživateli potřebné prostředky pro vývoj QML aplikací. Je to knihovna, která obsahuje potřebné prostředí a možnosti při vytváření aplikací nad QML jazykem. Qt Quick umožňuje vývojářům a designerům vytvářet výkonné, plynulé a vizuálně atraktivní aplikace.

4.1.1 QML

QML je deklarativní programovací jazyk s lehce čitelnou syntaxí založenou na formátu JSON. Dále QML podporuje JavaScriptová rozšíření přímo v kódu, nebo v odděleném souboru. Informace byly převzaty z knihy *Mastering Qt 5* [10].

V podstatě by se mohlo říct, že je QML součástí Qt. Klade důraz na snazší návrh UI a rychlé prototypování, navíc je snadné se jej naučit, díky tomu je oblíbený mezi neprogramátory. Typickým příkladem jsou například designeři, kteří si mohou rychle vyzkoušet správnost svého navrženého GUI. Mezi hlavní zájemce o tento programovací jazyk, patří již výše zmiňované automobilky Mercedess a Rimac, které využívají QML při implementaci svých infotainment systémů (vestavěný systém převážně u automobilů, přes který jsou zobrazovány informace o vozidle a lze skrze něj ovládat jednotlivé funkce automobilu více ve článku [13]).

4.2 Výhody Qt

Mezi výhody Qt spadá také implementace vlastního editoru Qt Creator. Ten zjednodušuje práci s Qt aplikačním rámcem. Součástí Qt Creatoru je také Qt Designer a Qt Quick Designer. Pomocí Qt Quick Designeru, lze vytvářet GUI napsané v QML. Další možností je použít Qt Designer, ve kterém je možné vytvořit GUI pomocí Qt Widgetů. Další výhodou při psaní QML je jednoduchá a lehce pochopitelná syntax, velice rychlý vývoj prototypů a finálního produktu, nebo také možnost, vytvořit si své vlastní styly, v případě nespokojenosti.

Podle informací z knihy *Cross-Platform Development with Qt 6 and Modern C++* [3] patří mezi největší výhody možnost použití Qt na velkém počtu platforem. Díky tomu se ušetří čas programátorům a peníze firmy.

4.3 CMake - qmake

CMake je hojně používaný, multiplatformní open source nástroj pro kompilaci, testování a sestavení balíčků aplikace. Používá se pro generování konfiguračních souborů a makefilů. CMake je podporován většinou IDE, včetně Qt Creatoru, Visual Studia Code, či Visual Studia, ve kterých je již integrovaný.

Pro kompilaci Qt aplikací bylo do nedávna možné používat jejich vlastní sestavovací nástroj qmake. Qmake je kompilační nástroj od společnosti Qt, který je možné použít stejně jako CMake. Od nové verze Qt 6.0 byl tento dřívější výchozí sestavovací nástroj vyměněn kvůli vzrůstající popularitě CMake-u, který je aktuálně považován za „standard“. Více informací o CMake v knize *Mastering CMake* [15].

4.4 Distribuce Qt aplikací

V případě zájmu komunity o námi vytvořenou aplikaci, máme poměrně mnoho možností, kam tuto aplikaci distribuovat. Jednou z možností je použít vestavěný obchod námi zvoleného operačního systému. Tato možnost bývá ve většině případů velmi zdlouhavá a to z důvodu různých kontrol a přezkoumání aplikace.

Dynamické knihovny

Při distribuci aplikací a tvorbě balíčků je potřeba spolu se zdrojovým kódem přidat do balíčku potřebné závislosti, které se používají v programu aplikace a bez kterých by se aplikace nespustila. Takovéto závislosti mohou být např. různé pluginy a knihovny. Dynamické knihovny známé také jako „sdílené knihovny“ mají na operačním systému Windows typicky koncovku *.dll*, *.o* a *.so* na operačním systému Linux nebo *.dylib* na macOS, ve všech případech se jedná o jednu a tu samou věc. Sdílené knihovny se jím říká, z toho důvodu, že jsou sdílené mezi více programy. Jejich nevýhodou je, že jsou při spouštění pomalejší než statické knihovny, což se může zdát jako veliká nevýhoda. Tuto nevýhodu však dynamické knihovny poměrně dobře vyvažují díky tomu, že jsou pouze propojené se spustitelným souborem a jsou volané až v případě, kdy jsou potřeba.

Při sestavení aplikace ověří překladač, že jsou veškeré proměnné a funkce připojené k programu ve formě statických nebo dynamických knihoven. Pokud se jedná o dynamickou knihovnu, je při startu aplikace zavolán tzv. „zavaděč“ (dynamic loader), který se podívá, jaké dynamické knihovny jsou připojené k programu. Nahraje tyto knihovny do paměti, pokud se v ní nenachází a předá programu adresy, na kterých se knihovny nachází.

Pokud si vezmeme typický příklad, kdy dva a více programů používají stejnou knihovnu, což bývá poměrně běžná záležitost, je tato knihovna nahrána do paměti zařízení pouze jednou. Díky tomu se šetří paměť zařízení.

4.4.1 Flatpak

Další z možností jak distribuovat aplikace na Linuxové systémy, je použít *Flatpak*[4]. To je distribuovaný open source systém správy verzí. Tento systém používá *OSTree*[5], který se velice podobá technologii Gitu. Při stažení aplikace z Flatpaku je vytvořen na místní (lokální) síti úložiště (repozitář), do kterého se aplikace uloží. Následně jsou vygenerovány tzv. *hard-links* (pevné cesty), které slouží k namapování na místní souborový systém. Každá taková aplikace je poté spouštěná ve svém *sandboxu*, který je oddělen od ostatních sandboxů. Sandbox označuje uzavřené a bezpečné prostředí, které slouží pro spouštění software. To znamená, že při chybě aplikace je operační systém zcela ochráněn, což je obrovská výhoda. Navíc se tomuto prostředí dají podle požadavků aplikace přiřadit oprávnění pro svůj běh. Mezi další kladné vlastnosti spadá používání tzv. *runtime*, což je set knihoven, který je sdílený mezi více Flatpaky. Díky této vlastnosti nedochází k duplikaci knihoven a tím pádem k úspoře místa. Každá Flatpak aplikace bude fungovat úplně stejně na všech distribucích Linuxu, tohoto bylo docíleno právě díky runtime (aplikace jsou sestavené od základu a obsahují veškeré knihovny pro svou funkčnost). Každá Flatpak aplikace má své unikátní ID (identifier), ve tvaru `org.foo.bar`. Tento formát je ve stylu obráceného DNS záznamu z TCP/IP tzn., že je 1. část tvaru tzv. doména spravující projekt (2. položka), tento projekt

[4] https://flatpak.org/
[5] https://ostreedev.github.io/ostree/introduction/

je ještě upřesněn ve 3. části tohoto ID. V TCP/IP je podobně jako organizace za doménou „.cz“ zodpovědná za správu poddomén např. jako je „seznam.cz“.

```
#Instalace runtime
$ flatpak install org.kde.Platform//6.2
#Instalace balíčku org.fedoraproject.MediaWriter
$ flatpak install --user org.fedoraproject.MediaWriter.flatpak
#Spuštění aplikace
$ flatpak run --user org.fedoraproject.MediaWriter
#Odinstalace aplikace
$ flatpak uninstall --user org.fedoraproject.MediaWriter
```

Výpis 4.1: Správa nad Flatpak balíčkem včetně závislostí

4.4.2 Snappy

Druhou možností distribucí aplikací na operační systém Linux je platforma *Snappy*[6]. Ta je označována za alternativu k package managerům yum, apt nebo k Flatpaku. Snappy bylo původně navrženo pro Ubuntu Touch, což je mobilní platforma založená na distribuci Ubuntu. Samotné balíčky jsou nazývány "snaps" a nesou příponu `.snap`. Při aktualizaci snapů jsou prováděny inkrementální updaty, není tak třeba aktualizovat celý snap, navíc je možné tyto snapy aktualizovat automaticky. Výhodou při používání Snapy jsou tzv. rollbacky (vrácení aplikace do předchozí stabilní verze), které jsou použity v případě, že došlo při aktualizaci balíčku k chybě. Porovnání Flatpaku a Snappy můžeme najít na stránce FOSSLinux[7].

4.4.3 Chocolatey

Stejně jako jsou *yum*, *apt*, *dnf*, nebo *packman* package managery pro Linux, *Chocolatey* je package manager pro operační systém Windows. Chocolatey je stejně jako Flatpak, open source systém pro správu balíčků. Dokáže komplexní úlohy zjednodušit na jednotlivé řádkové příkazy. K tomu využívá výhody Windows Power Shellu a s jehož pomocí je možné spravovat instalátory, binární soubory, zipy, nebo kompilovat námi napsané aplikace. Po instalaci Chocolatey se tento package manager volá pomocí příkazu `choco`.

Podporuje všechny verze operačního systému Microsoft od Windows 7. Dále je možné Chocolatey používat na veškerých serverech Microsoft počínaje Microsoft server 2003. Chocolatey podporuje také servery jako je AWS od Amazonu, Microsoft Azure, a nebo Docker taktéž od Microsoftu.

4.4.4 Homebrew

Posledním package managerem, který si v této bakalářské práci ukážeme bude *Homebrew*, pro operační systém macOS, ten je taktéž dostupný i na Linuxu. Tento package manager je také známý jako „Chybějící package manager pro macOS“, nejspíše z toho důvodu, že na Apple Store můžeme najít typicky pouze aplikace s GUI. Homebrew je stejně jako ostatní package managery open source nástroj pro správu open source balíčků. Dalšími variantami

[6] https://snapcraft.io/

[7] https://www.fosslinux.com/42410/snap-vs-flatpak-vs-appimage-know-the-differences-which-is-better.htm

na macOS, mohou být například package managery *Fink* nebo *Nix*. Aby bylo možné Homebrew používat, je nutné mít nainstalované na svém operačním systému *Xcode's Command Line Tools*, což jsou základní nástroje, které jsou používány vývojáři pro práci s terminálem. Po instalaci Homebrew dostaneme nový příkaz `brew`, pomocí kterého spravujeme balíčky. Jedním z hlavních zdrojů balíčků je server GitHub, který poskytuje velikou škálu projektů a kontributorů, kteří do těchto projektů přispívají. Informace byly převzaty z knihy *Tweak Your Mac Terminal: Command Line macOS* [19].

Projekt Homebrew je v současnosti členem neziskové organizace *Software Freedom Conservancy*, do které patří open source projekty, jako jsou Inkscape, Git, QEMU, Wine nebo phpMyAdmin. Homebrew je dostupné od verze macOS High Sierra (10.13). Na svých stránkách[8] zároveň poskytují počítadlo stažených balíčků za určitou časovou periodu, díky tomu můžeme vidět, jak je balíček atraktivní pro uživatele.

4.4.5 Git

Tato standardní technologie je využívána v open source komunitě, která je běžně použitá na serverech jako je GitHub, Bitbucket, Gerrit a další. Pomocí těchto serverů je možné provádět kontroly kódu, zda neobsahuje chyby nebo se nedá kód optimalizovat či napsat přehledněji. Díky integrovaným službám jako jsou „merge requesty" může každý přispívat do vývoje aplikace. Tuto možnost (Git) můžeme vidět např. když u dané aplikace vidíme „Download from source", což v znamená „Stáhnout zdrojové kódy aplikace". V takovém případě musí uživatel provést sestavení aplikace lokálně. Více informací o Gitu v knize Version Control with Git [12].

CI

Jednou z výhod použití GitHubu je možnost provádět různé reakce na události tzv. CI (anglicky continous integration nebo Workflow). Díky této schopnosti je možné provést různé operace jako reakce na události, např. příchozí „commit" nebo tvorba „pull requestu". Při úspěšném sestavení na specifikovaných platformách jim bude vytvořen příslušný balíček např. Flatpak pro Linux, nebo .exe pro Windows. Celá konfigurace spočívá ve dvou krocích, první část spočívá v konfiguraci `ccpp.yml` souboru, neboli tzv. „CI manifestu", který se nachází v adresáři `.github`. V tomto manifest souboru se specifikují platformy, na které chceme distribuovat, včetně postupu při sestavení aplikace spolu se závislostmi pro danou platformu.

```
steps:
- name: Install dependencies
    if: github.event_name == 'push'
    shell: bash
    run: |
        choco install nsis
        choco install dos2unix
```

Výpis 4.2: Zjednodušená struktura části CI manifestu na serveru GitHub

[8] https://brew.sh/

Flatpak

Druhou částí při tvorbě balíčku Flatpak je vytvoření speciálního souboru, který bude sloužit jako takový předpis. V našem případě se jedná o soubor `org.fedoraproject.MediaWriter` také tzv. *manifest*, ten je použitý pro konfiguraci instalačního souboru. Zároveň se v tomto souboru specifikují potřebná oprávnění pro správné fungování aplikace spolu s potřebnými závislostmi (*kde.Platform*[9] a *kde.Sdk*[10]) pro sestavení Flatpak balíčku. Dále je v tomto souboru potřeba doplnit knihovny, které nejsou běžnou součástí prostředí kde.Platform a kde.Sdk, v našem případě se jedná o knihovnu Adwaita-qt. Flatpak balíček je vytvořen pomocí nástroje `flatpak-builderu`, kterému je předán tento manifest soubor. Vytvořený balíček obsahuje spustitelnou aplikaci se všemi jejími závislostmi, které byly specifikované v manifest souboru. Poslední částí při tvorbě Flatpak balíčku je přiřazení jeho ID, v našem případě *org.fedoraproject.MediaWriter*. Více informací o tvorbě Flatpak balíčku pomocí GitHub workflow na stránkách Flatpaku[11].

```
"runtime": "org.kde.Platform",
"runtime-version": "6.2",
"sdk": "org.kde.Sdk",
"finish-args": [
    #přístup k základním zdrojům, které aplikace běžně vyžadují
    #práce se sítí
    #práce s úložištěm
],
"modules": [
    #knihovny které nejsou součástí Platform a Sdk
    #samotná aplikace MediaWriter
]
```

Výpis 4.3: Zjednodušená struktura konfiguračního souboru pro Flatpak

Windows

Při tvorbě spustitelného souboru .exe na Windows, se podobně jako u Flatpaku využívá jak manifest souboru, tak i *NSIS* (Nullsoft Scriptable Install System) souboru, který se využívá na tvorbu instalačních balíčků pro Windows, více v knize *A NSIS Developer's Notebook* [20]. Celý postup vytvoření instalačního balíčku je vykonáván po částech v souboru *ccpp.yml*. Mezi tyto části patří instalace závislostí aplikace pomocí nástroje chocolatey nebo také kopírování potřebných QML souborů. Při injekci potřebných QML souborů lze využít nástroje `windeployqt`, který slouží k automatizaci vytváření instalačního balíčku. Musíme si ale dávat pozor, protože windeployqt nakopíruje pouze Qt knihovny (Qt6Core.dll a Qt6Quick.dll), Qt pluginy, jako jsou qsvg.dll nebo qwindows.dll a QML moduly např. Controls.2. Pro vytvoření kompletního balíčku musíme ještě ručně nakopírovat systémové knihovny, jako jsou knihovny pro práci s C++, SSL a jiné.

Zjednodušená struktura adresáře na Windows obsahující Qt aplikaci:

```
Fedora Media Writer
└─ platforms
```

[9] Základní knihovny potřebné pro spuštění aplikace
[10] Vývojářské knihovny pro sestavení balíčku
[11] https://docs.flatpak.org/en/latest/building.html

```
│   └── qwindows.dll
├── QtQml
│   ├── WorkerScript
│   └── qmlplugin.dll
├── styles
│   └── qtwindowsvistastyle.dll
├── helper.exe
├── mediawriter.exe
└── Qt6Core.dll
```

Jakmile máme všechny závislosti nakopírovány vytvoříme spustitelný soubor .exe pomocí nástroje `makensis`, který použije soubor NSIS zmíněný výše a manifest soubor. Posledním krokem je podepsání výsledného instalačního balíčku a nahrání jej na daný server, např. GitHub. Více informací o tvorbě balíčku na Windows na stránkách Qt[12].

macOS

Tvorba spustitelného souboru .dmg[13] pro macOS se příliš neliší od tvorby spustitelného souboru na Windows s tím rozdílem, že se používá nástroj Homebrew. Proces vytvoření spustitelného balíčku je implementován ve skriptu, ze kterého se volají pouze funkce. Při tvorbě .dmg balíčku se podobně jako na operačním systému Windows používá nástroj od Qt, v tomto případě se jedná o nástroj `macdeployqt`. Při použití tohoto nástroje se volá rekurzivně nástroj `otool`, pomocí kterého můžeme zjistit potřebné závislosti aplikace, pluginů nebo knihoven.

U kopírovaných závislostí pomocí macdeployqt se implicitně volá vestavěný nástroj `install_name_tool`, který pracuje ve dvou módech a to `change`, která nastavuje cestu spustitelným souborům (helper a mediawriter) k jejich závislostem v tzv. „executable_path“[14]. Pokud by se nepřenastavila cesta, aplikace by hledala své závislosti v adresáři */usr*, i když jsou všechny potřebné závislosti ve složce s aplikací. Toto nemusí být problém pro uživatele, kteří mají všechny závislosti právě v adresáři /usr, problém by nastal v případě, kdy by uživatel spouštěl aplikaci a v adresáři /usr potřebné závislosti neměl. Druhým módem tohoto nástroje je mód `id`, která slouží ke změně ID u knihoven a frameworků. Pokud bychom kopírovali dodatečné závislosti, bylo by potřeba explicitně nastavit cestu k souborům Fedora Media Writer a helper pomocí nástroje install_name_tool.

Důležitým souborem pro tvorbu balíčku na macOS je soubor Info.plist (Information property list), který je ve formátu XML a obsahuje data pro macOS balíček, jako je název, ikona, identifikátor a další.

Zjednodušená struktura adresáře na macOS obsahující Qt aplikaci:

```
Fedora Media Writer
└── Contents
    ├── Frameworks
    ├── MacOS
    │   ├── mediawriter
    │   └── helper
    └── Resources
```

[12] https://doc.qt.io/qt-6/windows-deployment.html
[13] Souborový kontejner pro macOS aplikace
[14] Cesta k spustitelnému souboru

```
├── PlugIns
└── Info.plist
```

Další částí tvorby .dmg balíčku je podepisování binárních souborů, včetně knihoven a frameworků, pomocí příkazu `codesign`. Díky tomu vědí uživatelé odkud aplikace, kterou spouští ,pochází. Po podepsání je možné vytvořit balíček .dmg, toho je dosaženo nástrojem `hdiutil`. Více informací o distribucí na macOS na stránkách Qt[15].

Abychom mohli náš spustitelný balíček distribuovat, je potřeba provést tzv. notorizaci. To je proces, při kterém se pošle aplikace firmě Apple, jenž provede nad aplikací automatizační proces kontroly. Do této kontroly patří sekvence operací, mezi které patří kontrola, zda aplikace neobsahuje škodlivé komponenty a jestli jsou všechny části kódu správně podepsány. Při úspěšné notorizaci je vygenerován tzv. „ticket“, který se připojí k aplikaci a zároveň je nahrán na patřičný server, ze kterého může uživatel ověřit věrohodnost aplikace. Následuje už jen distribuce aplikace na vybraný server. Informace byly převzaty ze stránek Apple[16].

4.5 Novinky v Qt6 oproti Qt5

Na konci roku 2020 vyšla nová majoritní verze Qt a to Qt 6.0. Tato nová verze přináší mnoho novinek, ze kterých hojně čerpám v této bakalářské práci. Za největší změnu v QML považuji podporu nativního stylu operačních systémů Windows a macOS.

Tedy doposud mělo QtWidget a QML výchozí styl Fusion, který byl používán jako tzv. *fallback* (záchytný styl). Pro QtWidget aplikace napsané v C++ měly operační systémy Windows a macOS své nativní styly „windows“ a „macintosh“. Nativní styl u operačního systému Linux neexistuje, protože každá distribuce Linuxu má svůj hlavní styl viz Fedora.

Mezi další velké novinky a vylepšení můžeme zařadit podporu nové verze C++17, nebo změnu hlavního sestavovacího nástroje za Cmake a další [2].

[15] https://doc.qt.io/qt-6/macos-deployment.html

[16] https://developer.apple.com/documentation/security/notarizing_macOS_software_before_distribution

Kapitola 5

Fedora Media Writer - aktuální podoba

Abychom mohli nainstalovat jednu z mnoha edicí Fedory (Workstation, Server, IoT, Silverblue a mnoha dalších), budeme potřebovat stáhnout správný soubor `.iso`, ze kterého budeme muset vytvořit bootable USB. Zapsání tohoto souboru na USB disk lze docílit pomocí velkého množství různých aplikací, jako může být například aplikace Rufus, nebo právě Fedora Media Writer. Tato kapitola rozebírá funkce, vzhled a návrh nové generace tohoto nástroje.

5.1 Funkce

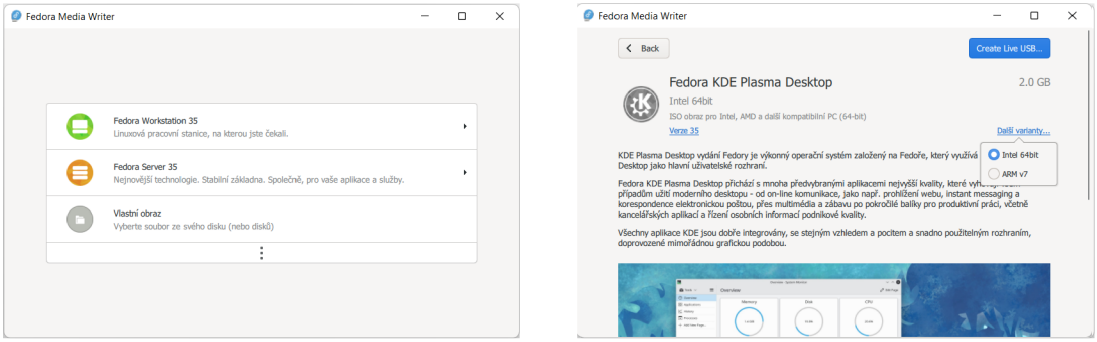
Aby byl celý proces tvorby bootovacího USB zjednodušen, byla vytvořena aplikace Fedora Media Writer. Pomocí této aplikace je uživatel schopen vytvořit toto USB s libovolným .iso souborem, navíc je aplikace schopna stáhnout uživatelem vybranou edici Fedory včetně různých architektur jako je Intel-64, ARM64, nebo Raspberry Pi. Součástí aplikace je také obnova USB disku, na kterém je tzv. živá verze Fedory. Pro vytvoření USB disku Fedora potřebujete tři různě velké oddíly a jejich speciální rozložení - tento stav není v některých nástrojích jednoduché změnit zpět na jediný oddíl, který se běžně používá.

Fedora Media Writer vychází z původního LiveUSB Creatoru, ze kterého byly převzaty části jeho funkčního jádra a zároveň přidává větší spolehlivost a možnost obnovy disku do čistého stavu s jedním oddílem. Informace byly převzaty z oficiálních stránek Moje Fedora[1].

5.2 Vzhled

Současná verze Fedora Media Writeru je napsaná v QML Qt5, která nepodporuje nativní styly operačních systémů Windows a macOS. Z tohoto důvodu bylo potřeba vybrat a naimplementovat jednotný styl ovládacích elementů aplikace (Controls), jaký bude Fedora Media Writer mít, tedy styl Adwaita viz obrázek 3.5. Tento styl je součástí aplikace a je instalován s ní. Aplikace vypadá tedy úplně stejně na operačním systému Windows, macOS a všech distribucí Linuxu.

[1] https://mojefedora.cz/fedora-media-writer-nastroj-na-vytvareni-bootovacich-flash-disku/



Obrázek 5.1: Fedora Media Writer

5.3 Instalace

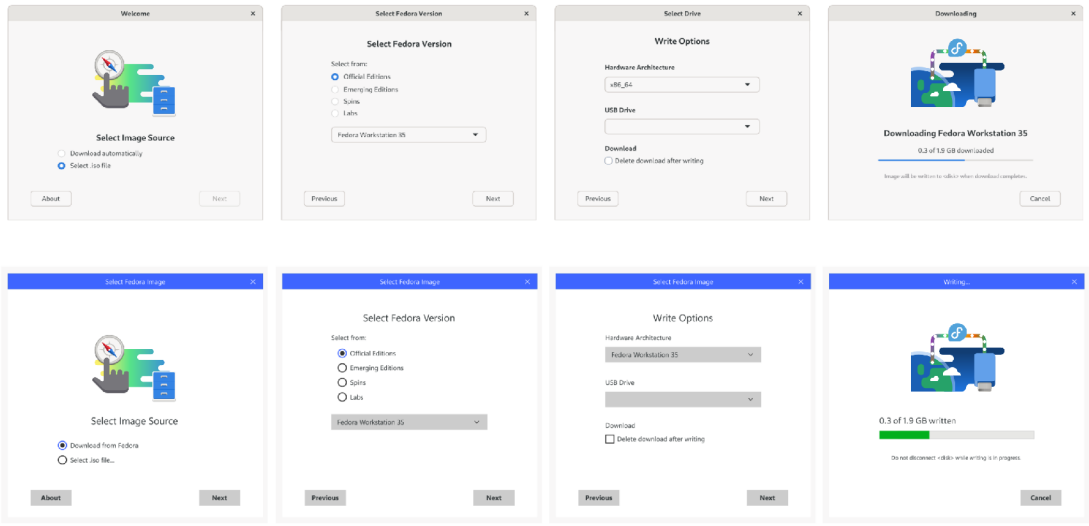
Díky tomu, že je Fedora Media Writer multiplatformní, jej můžeme nainstalovat na více operačních systémů viz Distribuce Qt aplikací. Navíc je tato aplikace open source, díky tomu máme možnost stáhnout samotné zdrojové soubory z GitHubu. Následně je možné nástroj sestavit a nainstalovat včetně veškerých závislostí. Tuto možnost bych však doporučil spíše zkušenějším uživatelům. Následnou instalaci operačního systému RHEL 8 můžeme najít v například v knize Red Hat Enterprise Linux 8 Essentials [21].

Pokud se rozhodneme instalovat nástroj Fedora Media Writer na operační systém Linux, máme poněkud více možností než na Windows a macOS, jak tento nástroj nainstalovat. Pro uživatele začínající s Linuxem bych spíše doporučil použití package manageru RPM nebo Flatpaku viz 4.4.1.

Pokud se rozhodneme použít tento nástroj na Windows, budeme si muset stáhnout spustitelný .exe, nebo .dmg na macOS, soubor ze stránek Fedory. Po stažení se nástroj nainstaluje a je připravený k používání.

5.4 Návrh nového GUI

Jak již bylo zmíněno výše, hlavním důvodem přepracování nástroje Fedora Media Writer je podpora nativního vzhledu ovládacích elementů na operačních systémech Windows a macOS. K tomu nám pomůže nová verze Qt.



Obrázek 5.2: Mockup nového Fedora Media Writeru pro Windows a Linux

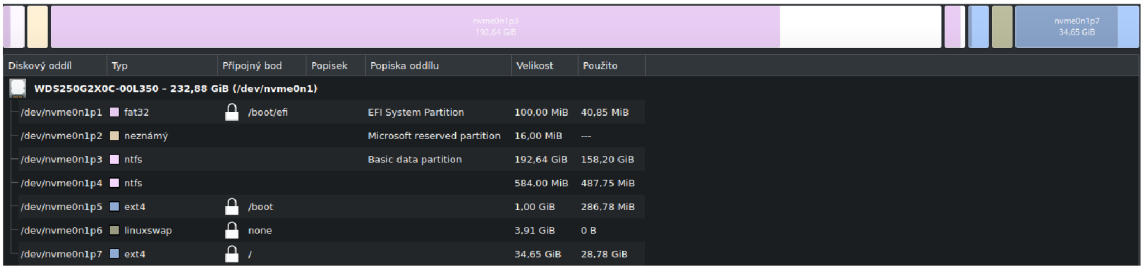
Na obrázku výše můžeme vidět mockup návrhu nového GUI, pro operační systém Linux a Windows, který byl navržen designery a příznivci komunity[2].

5.5 Současné nedostatky aplikace

Kromě současného stylu aplikace má Fedora Media Writer další nedostatky, které spočívají v nespolehlivé obnově USB disků na operačním systému Windows pomocí nástroje `diskpart`. Proto se zaměříme na tento problém, který pravděpodobně spočívá v druhu zápisu souboru .iso na USB disk. Při zápisu tohoto souboru na disk se přepíše současná tabulka oddílů (angl. partition table). Další nahlášené buggy pro opravu můžeme najít na Githubu MediaWriteru[3].

5.5.1 Oddíly

Každý disk je možné rozdělit na jeden či více oddílů. Tento pojem nese označení fyzická část disku, který může být naformátován s různými souborovými systémy (angl. file system) a sloužit k různým potřebám.



Obrázek 5.3: Oddíly dual-bootu na SSD NVMe disku

[2] https://github.com/FedoraQt/MediaWriter/issues/275
[3] https://github.com/FedoraQt/MediaWriter/issues?page=1&q=is%3Aissue+is%3Aopen

Na obrázku výše můžeme vidět rozložení oddílů na disku. Tyto oddíly jsou označené přípojným bodem např. */dev/nvme0n1*, kde je */dev* soubor reprezentující zařízení a názvem disku */nvme0n1* na kterém se nachází. Tento přípojný bod je navíc rozšířený o tzv. koncovku (suffix), která reprezentuje číslo oddílu např. *p1*. Tento disk obsahuje operační systém Windows, pro který jsou jeho oddíly naformátované se souborovým systémem NTFS, FAT16 nebo FAT32. Současně tento disk obsahuje operační systém Linux, pro který je typický souborový systém ext4, ext3 nebo linuxswap.

MBR

V dnešní době se nejvíce setkáme se stylem oddílů DOS od Microsoftu. Tento styl udává, jak bude disk obsahující tento styl rozčleněn na jednotlivé oddíly pomocí oddílů tabulek (partition table). Jednou z těchto tabulek je MBR (Master Boot Record), který byl vyvíjený v 80. letech. Z důvodu jeho stáří je spíše zaměřený na menší a starší disky, používá se na hardwarové architektuře Intel IA32 (tj. i386 / x86) a můžeme ji nalézt ve Windows, Linuxu nebo operačních systémech založených na FreeBSD, jako je třeba macOS. Tato tabulka oddílů nepodporuje více než čtyři oddíly na jednom disku, zároveň nepodporuje disky s kapacitou větší, než jsou 2TB. Disk, který je organizován jako styl DOS a obsahuje tabulku oddílů MBR, obsahuje v prvních 512 bytech sektoru disku informace o:

- Počtu sektorů v oddílu
- Adresu, na které nalezne operační systém
- Typ oddílu
- Různé příznaky (flagy), např. zda je oddíl bootovací

GPT

Další a novější tabulkou oddílů je GPT, ta se používá na hardwarové architektuře Intel 64-bit. Zároveň disky obsahující GPT neobsahují BIOS (Basic Input/Output System), jako disky s MBT, ale EFI (Extensible Firmware Interface), které se používají na moderních operačních systémech (Windows 8 a novější). Mezi výhody, oproti MBR, patří např. podpora až 128 oddílů, rychlejší bootování, podpora neomezeně velkého diskového oddílu nebo také zálohy důležitých datových struktur při selhání. V současné době můžeme v moderních systémech a na serverech s velkými disky převážně najít disky obsahující GPT.

Více informací o discích a souborových systémech můžeme najít v knize *File System Forensic Analysis* [1].

5.5.2 Současný postup při obnově

Při pokusu o obnovení a naformátování disku se souborovým systémem FAT32, jak již bylo zmíněno výše v 5.5, je při zápisu přepsaná tabulka oddílů. Díky tomu dělá nástroji diskpart problém s tímto diskem téměř jakkoliv pracovat.

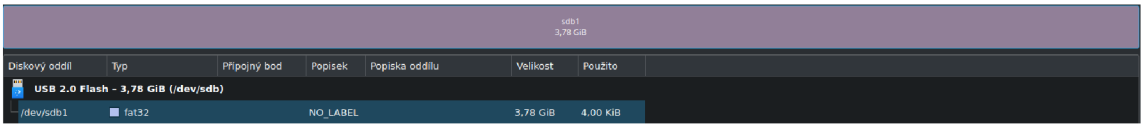
```
$ select disk X
$ clean
$ create partition primary
$ select part 1
$ format fs=fat32 quick
```

```
$ assign
```

Výpis 5.1: Obnova disku pomocí nástroje diskpart

Při pokusu o obnovení disku můžeme dostávat chybové zprávy, jako je například „Nemůžu najít požadovaný soubor“. Z toho můžeme odvodit, že problém bude nejspíše v tabulce oddílů. Zároveň můžeme vidět v různých diskových nástrojích označení souborového systému jako *RAW*. Toto označení není souborový systém, je to druh chyby, která se vyskytuje v případě, že je souborový systém na disku poškozený.

Na internetu můžeme najít mnoho diskuzí na téma nefunkčnosti obnovení disku pomocí Fedora Media Writeru, ve kterých můžeme najít spoustu spolehlivých alternativ pro obnovu disku, jako je třeba Rufus nebo EraseUS na Windows, nebo fdisk, či gparted na operačním systému Linux.



Obrázek 5.4: Správně naformátovaný disk bez živé Fedory

Na obrázku výše můžeme vidět úspěšně naformátovaný disk s partition tabulkou MBR a souborovým systémem FAT32.

5.6 Rufus

Je open source nástroj na čištění, formátování a vytváření bootovacích disků. Je vyvíjen Petem Batardem jako volnočasový projekt již od roku 2011 a je dostupný pouze na operačním systému Windows.

Jeho výhodou v porovnání s Fedora Media Writerem je poměrně široká škála možností nastavení disku, jako je vybrat si typ tabulky oddílů, zvolit si velikost clusteru[4] a další. Takové pokročilé možnosti Fedora Media Writer sice nemá, ale je dostupný na všech operačních systémech, nejen na Windows. Další výhodou nástroje Rufus je možnost vytvoření tzv. multibootu[5]. Díky těmto funkcionalitám je nástroj Rufus poměrně komplexní.

Zápis souboru

Při zápisu pomocí nástroje Rufus se nejdříve provede odstranění veškerých dat na USB disku včetně tabulky oddílů a veškerých oddílů. V dalším kroku se vytvoří nová tabulka oddílů, kterou si uživatel zvolil v nabídce, včetně naformátování na výchozí souborový systém tj. FAT32. Tuto skupinu kroků lze zvolit i samostatně pro vymazání dat na disku a vytvoření nového souborového systému. Posledním krokem je již pouze kopírování souboru .iso na disk.

Jakmile je nahraný soubor .iso na USB disku pomocí nástroje Rufus, je USB disk složen z jednoho primárního oddílu se souborovým systémem FAT32. Zároveň tento souborový systém obsahuje zapsaný soubor *Fedora-Live.iso*, který má následující strukturu:

```
Fedora-Live.iso
└── EFI
```

[4] Skupina adresovatelná pomocí ID, skládající se ze sektorů
[5] Disk obsahující více bootovatelných operačních systémů

```
├── isolinux
├── LiveOS
│   ├── squashfs.img
│   └── livecd-iso-to-disk
└── images
```

Ze struktury výše můžeme vidět, že obsahuje složky `EFI` a `isolinux`, které obsahují soubory potřebné pro bootování systému. Dále obsahuje složku a `LiveOS`, která obsahuje skript `livecd-iso-to-disk`, pomocí kterého se zapisuje operační systém na USB disk. Soubor `squashfs.img` je souborový systém typu *SquashFS*[6]. Tento soubor po rozbalení obsahuje samotný operační systém uložený v souboru *rootfs.img*.

[6]Kompresovaný souborový systém pouze pro četní, kompatibilní pouze s operačním systémem Linux

Kapitola 6

Implementace

V této části bakalářské práce se zaměřuji na zajímavé či problematické části při implementaci. V první části ukazuji, jak je celá aplikace členěna, dále popisuji jednotlivé operace s aplikací. Celá implementace vychází z kapitoly 5.

6.1 Struktura aplikace

Aplikace má následující adresářovou strukturu:

```
MediaWriter
├── src
│   ├── app
│   │   └── qml
│   ├── helper
│   ├── theme
│   └── translations
├── dist
│   ├── mac
│   ├── flatpak
│   └── win
└── po
```

Složka dist

Tato část aplikace se stará o distribuci a vytváření buildů na operační systémy Windows, macOS a Linux v podobě Flatpaku. Každá platforma má vytvořený vlastní soubor, který se používá buď jako předpis, nebo skript pro vytvoření aplikace na danou platformu. Více informací o distribuci v Distribuce Qt aplikací.

Složka po

V této části aplikace je implementována funkce překladu aplikace do jazyku země, ve které se uživatel nachází. Překládané řetězce jsou „označeny“ pomocí funkce *qsTr*, které jsou extrahovány z aplikace pomocí nástroje *lupdate*, který tyto řetězce uloží do .ts souboru. Lokalizační údaje resp. zeměpisná oblast je získávána pomocí funkce *Qt.locale()*, díky této

funkci aplikace ví do jakého jazyka se má přeložit. Překlad je stahován z internetu[1] a překládány komunitou pod open source licencí, tyto překlady jsou ukládány do .po souborů, které vypadají následovně:

```
#: ../src/app/downloadmanager.cpp:215
msgctxt "Download|"
msgid "Unable to fetch the requested image."
msgstr "Nepodařilo se získat požadovaný obraz."
```

Výpis 6.1: Příklad souboru .po

Odtud je překlad kopírován do .qm souborů ve složce *src/translations* a následně je překlad zobrazován v aplikaci.

Složka src

Pro ovládací elementy, které jsou použité pro operační systém Linux, je styl Adwaita uložen ve složce *theme* viz 2.1.1.

Téměř celá logika aplikace je uložená ve složkách *app* a *helper* v souborech `.cpp`. V těchto souborech je implementováno stahování a ukládání souboru, file dialog pro operační systém Linux nebo práce s USB diskem. Kvůli rozdílné práci operačních systémů s USB diskem jsou vytvořeny různé varianty těchto souborů.

Hlavní a také největší částí této bakalářské práce je předělání designu, který je implementován ve složce *app/qml*. Celé rozložení (anglicky layout) aplikace se skládá ze 2 hlavních částí. A to z tzv. *StackView*, ve kterém se zobrazují jednotlivé stránky. Toto StackView je doplněno o dvojici tlačítek tedy *prevButton* a *nextButton*, které slouží pro vykonání operací na jednotlivých stránkách včetně přecházení mezi těmito stránkami.

Pro pohodlné přecházení mezi stránkami bylo využito jedné z mnoha výhod QML a to stavy (anglicky states). Každá stránka je reprezentovaná svým stavem, tyto stavy mohou upravovat jednotlivé elementy na dané stránce, typicky text nebo zobrazení elementů pomocí své vlastnosti (anglicky property), tedy *PropertyChanges*. Při přechodu do stavu je možné spustit skript pomocí vlastnosti *StateChangeScript*.

```
State {
    name: "drivePage"
    when: selectedPage == Units.Page.DrivePage
    PropertyChanges { target: mainWindow; title: qsTr("Select drive") }
    PropertyChanges {
        target: nextButton
        visible: true
        onClicked: {
            selectedPage = Units.Page.DownloadPage
            if (selectedOption != Units.MainSelect.Write)
                releases.variant.download()
            drives.selected.setImage(releases.variant)
            drives.selected.write(releases.variant)
        }
    }
    StateChangeScript {
```

[1] https://translate.fedoraproject.org/projects/fedora-media-writer/mediawriter/

```
        script: { stackView.push("DrivePage.qml") }
    }
```

Výpis 6.2: Příklad použití stavu pro stránku

6.1.1 Stránky

Aplikace je tvořena pěti stránkami a každá z nich má vlastní soubor. Jedná se o soubory *MainPage.qml*, *VersionPage.qml*, *DrivePage.qml*, *DownloadPage.qml* a *RestorePage.qml*, které jsou zobrazovány v komponentě `StackView`. Tato komponenta funguje laicky řečeno na principu „štosu papíru“. Při změně stránky je nová stránka „pushnuta“ (přidána) nad předchozí, v opačném případě je stránka „popnuta“ (odebrána) a je zobrazena stránka předchozí.

6.1.2 Dialogy

Aplikace je doplněna dvěma dialogy tedy *AboutDialog.qml*, který obsahuje informace o aplikaci, jako je verze aplikace nebo odkaz na nahlašování bugů. Pomocí dialogu *CancelDialog.qml* je možné ukončit stahování nebo zápis souboru, jakmile se aplikace dostane do stavu „write_verifying“ je uživatel pomocí tohoto dialogu upozorněn na to, že může tuto operaci ukončit. Jedná se totiž o pouhou kontrolu, zda byla data korektně zapsána, není tedy nutné provést kontrolu celou. Oba tyto dialogy jsou vytvořeny z komponenty *ApplicationWindow*. Aby se předešlo nechtěným interakcím s aplikací, jsou tyto dialogy modální, tzn. hlavní okno zůstává viditelné, ale zablokované vůči jakékoliv interakci do té doby, než je dialog zavřený.

6.2 Výběr operace

Při spuštění aplikace se uživateli zobrazí hlavní stránka (MainPage). Na této stránce se vybírá jedna ze tří možností, které lze s aplikací provádět. První možností je stáhnutí jedné z několika variant Fedory v podobě souboru .iso a jeho následné zapsání na USB disk. Další možností je zapsání již staženého souboru .iso (nejen Fedory) na USB disk. Poslední možnost je dostupná pouze v případě, zda je k počítači připojen USB disk, který již obsahuje živý operační systém. V případě vybrání takové možnosti je USB disk naformátován.

6.2.1 Stahování souboru

Pokud je vybrána možnost „Download automatically“, je aktuální stránka nastavena na VersionPage. Na této stránce se zobrazují jednotlivé informace o různých variantách Fedory. Uživatel má možnost vybrat si jednu ze 4 hlavních variant (Official Editions, Spins, Labs nebo Emerging Editions).

Filtrace

Tyto informace jsou stažené ve formátu JSON a obsahují položky pomocí kterých je možné stáhnout danou variantu Fedory s danou architekturou.

```
"version":"34",
"arch":"x86\_64",
"link":"https://download.fedoraproject.org/pub/fedora/linux/...",
"variant":"Spins",
```

```
"subvariant":"KDE",
"sha256":"5f5a25271dd50c222b58d128ab3f31392e685f7a4408cb3e09b99484...",
"size":"2128625664"
```

Výpis 6.3: Příklad zápisu varianty Fedory ve formátu JSON

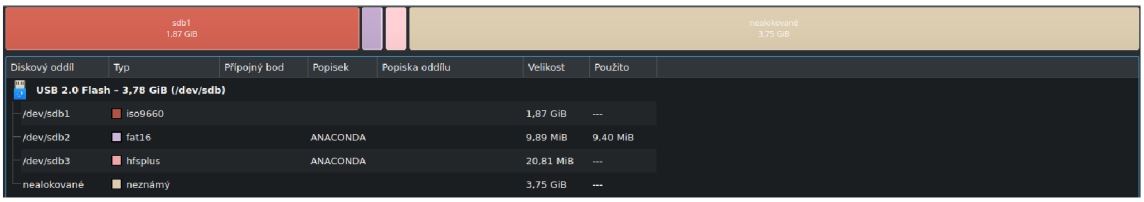
Po vybrání kategorie se filtrují z celého JSON souboru takové varianty, které odpovídají nastavenému filtru. Tato filtrace se provádí ve funkci `filterAcceptsRow`, na základě vybrané varianty a subvarianty (varianty), která vrací `true`, nebo `false`, podle toho zda daná varianta patří do nastaveného filtru.

Po vybrání varianty je v aplikaci změněna aktuální stránka na DrivePage, na které se vybírá podporovaná verze[2] (v době psaní této práce se jedná o verze 34, 35 nebo 36 Beta) a architektura (Intel64, ARM64 a další) v jaké má být vybraná varianta Fedory stažena. Na této stránce se zároveň vybere disk USB, na který se vybraná varianta Fedory v podobě souboru .iso zapíše.

Samotné stahování je rozděleno do několika kroků. Každý krok má definovaný svůj stav, podobně jak u jednotlivých stránek. S tím rozdílem, že aplikace mezi těmito stavy přechází automaticky, podle toho v jaké fázi se stahování souboru právě nachází. Mezi jednotlivé stavy patří také chybové stavy. Téměř všechny stavy informují uživatele o průběhu stahování a zápisu souboru patřičnými zprávami.

6.2.2 Zápis lokálního souboru

Při vybrání zápisu souboru .iso, je uživatel přesměrován rovnou na stránku DrivePage, na které mu bude umožněno vybrat soubor .iso uložený v jeho lokálním úložišti. Díky tomu, že se při stahování varianty Fedory soubor uloží lokálně a až poté je zapisován. Poté je možné tento soubor znovu vybrat pro zápis na USB disk. Na operačním systému macOS a Windows se používají nativní dialogy pro výběr souboru. Operační systém Linux je v tomto ohledu výjimkou, má implementován speciální dialog, z důvodu udržení celistvosti systému na Linuxu, podobně jako je tomu u stylu Adwaita. Na této stránce tedy DrivePage má uživatel možnost nechat vymazat soubor .iso po jeho zápisu na disk.



Obrázek 6.1: Disk obsahující soubor .iso nahraný pomocí Fedora Media Writeru

Stahování a následující zápis se na všech platformách snaží napodobit princip Linuxového příkazu `dd`, který se používá k převodu a kopírování souborů. Při kopírování souboru .iso na disk je daný disk rozdělen na 3 části, které obsahují soubor.iso. Tím je naprosto stejný jako Rufus viz 5.6, s tím rozdílem, že v případě Fedora Media není soubor .iso zapsán do jediného souborového systému. Soubory pro zápis jsou implementovány v samostatném nástroji helper, který je napojený na Fedora Media Writer.

[2]Každá verze je podporována přibližně po dobu 13-ti měsíců, po tuto dobu dostává pravidelné bezpečnostní záplaty

6.2.3 Obnova disku

Pokud je připojen disk s bootovacím diskem objeví se na hlavní stránce možnost tento disk obnovit. Proces obnovy disku má podobně jako stahování a zápis souboru .iso, množství kroků reprezentovanými stavy, mezi kterými aplikace jako v případě stahování přechází automaticky. Samotný proces obnovy disku je umístěn v souboru *restorejob.cpp*, který obsahuje patřičnou implementaci na základě operačního systému, na kterém běží. V případě operačního systému Linux se používá nástroj *udisk2*, který je součástí balíčku Flatpak, není tedy již potřeba nic instalovat. Na operačních systémech Windows a macOS se používají již vestavěné aplikace a to *diskpart* na Windows nebo *diskutil* na macOS.

Oprava obnovy disku na Windows

Jednou z částí zadání této práce byla oprava nespolehlivé obnovy USB disku na operačním systému Windows. Jak jsme se dozvěděli výše viz 5.5. Problém s obnovou disku byl nejspíše způsobený formou zápisu souboru .iso na disk, který přepíše tabulku oddílů.

```
$ select disk X
$ clean
$ convert gpt
$ convert mbr
$ create partition primary
$ format fs=fat32 quick
$ assign
```

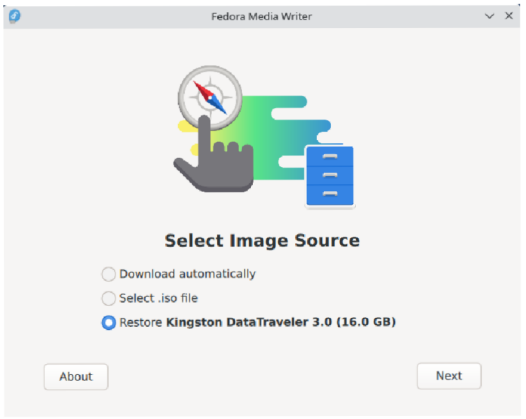
Výpis 6.4: Nový způsob obnovy disku pomocí nástroje diskpart

Celý proces se příliš neliší od původní implementace viz 5.5.2, hlavním rozdílem jsou příkazy `convert gpt` a `convert mbr`. Příkaz `convert gpt` bylo nutné použít z důvodu vytvoření nové tabulky GPT oddílů. Tohoto výsledku nelze dosáhnout pomocí příkazu `convert mbr`, důvodem je, že daný disk již tuto tabulku obsahuje, i když přepsanou souborem .iso. Pomocí příkazu `convert mbr` tak zajišťujeme zpětnou kompatibilitu se staršími USB disky. Po tomto kroku je proces stejný, oproti původní implementaci, tedy vytvoření nového oddílu a její naformátování na požadovaný souborový systém tedy FAT32.

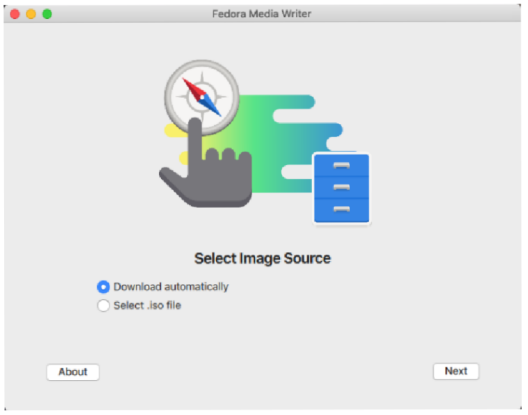
6.3 Výsledná aplikace

Výsledná aplikace je do velké míry vytvořená podle mockupu vytvořeného grafickými návrháři. Oproti vytvořenému mockupu bylo pozměněno vybírání verze Fedory, které mělo být původně na stránce VersionPage spolu s výběrem subvarianty (KDE, Workstation, Silverblue, atd.). Tato možnost by byla ale velice nepraktická a nepřehledná z důvodu velkého množství variant v comboboxu (Official editions, Spins, Labs, atd.) na stránce. Z toho důvodu byla tato možnost, tedy výběr stahující verze přesunuta na DrivePage spolu s výběrem architektury.

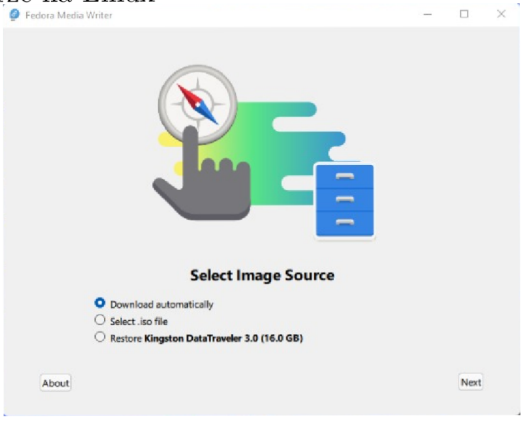
Jelikož byla aplikace implementována na základě mockupu, který neobsahuje detailní návrh, bylo nutné si podstatné části aplikace (dialogy a pozice tlačítek) po diskuzi s technickým vedoucím této práce, správně navrhnout, tak aby byla aplikace co nejvíce uživatelsky přívětivá. Na obrázcích níže můžete vidět úvodní stránku tj. MainPage, na všech operačních systémech.



Obrázek 6.2: Verze na Linux



Obrázek 6.3: Verze na macOS



Obrázek 6.4: Verze na Windows

Kapitola 7

Testování

Tato kapitola se zabývá testováním a zpětné vazby z implementační části bakalářské práce. Cílem tohoto testování bylo odhalit nekorektní chování aplikace a dát možnost zájemcům v komunitě a potenciálním uživatelům si aplikaci předčasně vyzkoušet. Celý proces byl rozdělený do několika částí a to tzv. code review (zkoumání programu programátorem obeznámeným s danou tématikou) a v závislosti na předchozím bodu vytvoření balíčku a následné publikování viz Distribuce Qt aplikací. Jakmile byl vytvořený balíček, přišlo na řadu testování aplikace programátorem spolu s grafickým návrhářem a posléze uživateli.

7.1 Testování programátorem

V průběhu implementace byla aplikace neustále testována, díky tomu se předešlo mnoha budoucím problémům, které by mohly potenciálně nastat. Při každé nové funkcionalitě nebo změně aplikace byl při nahrání na server GitHub s technologií Git, přezkoumán zdrojový kód (code review) a vyzkoušena funkcionalita. Testování funkcionality a code review prováděl konzultant této práce, pan *Bc. Jan Grulich*. Testování aplikace po vzhledové stránce prováděl hlavní designer a autor mockupů aplikace.

V případě možného vylepšení nebo chyby v aplikaci byl vytvořen patřičně označený bug v sekci Issues na GitHubu pro budoucí opravu. Při veškerém testování byl použitý nejnovější build aplikace. Mezi nejčastější opravy po vizuální stránce se jednalo o rozložení elementů aplikace spolu s formátováním textu. Mezi funkcionální opravy stránky patřily typicky opravy stavů, které nastavovaly vlastnosti ostatních elementů, jako byly například tlačítka nebo texty.

Příklady chyb při implementaci této práce v průběhu vývoje:

- Při dokončení zápisu zůstane komponenta `progressBar` prázdná a nikoliv na 100%
- Problematické chování aplikace při změně její velikosti na monitoru s větším rozlišením
- Nepřesné popisky tlačítek při dokončení zápisu souboru .iso a obnově disku
- Špatná výchozí animace při přecházení mezi jednotlivými stránkami způsobovala tzv. ghosting effect[1]

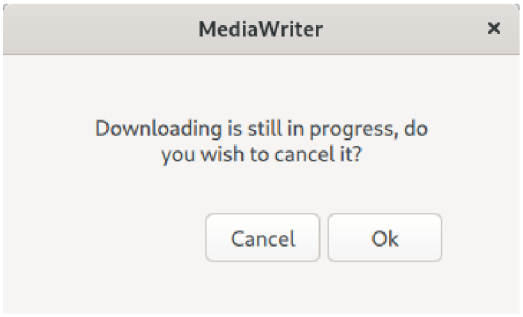
[1] Efekt připomínající stín, který se vyskytuje za pohybujícím se objektem

Úprava dialogu

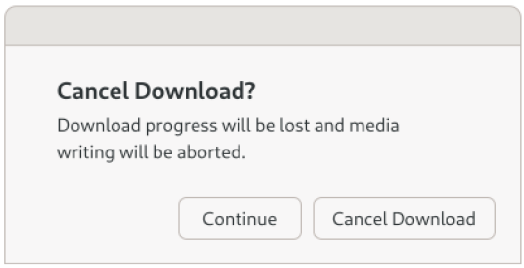
Po testování grafickým designerem byl jeden z vytvořených bugů, úprava layoutu a fomárování textu u cancelDialogu. Mezi problémy tohoto dialogu patřilo například:

- Úprava nadpisu „MediaWriter“, který byl neinformující
- U dialogu je zavírací symbol v rámu okna, což je nestandardní u Qt Quick dialogů
- Nepřítomný nadpis uvnitř dialogu, který je těžší pro čtení
- Zarovnání textu na střed oproti zarovnání tlačítek doprava

U vytvoření layoutu nastává problém z důvodu různého layoutu u nativních dialogů na Windows, macOS a Linuxu (stylu Gnome). Protože na Windows a macOS se používá zarovnání doleva u textu a doprava u tlačítek v potvrzovacích dialozích, byl navržen tento univerzální layout včetně popisků který vypadá následovně. Stejné rozložení dialogu bylo aplikováno i na aboutDialog.



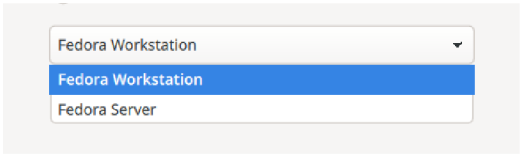
Obrázek 7.1: Předchozí verze dialogu



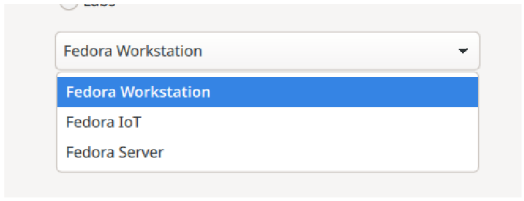
Obrázek 7.2: Opravený dialog

Oprava stylu Adwaita

Další potřebnou opravou byla implementace stylu Adwaita viz obrázek 3.5. Tímto bugem bylo pozadí přetékající kontejner při vybrání položky.



Obrázek 7.3: Předchozí verze combo boxu



Obrázek 7.4: Opravená verze combo boxu

Pro opravu tohoto stylu byla jako vzor použitá aplikace *gtk3-widget-factory* na operačním systému Linux která ve výchozím nastavení používá styl Adwaita. Tato aplikace slouží jako přehled všech dostupných typů widgetů a vizuálních efektů dostupných v gtk2 a gtk3 v jednom okně. Pomocí této aplikace je možné dobře demonstrovat vzhled i jiných stylů (Fusion, Material, aj.) nainstalované v systému.

7.2 Testování veřejností

Abych získal více zpětné vazby o vývoji také od zákazníků a nadšenců okolo Fedory, začal jsem psát blog[2]. V tomto blogu píši články o vývoji, jakmile je k dispozici větší update aplikace (typicky build na nový operační systém). Pomocí blogu dávám možnost komunitě vyjádřit svůj názor o vývoji, podělit se o zajímavých, či problematických částech vývoje a pomoci odhalit případné nedostatky aplikace. Abych zvýšil známost mého blogu, stal jsem se členem komunity Fedora Planet[3]. Na tomto fóru sdílí členové této komunity různé informace okolo distribuce Fedory, stejně jako blogy jednotlivých členů komunity nebo jejich články. Všichni testující uživatelé mají stejně jako programátoři možnost vytvořit bug, pokud by s aplikací nebylo něco v pořádku.

Další částí testování veřejností bylo otestování aplikace uživateli s cílem zjistit přehlednost a jednoduchost používání aplikace. Toto testování bylo provedeno se 4 uživateli a vždy byly provedeny celkem 4 scénáře. Každý z těchto scénářů měl vyzkoušet určitou část aplikace, které dohromady tvoří téměř celou aplikaci. Cílem tohoto testování bylo zjistit, zda je aplikace v daném scénáři intuitivní na používání.

Uživatelé byli vybráni tak, aby pokrývali všechny možné skupiny lidí, kteří by mohli aplikaci používat. První dva uživatelé byli mí vrstevníci, kteří nemají IT vzdělání, zato ale neměli s překladem anglických popisků problémy. Dalším uživatelem byl vystudovaný Ing. IT, který měl reprezentovat nejpotenciálnějšího uživatele. Posledním uživatelem byl člověk bez IT vzdělání a ne tak dobrou angličtinou, z toho důvodu jsem musel určité pasáže aplikace uživateli pomoci přeložit.

Před začátkem testování nebyla aplikace ukázána uživateli, díky tomu lze zjistit jak moc intuitivní je celá aplikace jako celek.

7.2.1 První scénář

Cílem prvního scénáře bylo vyzkoušet stažení a zápis Fedory v dané variantě, verzi a architektuře. Do celkového času stráveným nad tímto úkolem se nepočítá čas strávený zapisováním a stahováním souboru (z důvodu veliké závislosti aplikace na internetovém připojení a rychlostí USB disku).

1. Vyberte 1. možnost „Download automatically“ a přesuňte se na další stránku
2. Vyberte variantu Fedory s názvem „SoaS“ v kategorii „Spins“
3. Na další stránce vyberte verzi 35 a hardwarovou architekturu „ARM v7“ a začněte zapisovat

Na základě operací s aplikací byly uživateli po každé úloze pokládány otázky se zaměřením právě na intuitivnost a rychlost prováděné úlohy.

Rychlost prováděného scénáře proběhlo u všech uživatelů v řádů desítek sekund a nikdy nepřekročilo více než jednu minutu. První uživatel měl problém s výběrem varianty Fedory, nevšiml si, že se změnil obsah elementu combo box. Z toho důvodu by se dala nejspíše přidat animace, nebo nějaký druh barevného zvýraznění této komponenty při změně obsahu. Třetí uživatel měl připomínku, že by bylo dobré, aby se při vracení se na úvodní stránku („MainPage“) nenastavoval implicitně výběr hlavní operace na první možnost (Download

[2] https://egastablog.wordpress.com/
[3] http://fedoraplanet.org/

automatically), ale na tu vybranou uživatelem. Druhý a čtvrtý uživatel neměli s tímto scénářem žádný problém či připomínky.

7.2.2 Druhý scénář

Druhý scénář měl za úkol zjistit, zda je pohodlné zapsat již stažený soubor s Fedorou a zároveň tento soubor následně smazat. Podmínky byly stejné jako u předchozího scénáře.

1. Vyberte možnost „Select .iso file“ a přesuňte se na další stránku
2. Zde vyberte soubor, který jste stáhli v předchozím kroku (SoaS) pomocí tlačítka „Select...“
3. Zatrhněte možnost pro smazání po zápise a začnete zapisovat

Tento testovací scénář nedělal žádnému z uživatelů jakékoliv problémy. Čtvrtý uživatel měl připomínku, že by bylo dobré zkusit implementovat „informační tabulky“ při najetí na daný element. Tyto tabulky by mohly popisovat prováděnou operaci více do podrobnosti. A zároveň přidat malé šipky k tlačítku „Next“ a „Previous“. Tento krok bude nejspíše zbytečný jakmile bude aplikace přeložená do českého jazyka.

7.2.3 Třetí scénář

Třetí scénář měl za úkol vyzkoušet intuitivnost dialogů, které se při implementaci poměrně často měnily. Pro provedení tohoto scénáře bylo využito předchozího scénáře, kdy je v poslední úloze zapisována Fedora na disk. Ostatní podmínky zůstávají stejné, jako u předchozích scénářů.

1. Při zapisování souboru z předchozího úkolu, otevřete dialog pro zrušení zápisu
2. Tento dialog zkuste zavřít, aby se pokračovalo v zápise a poté zkuste dialog opět otevřít a zrušit zápis
3. Po zrušení zápisu otevřete dialog „About“, následně zkuste provést operaci s hlavní aplikací a poté tento dialog opět zavřete

Navržený testovací scénář měl vyzkoušet, zda byly vhodně zvolené popisky tlačítek u dialogů a zároveň dá-li se manipulovat s aplikací, je li otevřený dialog.

Ve třetím scénáři se oproti předchozím scénářům vyskytlo více problémů, připomínek a poznatků. Také zároveň celý scénář trval o něco déle než předchozí scénáře, toto bylo dáno nutností pochopit dialog pro rušení (CancelDialog) a napsaný scénář. Uživatelé zmiňovali, že neví co je to dialog, proto mohly být následující operace uživatelů touto skutečností ovlivněny. První a druhý uživatel měli problém správně vybrat požadovanou možnost při rušení zápisu. Třetí uživatel upozornil na to, že by bylo nejspíše dobré upravit text v Cancel Dialogu z „Continue“ na „Continue Write“ nebo „Continue Download“ podle aktuálně prováděné operace. Toto by pravděpodobně předešlo problémům u prvního a druhého uživatele ve vybrání správné možnosti. U tohoto scénáře jsem se také potýkal s problémem, že uživatelé často odklikávají informační dialogy automaticky, aniž by si jej vůbec přečetli.

7.2.4 Čtvrtý scénář

Poslední navržený scénář měl za úkol otestovat obnovu USB disku. Tato funkce byla opravována na operačním systému Windows, z toho důvodu jsem se rozhodl provést toto testování s uživateli právě na tomto operačním systému.

1. Připojte USB disk k počítači a zvolte na úvodní stránce aplikace poslední možnost s obnovou disku (Restore)
2. Pokračujte dále a zkuste disk obnovit. Následně zkontrolujte zda je disk opravdu prázdný

Při startu tohoto scénáře se bude muset uživatel navigovat na úvodní stránku, díky tomu zjistíme zda se uživatel dokáže orientovat v aplikaci.

Tento poslední scénář proběhl jako jediný bez jakýchkoliv problémů nebo připomínek. Všem uživatelům tato operace trvala desítky sekund.

7.3 Zpětná vazba - Zimní semestr

V průběhu zimního semestru bylo prováděno testování programátorem, na konci semestru byl vytvořen testovací build v podobě Flatpaku, který byl prostřednictvím blogu poskytnut zájemcům v komunitě. Aplikace je na konci semestru a po testování uživatelem schopná:

- Zatím je možné testovat pouze na operačním systému Linux
- Aplikace je téměř kompletní z hlediska hlavní funkcionality
  - Umí automaticky stáhnout ISO z internetu
  - Dokáže zapsat toto ISO na USB disk
  - Obnoví bootovací USB do předchozího stavu
- Stále potřebuje spoustu oprav, aby byla plynulejší, intuitivnější a robustnější

Stav aplikace včetně jejího rozhraní je nutné konzultovat s grafickým designerem a doladit nedokonalosti, jelikož byla vyvíjena podle jednoduchého a základního mockupu.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo vytvořit nové GUI nástroje Fedora Media Writer, tak aby podporovalo nativní styly na operačních systémech Windows a macOS. Upravit současný styl Adwaita na Linuxu včetně instalátorů na všechny platformy.

Podařilo se mi vytvořit aplikaci, která zcela splňuje zadání této práce. Zároveň jsem opravil různé nedostatky předchozí verze této aplikace, jako je obnova disku na operačním systému Windows.

Před započetím práce bylo potřeba zorientovat se ve velkém open source projektu, jako je Fedora Media Writer. Nastudovat a seznámit se s grafickým frameworkem Qt, kterému byla věnována celá jedna kapitola této práce viz 4. V této kapitole byla poměrně veliká část kapitoly věnována distribuci Qt aplikací, kde jsem se potřeboval seznámit jak s tvorbou balíčků na různé platformy včetně použitých nástrojů, tak i vestavěnou funkcí CI na serveru GitHub.

Ve druhé kapitole byly rozebrány operační systémy, tedy to, co se ve formě souboru .iso zapisuje na USB disk pomocí Fedora Media Writeru. V této kapitole byly zároveň porovnány jednotlivé hardwarové požadavky na daný operační systém.

Kapitola 5 se zaměřuje na současnou a na novou verzi nástroje Fedora Media Writer. Tato kapitola rozebírá současný vzhled aplikace zároveň s jejími nedostatky. Část kapitoly je věnována teorii ohledně oddílů na disku, které byly problematické na platformě Windows. Poslední část porovnává konkurenční nástroj Rufus dostupný na operačním systému Windows s nástrojem FMW, zároveň je zde popsána struktura .iso souboru obsahující Linuxovou distribuci Fedoru.

Kapitola Implementace popisuje jak nový vzhled aplikace, tak i její funkční část využívající nástroj helper, který je součástí nástroje FMW.

Poslední kapitola této bakalářské práce zmiňuje výsledky testování a zpětnou vazbu jak od grafických návrhářů, tak i od obyčejných uživatelů.

Prací na tomto projektu jsem získal spoustu zkušeností z různých odvětví, které bylo nutné nastudovat a pochopit do hloubky včetně všech závislostí, bez kterých by jednotlivé části nedávaly smysl. Díky těmto nastudovaným vědomostem jsem byl schopný implementovat výsledný styl včetně různých oprav nedostatků aplikace. Výsledkem je zdokonalená aplikace, která zapadá do ekosystému operačního systému na kterém běží.

8.1 Možné vylepšení aplikace

Na základě zpětné vazby od uživatelů byl návrh přidat při obnově disku možnost změnit název disku nebo na operačním systému Windows možnost přiřadit i nové písmeno u disku. Mezi vylepšení lze zařadit výsledky testování viz Testování veřejností. Dalším vylepšením by mohla být podpora procesorů Apple silicon (Procesory značky Apple, založené na ARM architektuře, které jsou záměnou od procesorů Intel od roku 2020). Toto vylepšení by pravděpodobně spočívalo v upravení současné architektury ARM64. Mezi tato vylepšení může spadat oprava současných chyb aplikace viz Současné nedostatky aplikace, kterých je v době psaní této práce tři a třicet.

Literatura

[1] Carrier, B. *File System Forensic Analysis*. 1. vyd. Pearson Education, 2005. ISBN 9780134439549.

[2] Company, Q. *New Features in Qt 6.0* [online]. 2020 [cit. 29.11.2021]. Dokumentace. Dostupné z: https://wiki.qt.io/New_Features_in_Qt_6.0.

[3] Dey, N. *Cross-Platform Development with Qt 6 and Modern C++: Design and build applications with modern graphical user interfaces without worrying about platform dependency*. 1. vyd. Packt Publishing, 2021. ISBN 9781800208858.

[4] Felix, M. *Monitorovací a zabezpečovací systém*. Brno, CZ, 2010. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[5] Foundation, F. S. *GNU Operating System* [online]. 2020 [cit. 30.11.2021]. Dostupné z: https://www.gnu.org/.

[6] GRULICH, J. *Nástroje pro práci s Google službami v KDE [online]*. 2012 [cit. 2022-01-04]. Bakalářská práce. Univerzita Palackého v Olomouci, Přírodovědecká fakultaOlomouc. Vedoucí práce Kühr, M. T. Dostupné z: https://theses.cz/id/jeffkv/.

[7] Hladík, D. *Prohlížeč obrázků pro desktopové prostředí KDE*. Brno, CZ, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/19447/.

[8] KDE. *KDevelop4/Manual* [online]. Srpen 2012 [cit. 26.4.2022]. Dostupné z: https://userbase.kde.org/KDevelop4/Manual.

[9] KDE. *KDevelop: A cross-platform IDE for C, C++, Python, QML/JavaScript and PHP* [online]. 2016 [cit. 26.4.2022]. Dostupné z: https://www.kdevelop.org/.

[10] Lazar, G. a Penea, R. *Mastering Qt 5: Create stunning cross-platform applications using C++ with Qt Widgets and QML with Qt Quick, 2nd Edition*. 2. vyd. Packt Publishing, 2018. ISBN 9781788993890.

[11] LLVM. *The LLVM Compiler Infrastructure* [online]. 2022 [cit. 26.4.2022]. Dostupné z: https://llvm.org/.

[12] Loeliger, J. a McCullough, M. *Version Control with Git: Powerful tools and techniques for collaborative software development*. 2. vyd. O'Reilly Media, 2012. ISBN 9781449345044.

[13] Lüddecke, D., Seidl, C., Schneider, J. a Schaefer, I. Modeling context-aware and intention-aware in-car infotainment systems: Concepts and modeling processes. *Software and systems modeling*. 3. vyd. Berlin/Heidelberg: Springer Berlin Heidelberg. 2016, sv. 17, č. 3, s. 973–987. ISSN 1619-1366. Dostupné z: https://doi.org/10.1007/s10270-016-0543-z.

[14] Marsh, J., Marzán, J. a Suchánek, T. *UX pro začátečníky: (rychlokurz - 100 lekcí)*. 1. vyd. Zoner Press, 2019. ISBN 9788074133978.

[15] Martin, K. a Hoffman, B. *Mastering CMake*. 3.1. Kitware Incorporated, 2015. ISBN 9781930934313.

[16] MOHAN, I. *OPERATING SYSTEMS*. 1. vyd. PHI Learning, 2013. ISBN 9788120347267.

[17] Omacht, M. *Grafický prohlížeč a jednoduchý editor ELF souboru*. Brno, CZ, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[18] OmniSci. *Graphical User Interface Definition* [online]. 2021 [cit. 16.11.2021]. Dostupné z: https://www.omnisci.com/technical-glossary/graphical-user-interface.

[19] Platt, D. *Tweak Your Mac Terminal: Command Line MacOS*. 1st edition. Berkeley, CA: Apress L. P, 2020. ISBN 1484261704.

[20] Scribble. *A NSIS Developer's Notebook*. 1. vyd. Independently Published, 2019. A Dev NB Blue and Orange Series. ISBN 9781704224145.

[21] Smyth, N. *Red Hat Enterprise Linux 8 Essentials: Learn to Install, Administer and Deploy RHEL 8 Systems*. 2. vyd. Payload Media, 2019. ISBN 9781951442040.

[22] Yādava, S. *Introduction To Client Sever Computing*. 1. vyd. New Age International Limited, 2009. ISBN 9788122426892.

Příloha A

Obsah SD karty

Přiložená SD karta obsahuje:

- Git repozitář obsahující zdrojové kódy této práce
- Návod jak tuto práci přeložit a nainstalovat
- Zdrojové kódy textu této bakalářské práce uložené v adresáři
- README.txt obsahující seznam upravovaných souborů
- Tuto technickou zprávu v podobě PDF souboru

Příloha B

Blog

New generation of Fedora MediaWriter

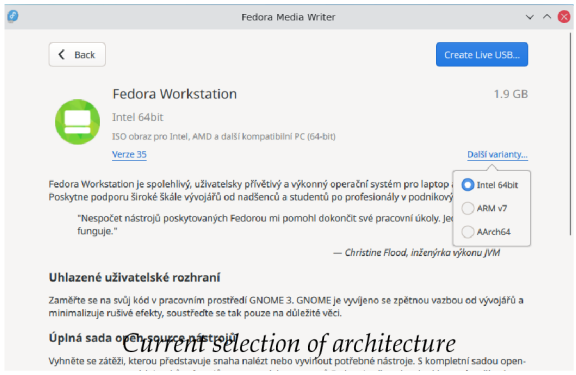
egasta Fedora December 10, 2021April 24, 2022 1 Minute

Welcome on my blog page, my name is Evžen Gasta. I live In Czech Republic and I'm studying FIT VUT in Brno.

I would like to share my news and updates about my development of new generation of Fedora MediaWriter (shortly FMW) as my Bachelor's thesis.

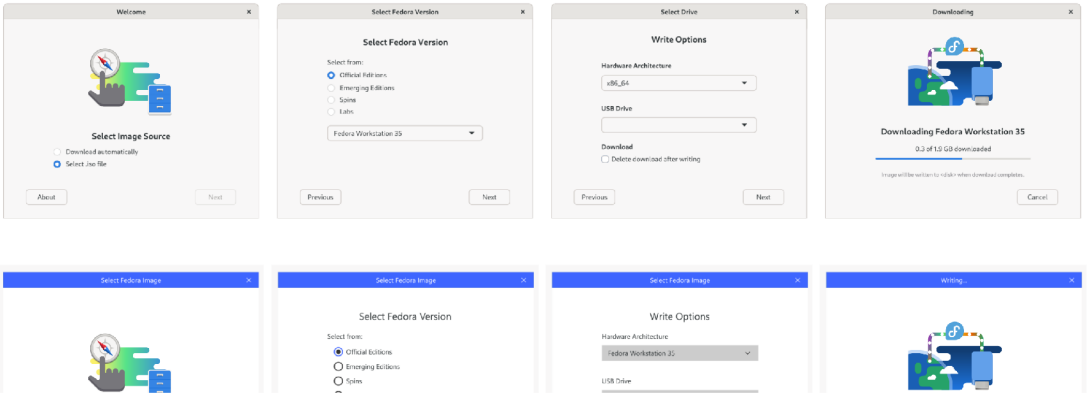
Current version of FMW is running on Qt5, because this version qml doesn't support native look on Mac and Windows, FMW looks on all platforms the same. At the beginning of the year 2021 was released new major version Qt. This version comes with many usefull news, such as support CMake as default build system, support of new version C++17 and support of native styles in qml…

Current Main Page FMW



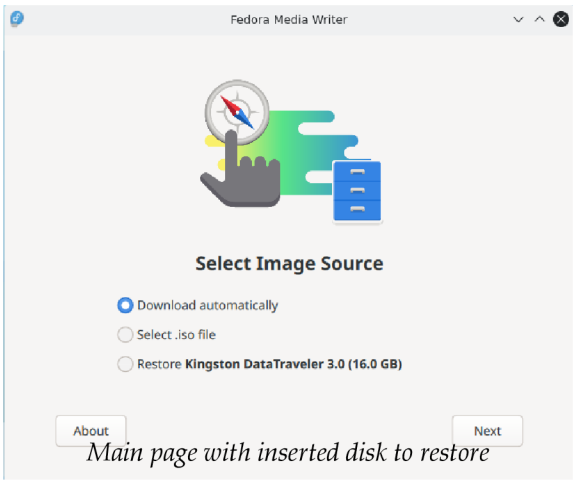
Current selection of architecture

As result new generation of FMW will run on Qt6 and will come with Adwaita theme only for Linux. Mac and Windows will be used native styles of Controls. Probably also fix some current issues after returning back full functionality later.

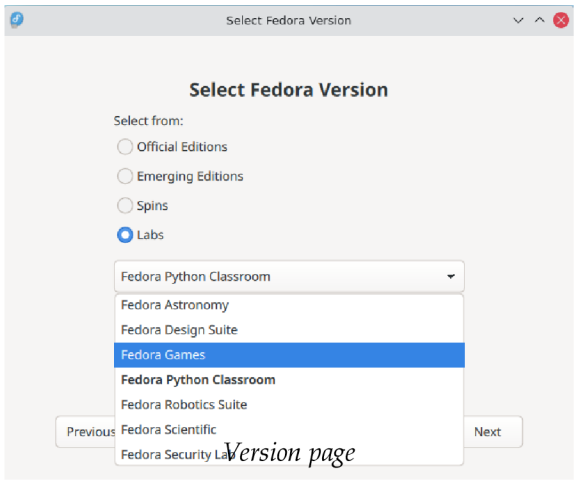


Mockups of new FMW

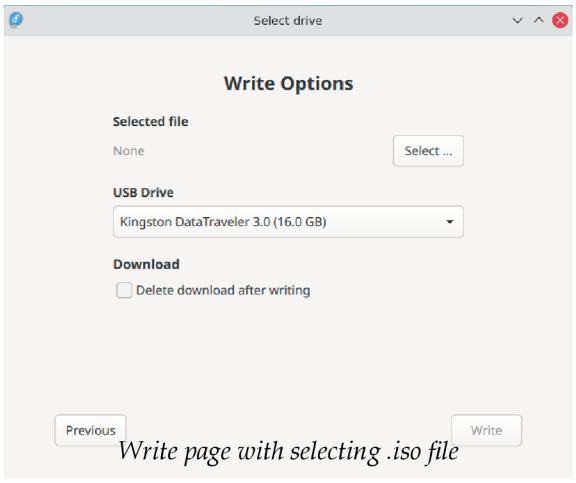
I have started development already and at the time of writing, FMW has prototype look and is able to restore drive with live Fedora. Also can write selected .iso file to drive and download file, on the other hand you can't see progress of writing or download for now. After new year should be available preview version on Linux.



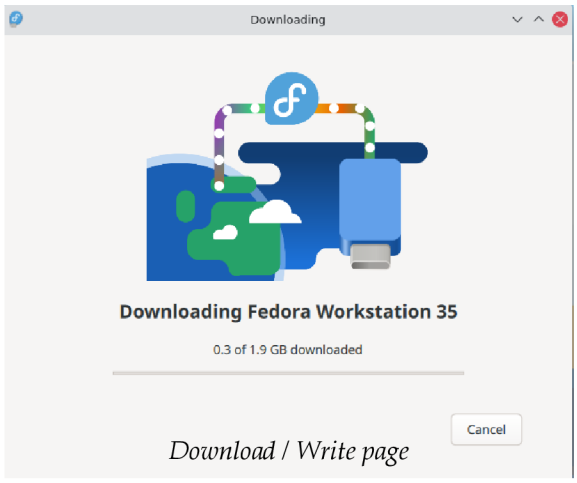
Main page with inserted disk to restore



Version page



Write page with selecting .iso file



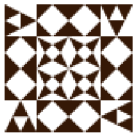
Download / Write page

In college, we usually don't develop such a complex aplications and if we do so, it is from scratch. So, when I started this project, it was quite confusing to orient in. But now I'm quite enjoying development of qml, it is fast and easy to learn and doesn't need to be compiled, like C, or Java… On the other hand integrating with C++ especially understanding, which function is usefull can be sometimes more time consuming, than I would like.

The final version, with full functionality should be finished by May. If the development goes well, I'll try update drive restoring on Windows. Because FMW is currently used mostly on Windows.

Stay tuned for more updates coming soon.

Published by egasta



Live in Czech Republic Study at FIT VUT in Brno View all posts by egasta

One thought on “New generation of Fedora MediaWriter”

Arsi says:
December 14, 2021 at 1:46 pm | Edit
Beautiful, can’t wait for the release!

↳ Reply

Last Modified:
05/03/2022 13:18:15
(timezone is UTC)

Updates on the new generation of Fedora MediaWriter

egasta Fedora March 14, 2022April 17, 2022 2 Minutes

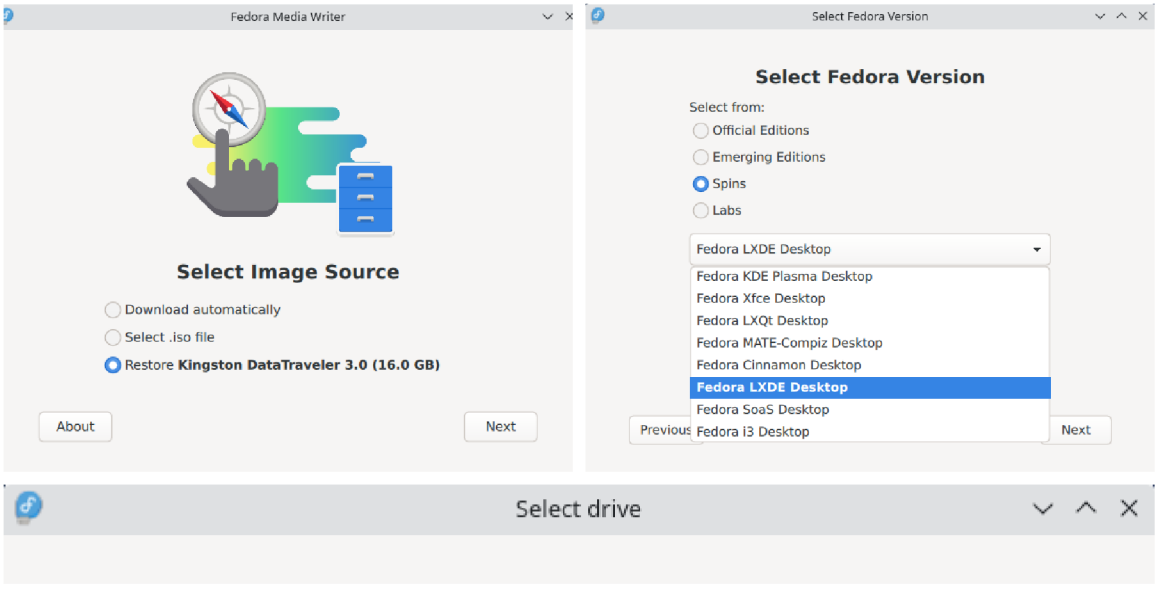
In the past few months I have been developing new generation of Fedora New generation of FMW with a new UI written in Qt6 which will use native QtQuick styles for Windows and MacOS. At this point I have a fully functional application with all the features from the current version.

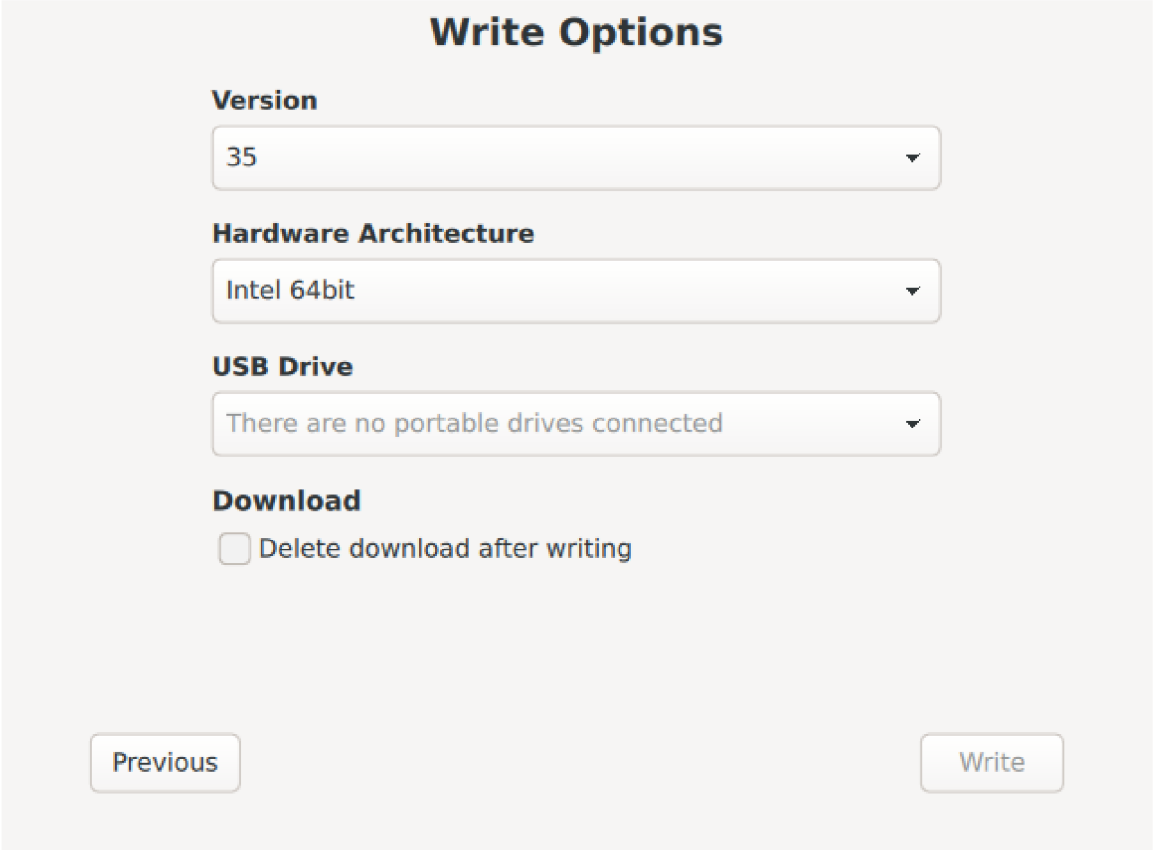
The application can be now build for Windows and Linux. Linux builds are also available as Flatpak for testing pourpose. Bare in mind this is still not the final version and there still might be some issues.

To develop new generation of FMW I had to learn, rework or update many things. To develop a new generation of FMW I had to learn, rework or update many things. A lot of them I saw for the first time like a complex project, QML, CMake, Qt… First of all I've started removing deprecated code that is no longer supported in Qt6 and made sure FMW can be built. After that I could start working on QML. I've started making pages and gradually adding basic functionality step by step.

Flatpak

For testing I've learned what is Flatpak and how Flatpak works. I've also learned how to create Flatpak using manifest file, this file was also needed to be updated. Thanks to GitHub CI we have available a test build made after every commit.



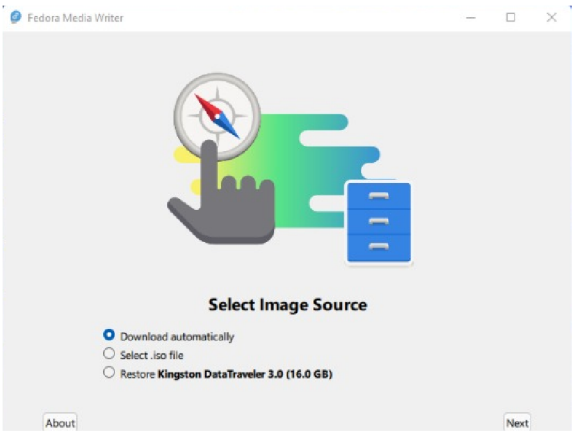


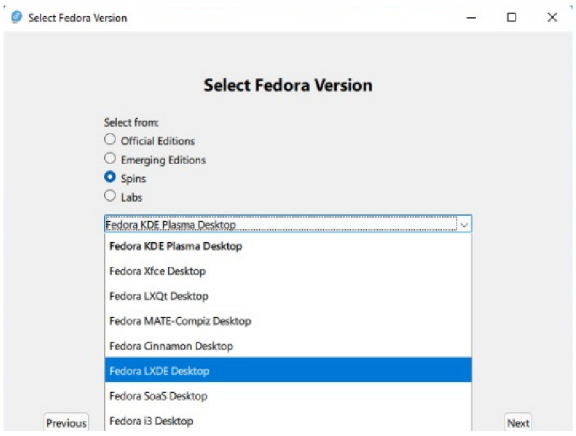
Flatpak

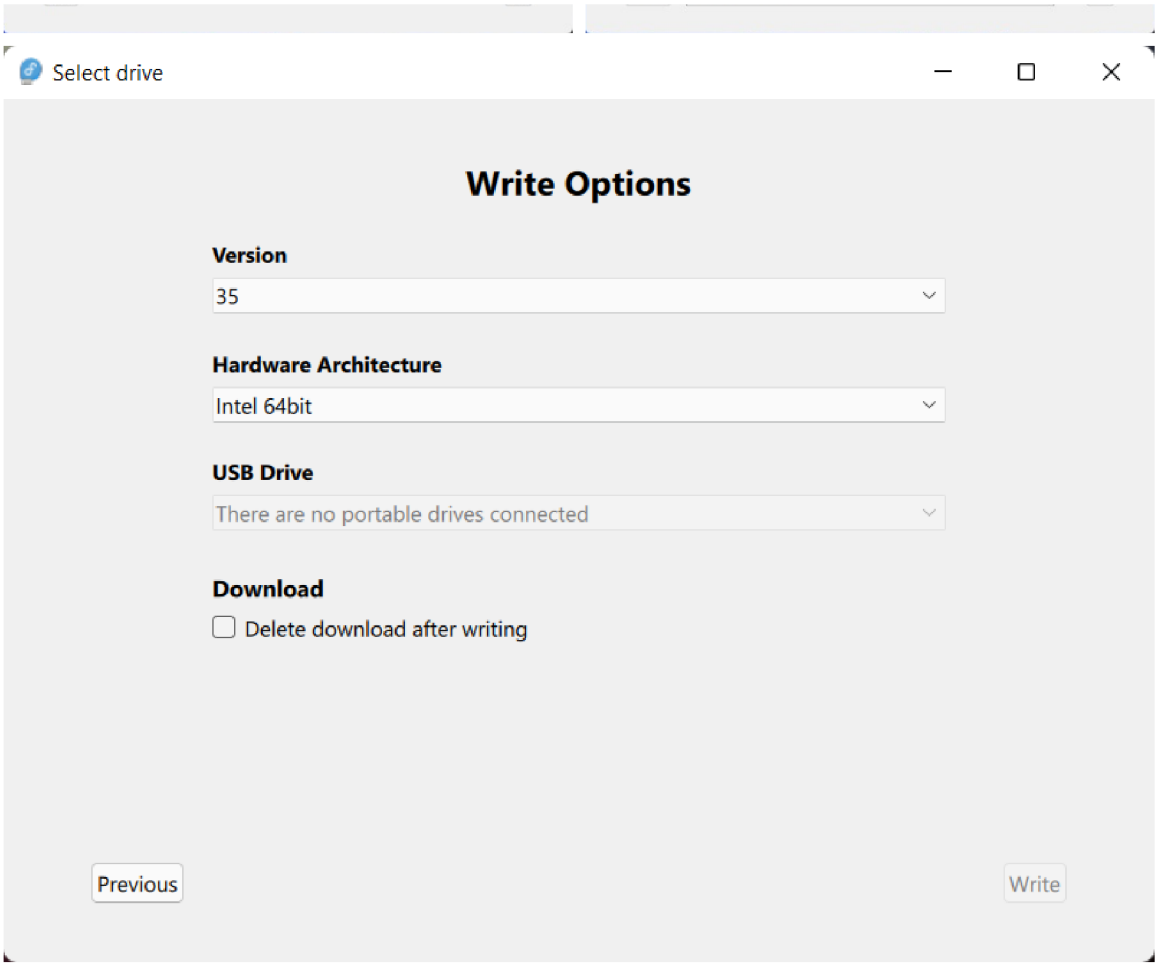
Windows

To develop Windows build I had to update the current script to support Qt6. That means updating qmake to cmake and removing all deprecated things. Also remove Adwaita theme, which is no longer used in FMW build on Windows. I also needed to copy all dependencies to create an installer for Windows.

Testing was pretty difficult for me, because I'm making the Windows builds on Linux and debugging was possible only in QtCreator on Windows. I had to boot multiple times from Linux to Windows and the other way around.







Windows

Windows version still needs some adjustment and fixes, such as restoring USB drive. For that reason we have available only the Linux version for testing, which has full functionality. I would be happy if you give me any feedback either (https://github.com/FedoraQt/MediaWriter/releases/tag/latest)here, or on github in the issues (https://github.com/gastoner/MediaWriter/issues) mentioning you use the nextgen version. You can get the development version here (https://github.com/FedoraQt/MediaWriter/releases) in the releases section.

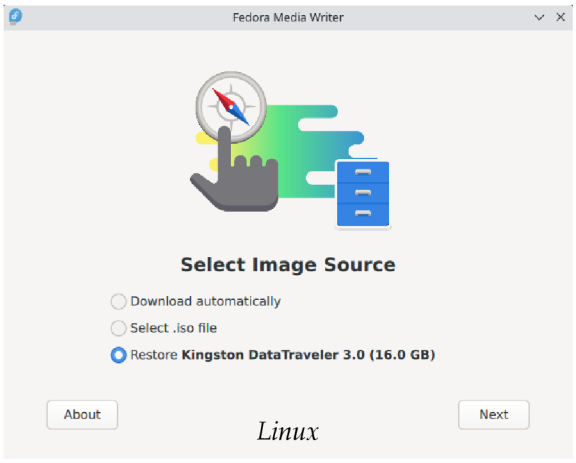
Published by egasta

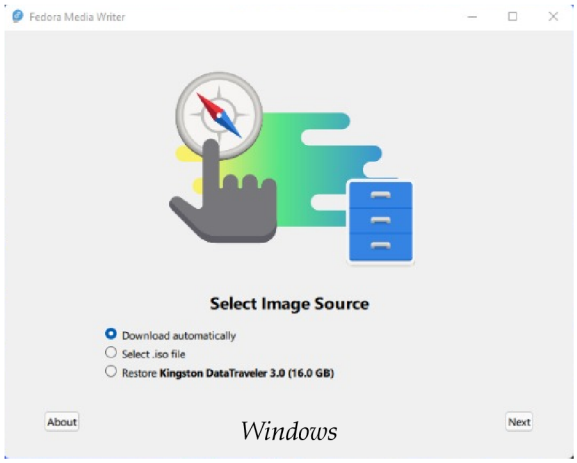
Live in Czech Republic Study at FIT VUT in Brno View all posts by egasta

Fedora MediaWriter (nextgen) on MacOS

egasta Fedora April 19, 2022April 19, 2022 2 Minutes

Hi, for the past few months I have been working on MacOS build and fixing USB drive restoration on Windows as well as fixing few design bugs. I fixed overflowing highlighted option background in combobox used in Adwaita style on Linux. Next design fix was updating animations when changing pages. Now when we build the new FMW on all platforms, I can say I like MacOS and Linux versions a lot more than Windows version. I don't like design of buttons. What about your opinion?





MacOS

Windows

On Windows I have fixed the problem with USB drive restoration. This was present for a long period of time. Problem was caused by corruption of the partition table after the ISO file was written onto a USB drive and utility diskpart was not able to work with this drive anymore. This was quite tricky to figure out, but after all the fix was quite easy. I had to recreate new partition table using command "convert gpt" which creates new partition table on USB drive. For backwards compatibility with older systems and USB drives I had to run command "convert mbr". After that utility diskpart now can restore the USB drive to original state.

MacOS

Making build for MacOS was quite tricky, because I don't have any Apple device at home. So I had to use virtual machine (app sosumi available as snap) to try my changes I made to FMW. Even with this app I could not connect USB drive to properly try out full functionality of this new version. Thanks to my mentor and leader of my bachelor's thesis Jan Grulich (his blog (https://jgrulich.cz/) and twitter (https://twitter.com/JanGrulich)) for all the testing.

While updating CI manifest I had to make some tweaks to build.sh file, such as adjusting paths of few modules and libraries missed by macdeployqt or libraries that has to be copied manually.

For those of you who don't know what is macdeployqt… Its "The Mac deployment tool can be found in QTDIR/bin/macdeployqt. It is designed to automate the process of creating a deployable application bundle that contains the Qt libraries as private frameworks" definition of macdeployqt from Qt docs (https://doc.qt.io/qt-6/macos-deployment.html).

All available builds you can download from releases (https://github.com/FedoraQt/MediaWriter/releases) on GitHub, if there are any problems with the application, let me know either here or create bug in issues (https://github.com/FedoraQt/MediaWriter/issues) on GitHub.

Published by egasta

Live in Czech Republic Study at FIT VUT in Brno View all posts by egasta

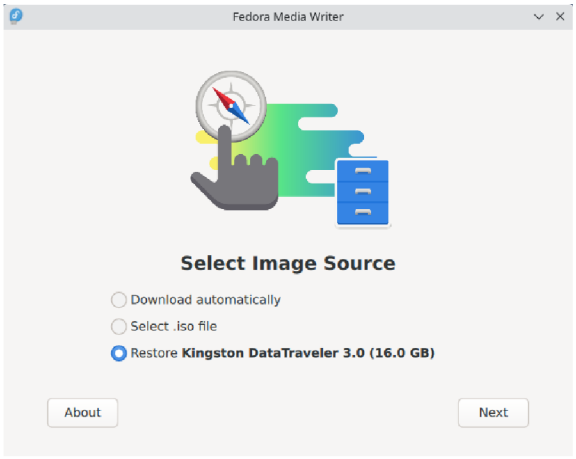
FMW is finished

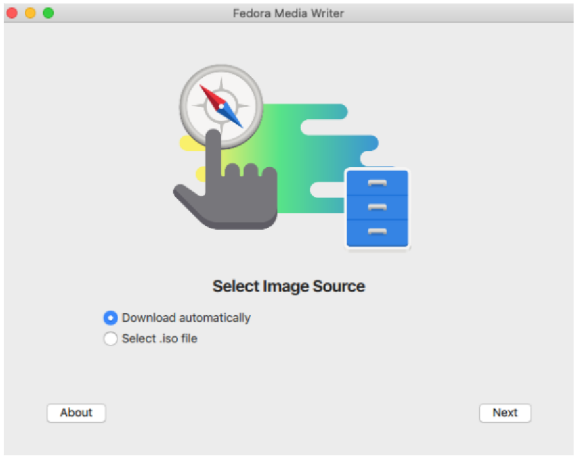
egasta Fedora May 4, 2022May 4, 2022 1 Minute

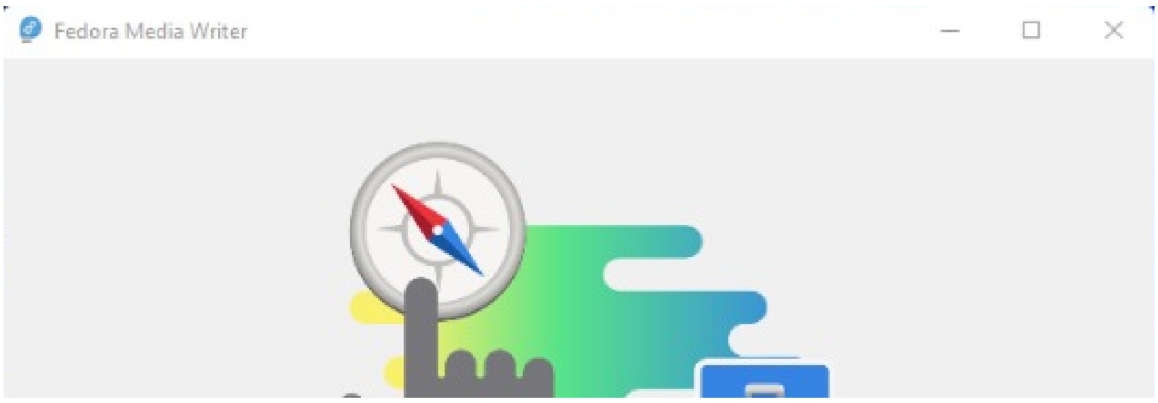
As all good things have to come to an end, my bachelor thesis has to end someday also. But I'm still looking forward to contributing to FMW, when are any updates or fixes needed. Official FMW 5.0.0 will be released soon. This was my first experience with an open source project and I liked it very much. I'm looking forward to start working on new open source projects in the future.

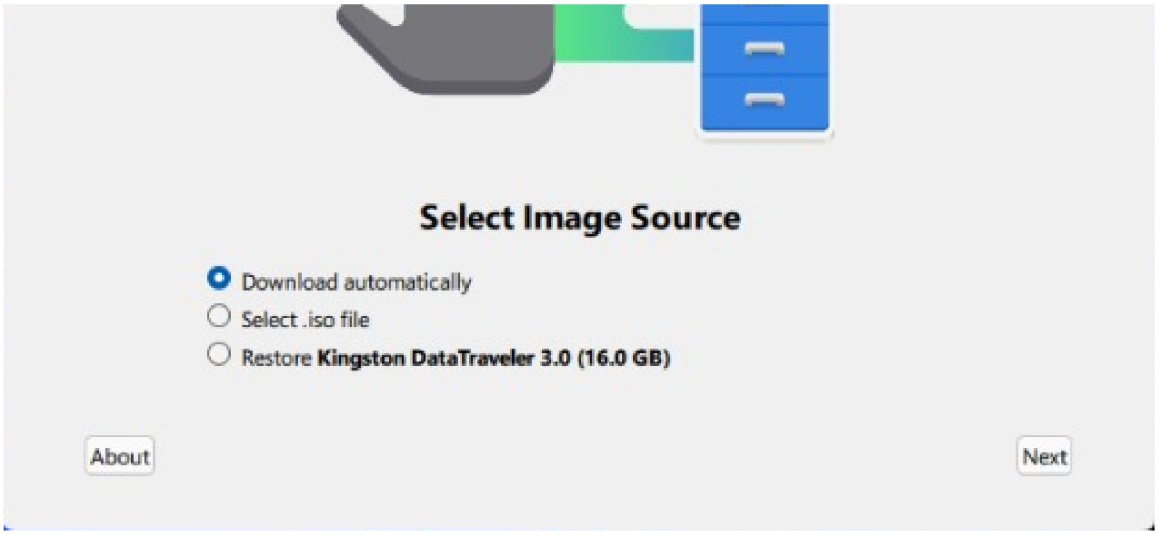
This project gave me a lot of experience such as learning how to deploy Qt applications on various operating systems, and how the structure of applications looks. I have learned the new programming language QML and how to use CMake in open source project. Another big experience for me is understanding how GitHub CI works. This was used to automatically create builds for various systems. I have practiced many options and advantages of Git (from the beginning I have been using only "git pull" with "git push").

As a result I was able to create new generation of FMW.









New generation of FMW

This result couldn't be achieved without my technical mentor Jan Grulich (his blog (https://jgrulich.cz/) and twitter (https://twitter.com/JanGrulich)). And also without my leader of my bachelor's thesis Dominika Regéciová (her blog (https://regeciova-dominika.medium.com/) and twitter (https://twitter.com/regeciovad)). So I would like to thank you both, for the opportunity, patients and overall help.

Published by egasta

Live in Czech Republic Study at FIT VUT in Brno View all posts by egasta